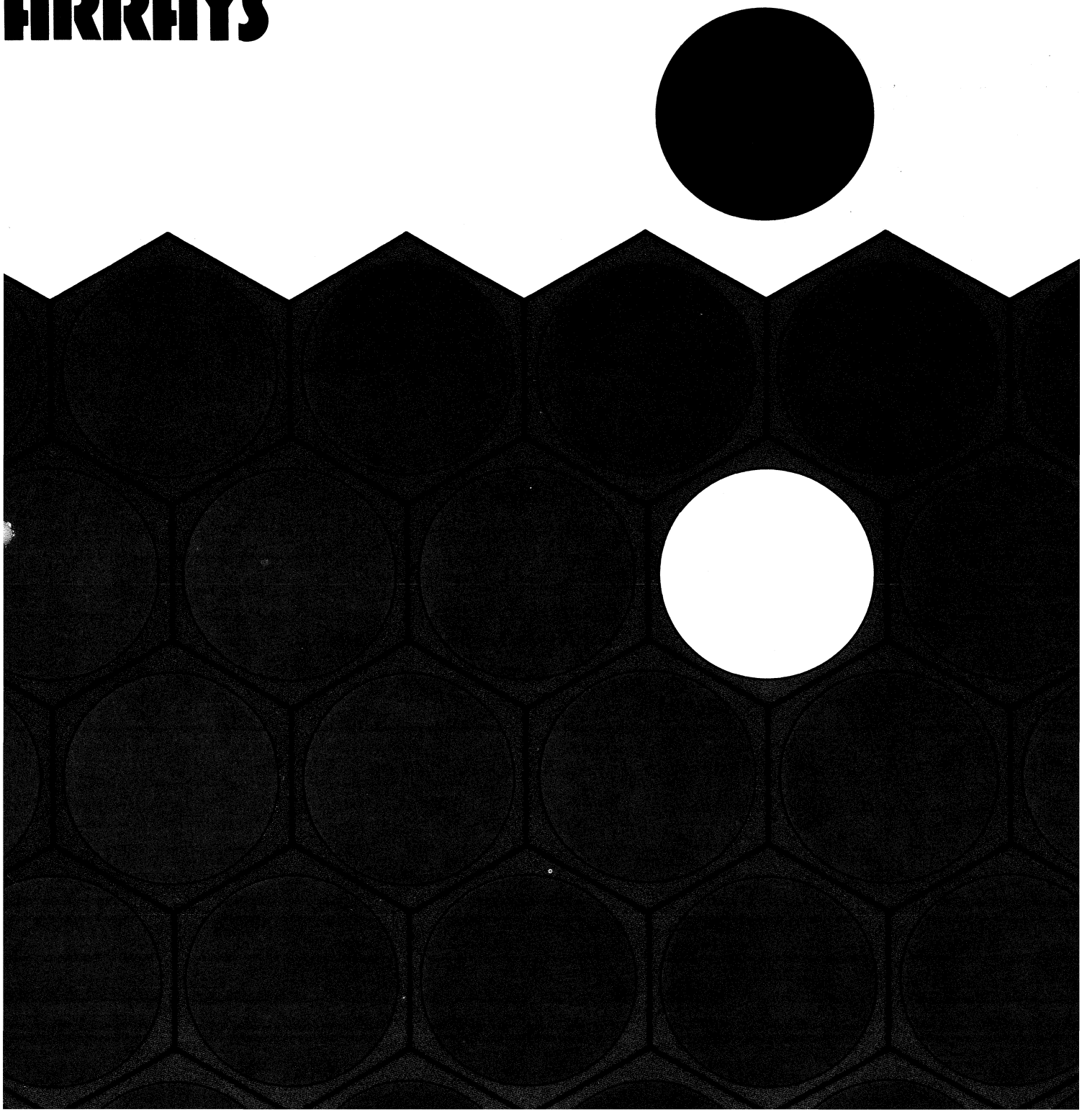
SIGNETICS FIELD PROGRAMMABLE LOGIC ARRAYS



SIGNETICS' FIELD PROGRAMMABLE LOGIC ARRAYS

Napoleone Cavlan
Manager, Advanced Products Marketing
February 1976

INTRODUCTION

Since the practical introduction of microprogramming in the last decade or so, microcode has progressively displaced random logic in step with the growing availability of user Programmable Read-Only Memories (PROMs). However, even with PROMs, designers soon realized that their rigid addressing structure made them unsuitable in a wide variety of applications which could greatly benefit from a structured logic approach.

Recently, microprocessors have provided a quantum jump in design flexibility in applications requiring about 30 IC packages, and beyond. When fewer packages are required, the inherent speed limitation, software requirements, and support circuitry of microprocessors place them out of range of a broad spectrum of applications.

These in general involve algorithms which require a high speed logic decision based on a large number of controlling variables. It is here that we step into the basic domain of Field Programmable Logic Arrays, encompassing applications in microprogramming, code conversion, random logic, look-up and decision tables, high speed character generators etc. Moreover, when combined with a few storage elements (flip-flops), FPLAs can implement powerful logic machines of the Mealy/Moore form for the realization of finite state sequential controllers for traffic, process, peripheral devices, and other similar applications.

DESCRIPTION

The 82S100 (Tri-State Outputs) and the 82S101 (Open Collector Outputs) are Bipolar Programmable Logic Arrays, containing 48 Product terms (AND terms), and 8 Sum terms (OR terms). Each OR term controls an output function which can be programmed either true active-High (Fp), or true active-Low ($F\overset{*}{p}$). The true state of each output function is activated by any logical combination of 16 input variables, or their complements, up to 48 terms. Both devices are field-programmable, which means that custom patterns are immediately available by following the fusing procedure outlined in this data sheet.

The 82S100 and 82S101 are fully TTL compatible, and include chip-enable control for expansion of input variables, and output inhibit. They feature either Open Collector or Tri-State outputs for ease of expansion of product terms and application in bus-organized systems.

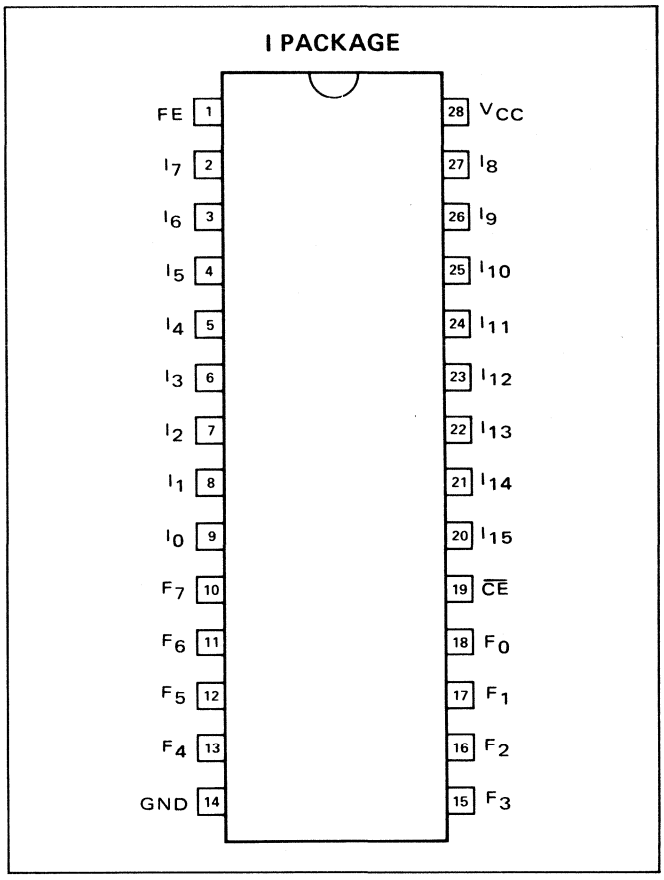
FEATURES

- **FIELD PROGRAMMABLE (Ni-Cr LINK)**
- **INPUT VARIABLES—16**
- **OUTPUT FUNCTIONS—8**
- **PRODUCT TERMS—48**
- **ADDRESS ACCESS TIME—50 ns, MAXIMUM**
- **POWER DISSIPATION—600mW, TYPICAL**
- **INPUT LOADING—(−100μA), MAXIMUM**
- **OUTPUT OPTION:**
  - **TRI-STATE OUTPUTS—82S100**
  - **OPEN COLLECTOR OUTPUTS—82S101**
- **OUTPUT DISABLE FUNCTION:**
  - **TRI-STATE—Hi-Z**
  - **OPEN COLLECTOR—Hi**
- **CERAMIC DIP**

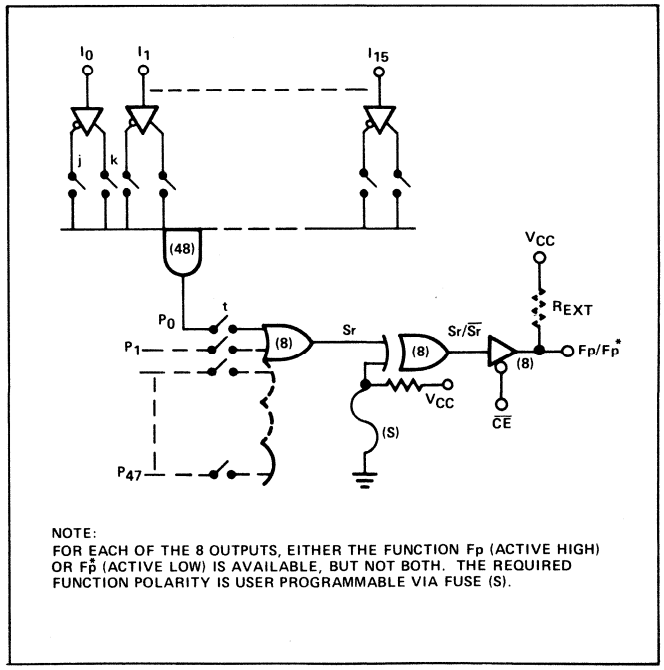
APPLICATIONS

LARGE READ ONLY MEMORY
RANDOM LOGIC
CODE CONVERSION
PERIPHERAL CONTROLLERS
LOOK-UP AND DECISION TABLES
MICROPROGRAMMING
ADDRESS MAPPING
CHARACTER GENERATORS
SEQUENTIAL CONTROLLERS

PIN CONFIGURATION

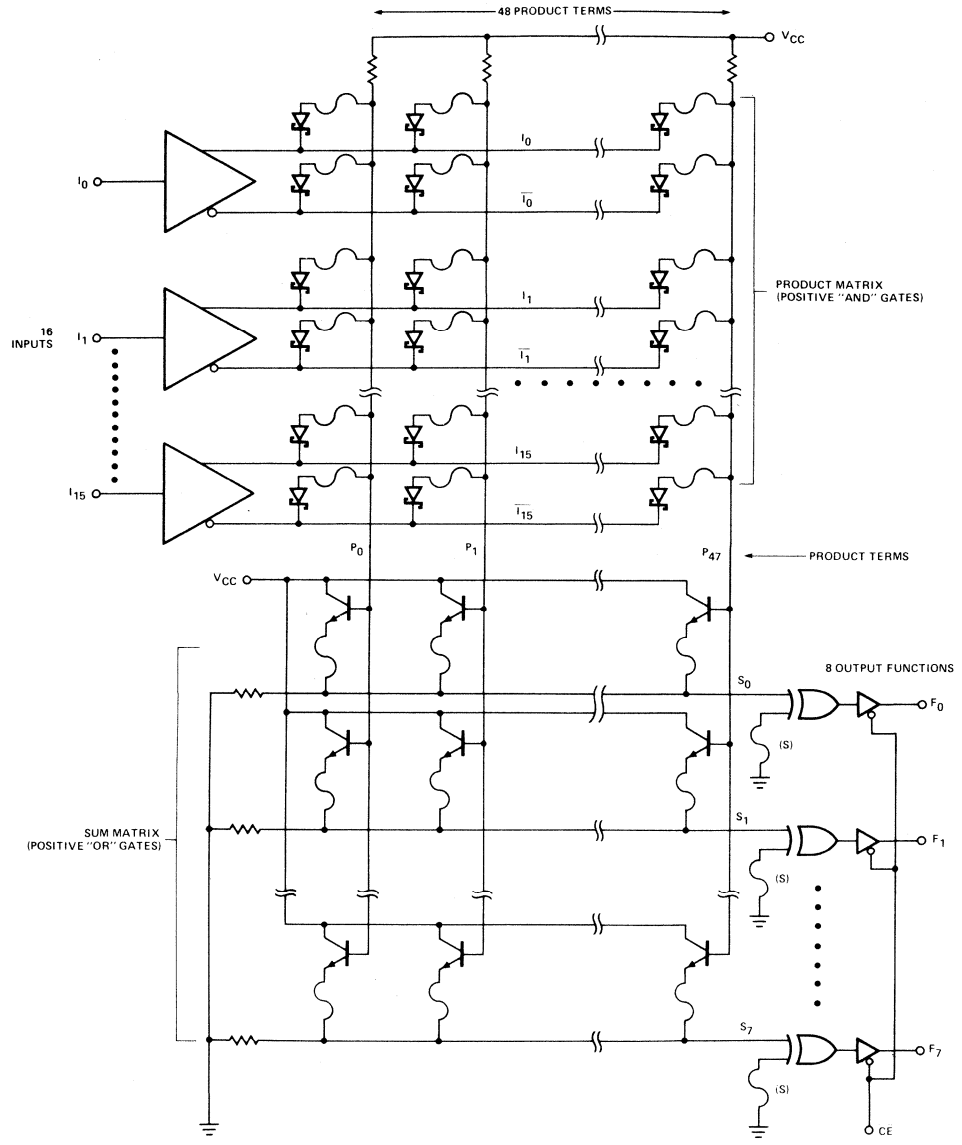


FPLA EQUIVALENT LOGIC PATH



NOTE:
FOR EACH OF THE 8 OUTPUTS, EITHER THE FUNCTION Fp (ACTIVE HIGH) OR $F\overset{*}{p}$ (ACTIVE LOW) IS AVAILABLE, BUT NOT BOTH. THE REQUIRED FUNCTION POLARITY IS USER PROGRAMMABLE VIA FUSE (S).

BLOCK DIAGRAM



TRUTH TABLE

LET:

$P_n = \Pi_0^{15} (k_m I_m + j_m \overline{I_m})$; k = 0, 1, X (Don't Care)

n = 0, 1, 2, ... , 47

$S_r = f\{\Sigma_0^{47} t_n P_n\}$; r ≡ p = 0, 1, 2, ... , 7

where:

Unprogrammed state : $j_m = k_m = 0$; $t_n = 1$

Programmed state : $j_m = \overline{k_m}$; $t_n = 0$

MODE	P_n	$\overline{CE}$	$S_r \stackrel{?}{=} f(P_n)$	F_p	F_p^*
Disabled (82S101)	X	1	X	1	1
Disabled (82S100)				Hi-Z	Hi-Z
Read	1	0	YES	1	0
	0	0		0	1
	X	0	NO	0	1

ABSOLUTE MAXIMUM RATINGS

PARAMETER[1]		RATING	UNIT
V_{CC}	Power Supply Voltage	+7	Vdc
V_{in}	Input Voltage	+5.5	Vdc
V_{OH}	High Level Output Voltage (82S101)	+5.5	Vdc
V_O	Off-State Output Voltage (82S100)	+5.5	Vdc
T_A	Operating Temperature Range	0° to +75°	°C
T_{stg}	Storage Temperature Range	−65° to +150°	°C

ELECTRICAL CHARACTERISTICS $0°C \leq T_A \leq 75°C$; $4.75V \leq V_{CC} \leq 5.25V$

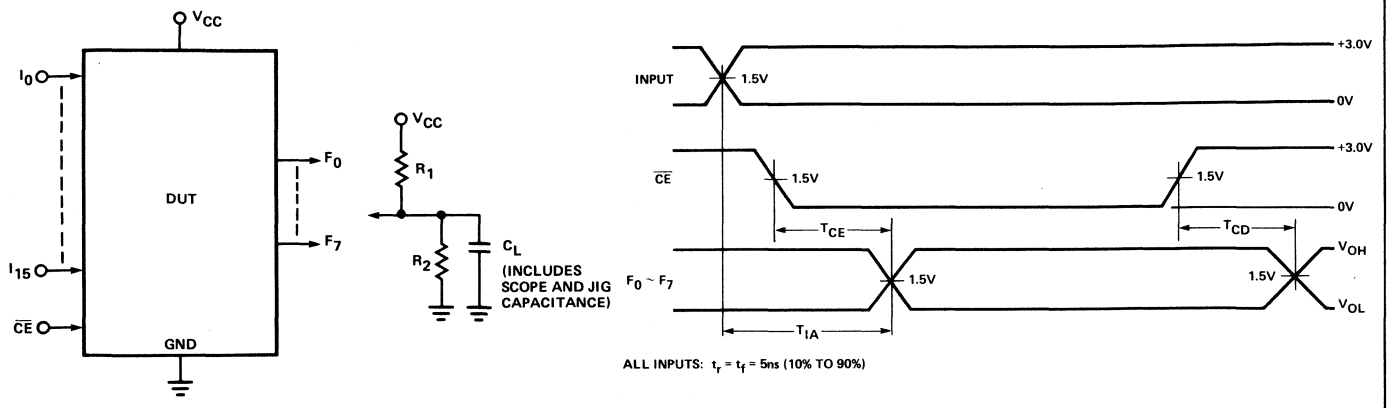
PARAMETER[1]		TEST CONDITIONS		LIMITS MIN	LIMITS TYP[2]	LIMITS MAX	UNIT	NOTES
V_{IH}	High-Level Input Voltage	V_{CC} = 5.25V			2		V	1
V_{IL}	Low-Level Input Voltage	V_{CC} = 4.75V				0.8	V	
V_{IC}	Input Clamp Voltage	V_{CC} = 4.75, I_{IN} = −18mA			0.8	1.2	V	1, 7
V_{OH}	High-Level Output Voltage (82S100)	V_{CC} = 4.75V, I_{OH} = −2mA		2.4			V	1, 5
V_{OL}	Low-Level Output Voltage	V_{CC} = 4.75V, I_{OL} = 9.6mA			0.35	0.45	V	1, 8
I_{OLK}	Output Leakage Current (82S101)	V_{CC} = 5.25V	V_{OUT} = 5.25V		1	40	μA	6
$I_{O(OFF)}$	Hi-Z State Output Current (82S100)	V_{CC} = 5.25V	V_{OUT} = 5.25V V_{OUT} = 0.45V		1 −1	40 −40	μA μA	6
I_{IH}	High-Level Input Current	V_{IN} = 5.5V			<1	25	μA	
I_{IL}	Low-Level Input Current	V_{IN} = 0.45V			−10	−100	μA	
I_{OS}	Short-Circuit Output Current (82S100)	V_{CC} = 5.25V, V_{OUT} = 0V		−20		−70	mA	3, 7
I_{CC}	V_{CC} Supply Current (82S100, 82S101)	V_{CC} = 5.25V			120	170	mA	4
C_{IN}	Input Capacitance	V_{CC} = 5.0V	V_{IN} = 2.0V		5		pF	
C_O	Output Capacitance		V_{OUT} = 2.0V		8		pF	6

SWITCHING CHARACTERISTICS $0°C \leq T_A \leq +75°C$, $4.75V \leq V_{CC} \leq 5.25V$

PARAMETER		TEST CONDITIONS	LIMITS MIN	LIMITS TYP[2]	LIMITS MAX	UNIT
Propagation Delay						
T_{IA}	Input to Output	C_L = 30pF		35	50	ns
T_{CD}	Chip Disable to Output	R_1 = 270		15	30	ns
T_{CE}	Chip Enable to Output	R_2 = 600		15	30	ns

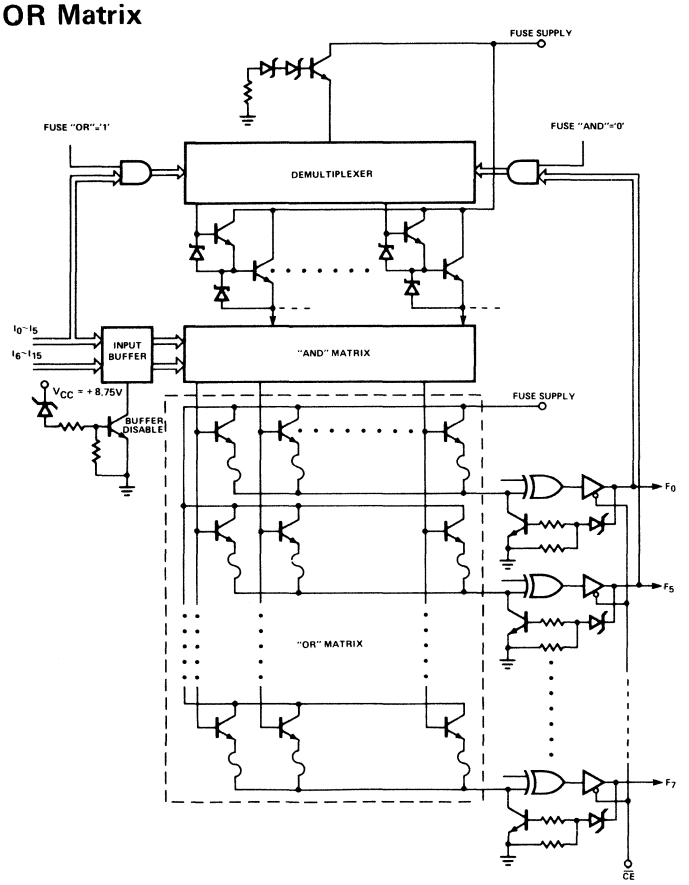
NOTES:
1. All voltage values are with respect to network ground terminal.
2. All typical values are at V_{CC} = 5V, T_A = 25°C.
3. Duration of short circuit should not exceed one second.
4. I_{CC} is measured with the chip enable input grounded, all other inputs at 4.5V and the outputs open.
5. Measured with V_{IL} applied to $\overline{CE}$ and a logic "1" stored.
6. Measured with V_{IH} applied to $\overline{CE}$.
7. Test each output one at the time.
8. Measured with a programmed logic condition for which the output under test is at a "0" logic level. Output sink current is supplied thru a resistor to V_{CC}.

AC TEST FIGURE AND WAVEFORMS

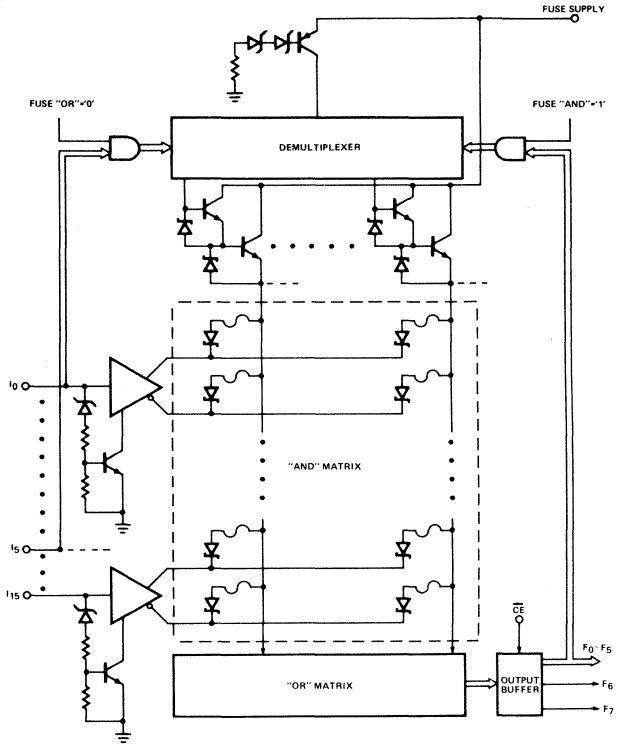


TYPICAL FUSING PATHS

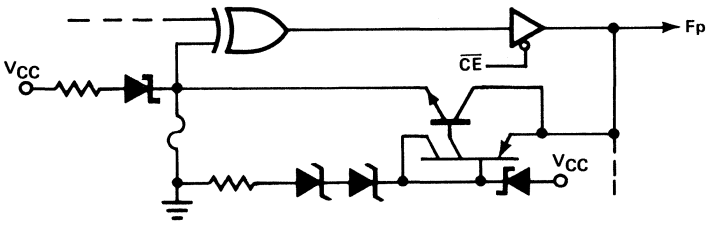
OR Matrix



AND Matrix



OUTPUT POLARITY

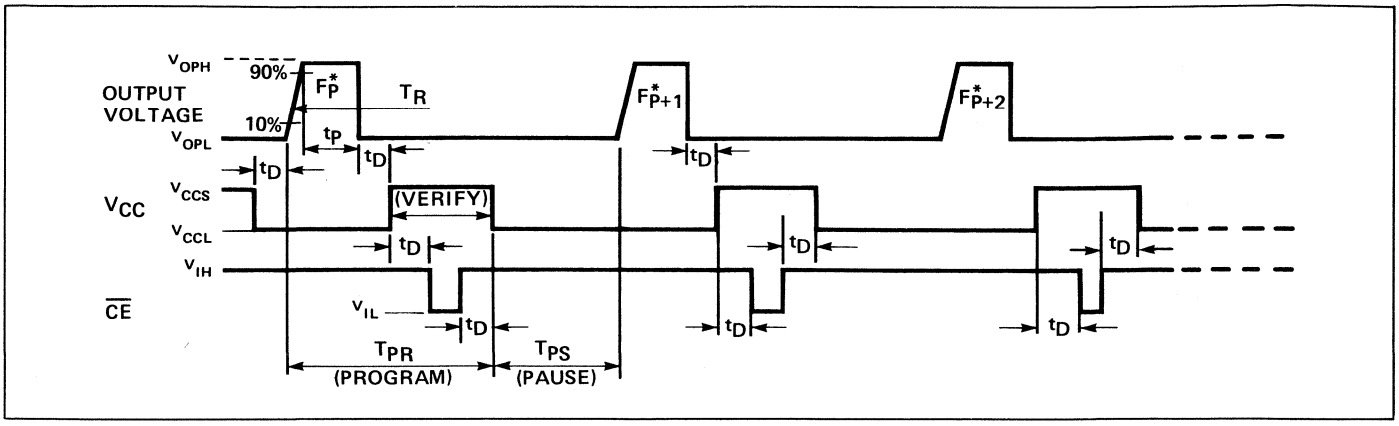


PROGRAMMING SPECIFICATIONS[4] (Testing of these limits may cause programming of device.) T_A = +25°C

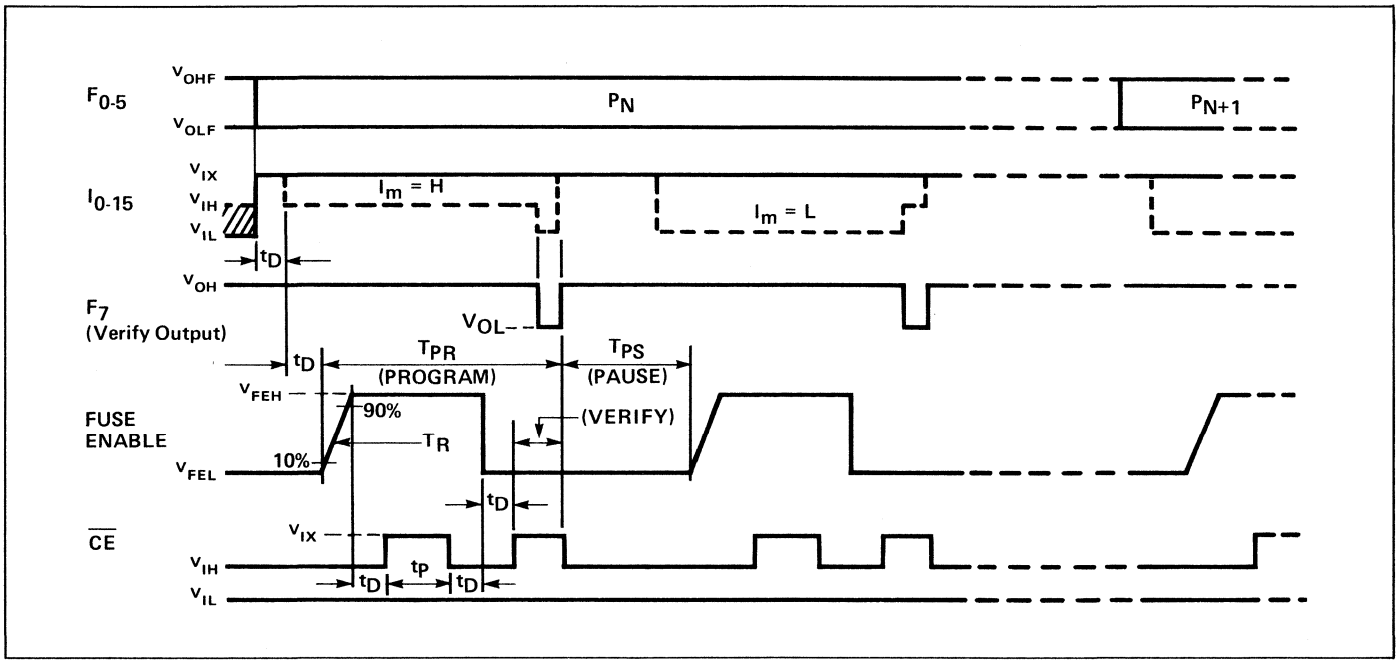
PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			MIN	TYP	MAX	
V_{CCS}[1]	V_{CC} Supply (Program "OR")	I_{CCS} = 550 mA, min. (Transient or steady state)	8.5	8.75	9.0	V
V_{CCL}	V_{CC} Supply (Program Output Polarity)		0	0.4	0.8	V
I_{CCS}	I_{CC} Limit (Program "OR")	V_{CCS} = +8.75 ±.25V	550		1,000	mA
V_{OPH}[2]	Output Voltage (Program Output Polarity)	I_{OPH} = 300 ±25mA	16.0	17.0	18.0	V
V_{OPL}	Output Voltage (Idle)		0	0.4	0.8	V
I_{OPH}	Output Current Limit (Program Output Polarity)	V_{OPH} = +17 ±1V	275	300	325	mA
V_{IH}	Input Voltage (Logic "1")		2.4		5.5	V
V_{IL}	Input Voltage (Logic "0")		0	0.4	0.8	V
I_{IH}	Input Current (Logic "1")	V_{IH} = +5.5V			50	μA
I_{IL}	Input Current (Logic "0")	V_{IL} = 0V			−500	μA
V_{OHF}	Forced Output (Logic "1")		2.4		5.5	V
V_{OLF}	Forced Output (Logic "0")		0	0.4	0.8	V
I_{OHF}	Output Current (Logic "1")	V_{OHF} = +5.5V			100	μA
I_{OLF}	Output Current (Logic "0")	V_{OLF} = 0V			−1	mA
V_{IX}	$\overline{CE}$ Program Enable Level		9.5	10	10.5	V
I_{IX1}	Input Variables Current	V_{IX} = +10V			2.5	mA
I_{IX2}	$\overline{CE}$ Input Current	V_{IX} = +10V			5.0	mA
V_{FEH}[2]	FE Supply (Program)	I_{FEH} = 300 ±25mA (Transient or steady state)	16.0	17.0	18.0	V
V_{FEL}	FE Supply (Idle)		0	0.4	0.8	V
I_{FEH}	FE Supply Current Limit	V_{FEH} = +17 ±1V	275	300	325	mA
V_{CCP}[1]	V_{CC} Supply (Program "AND")	I_{CCP} = 550 mA, min. (Transient or steady state)	4.75	5.0	5.25	V
I_{CCP}	I_{CC} Limit (Program "AND")	V_{CCP} = +5.0 ±.25V	550		1,000	mA
V_{OPF}	Forced Output (Program)		9.5	10	10.5	V
I_{OPF}	Output Current (Program)				10	mA
T_R	Output Pulse Rise Time		10		50	μs
t_P	$\overline{CE}$ Programming Pulse Width		1		1.5	ms
t_D	Pulse Sequence Delay		10			μs
T_{PR}	Programming Time			2		ms
$\frac{T_{PR}}{T_{PR}+T_{PS}}$	Programming Duty Cycle				50	%
F_L	Fusing Attempts per Link				3	cycle
V_S[3]	Verify Threshold		0.9	1.0	1.1	V

NOTES:
1. Bypass V_{CC} to GND with a 0.01 μf capacitor to reduce voltage spikes.
2. Care should be taken to ensure that the voltage is maintained during the entire fusing cycle. The recommended supply is a constant current source clamped at the specified voltage limit.
3. V_S is the sensing threshold of the FPLA output voltage for a programmed link. It normally constitutes the reference voltage applied to a comparator circuit to verify a successful fusing attempt.
4. These are specifications which a Programming *System* must satisfy in order to be qualified by Signetics.

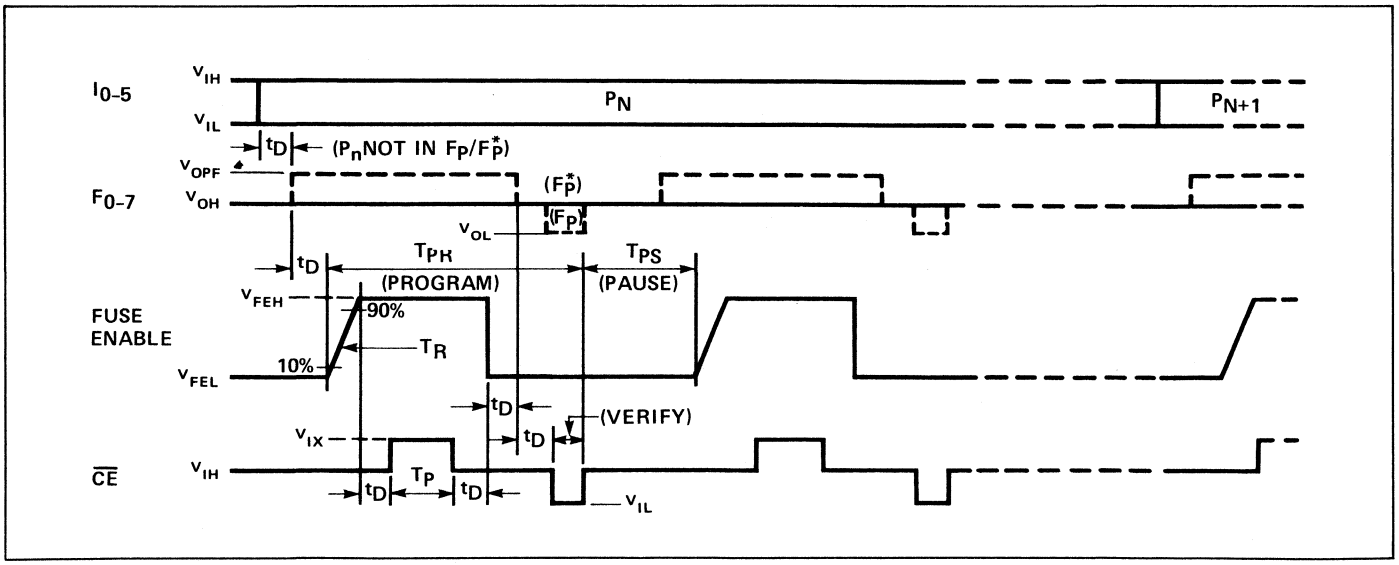
OUTPUT POLARITY PROGRAM-VERIFY SEQUENCE (Typical)



"AND" MATRIX PROGRAM-VERIFY SEQUENCE (Typical)



"OR" MATRIX PROGRAM-VERIFY SEQUENCE (Typical)



VIRGIN DEVICE

The 82S100/101 are shipped in an unprogrammed state, characterized by:

A. All internal Ni-Cr links are intact.
B. Each Product term (P-term) contains both true and complement values of every input variable I_m (P-terms always logically "FALSE").
C. The "OR" Matrix contains all 48-P-terms.
D. The polarity of each output is set to active HIGH (F_P function).
E. All outputs are at a LOW logic level.

RECOMMENDED PROGRAMMING PROCEDURE

To program each of 8 Boolean logic functions of 16 true or complement variables, including up to 48 P-terms, follow the Program/Verify procedures for the "AND" Matrix, "OR" Matrix, and Output Polarity outlined below. To maximize recovery from programming errors, leave all links in unused device areas intact.

SET-UP

Terminate all device outputs with a 10KΩ resistor to +5V.

OUTPUT POLARITY

PROGRAM ACTIVE LOW ($\overline{Fp}$ Function)*

Program output polarity before programming "AND" Matrix and "OR" Matrix. Program one output at the time. (S) links of unused outputs are not required to be fused.

1. Set GND (pin 14), and FE (pin 1) to 0V.
2. Set V_{CC} (pin 28) to V_{CCL}.
3. Set $\overline{CE}$ (pin 19), and I_0 through I_{15} to V_{IH}.
4. Apply V_{OPH} to the appropriate output, and remove after a period t_p.
5. Repeat step 4 to program other outputs.

VERIFY OUTPUT POLARITY

1. Set GND (pin 14) to 0V, and V_{CC} (pin 28) to V_{CCS}.
2. Enable the chip by setting $\overline{CE}$ (pin 19) to V_{IL}.
3. Address a non-existent P-term by applying V_{IH} to all inputs I_0 through I_{15}.
4. Verify output polarity by sensing the logic state of outputs F_0 through F_7. All outputs at a HIGH logic level are programmed active LOW ($\overline{Fp}$ function), while all outputs at a LOW logic level are programmed active HIGH (Fp function).
5. Return V_{CC} to V_{CCP} or V_{CCL}.

"AND" MATRIX

PROGRAM INPUT VARIABLE

Program one input at the time and one P-term at the time. All input variable links of unused P-terms are not required to be fused. However, unused input variables must be programmed as Don't Care for all programmed P-terms.

1. Set GND (pin 14) to 0V, and V_{CC} (pin 28) to V_{CCP}.
2. Disable all device outputs by setting $\overline{CE}$ (pin 19) to V_{IH}.
3. Disable all input variables by applying V_{IX} to inputs I_0 through I_{15}.
4. Address the P-term to be programmed (No. 0 through 47) by forcing the corresponding binary code on outputs F_0 through F_5 with F_0 as LSB. Use standard TTL logic levels V_{OHF} and V_{OLF}.

5a. If the P-term contains neither I_0 nor $\overline{I}_0$ (input is a Don't Care), fuse both I_0 and $\overline{I}_0$ links by executing both steps 5b and 5c, before continuing with step 7.

5b. If the P-term contains I_0, set to fuse the $\overline{I}_0$ link by lowering the input voltage to I_0 from V_{IX} to V_{IH}. Execute step 6.

5c. If the P-term contains $\overline{I}_0$, set to fuse the I_0 link by lowering the input voltage to I_0 from V_{IX} to V_{IL}. Execute step 6.

6a. After t_D delay, raise FE (pin 1) from V_{FEL} to V_{FEH}.

6b. After t_D delay, pulse the $\overline{CE}$ input from V_{IH} to V_{IX} for a period t_p.

6c. After t_D delay, return FE input to V_{FEL}.

7. Disable programmed input by returning I_0 to V_{IX}.
8. Repeat steps 5 through 7 for all other input variables.
9. Repeat steps 4 through 8 for all other P-terms.
10. Remove V_{IX} from all input variables.

VERIFY INPUT VARIABLE

1. Set GND (pin 14) to 0V, and V_{CC} (pin 28) to V_{CCP}.
2. Enable F_7 output by setting $\overline{CE}$ to V_{IX}.
3. Disable all input variables by applying V_{IX} to inputs I_0 through I_{15}.
4. Address the P-term to be verified (No. 0 through 47) by forcing the corresponding binary code on outputs F_0 through F_5.
5. Interrogate input variable I_0 as follows:
   A. Lower the input voltage to I_0 from V_{IX} to V_{IH}, and sense the logic state of output F_7.
   B. Lower the input voltage to I_0 from V_{IH} to V_{IL}, and sense the logic state of output F_7.

The state of I_0 contained in the P-term is determined in accordance with the following truth table:

I_0	F_7	Input Variable State Contained in P-Term
0 1	1 0	$\overline{I}_0$
0 1	0 1	I_0
0 1	1 1	Dont Care
0 1	0 0	(I_0), $(\overline{I}_0)$

note that two tests are required to uniquely deter-

mine the state of the input variable contained in the P-term.

6. Disable verified input by returning I_0 to V_{IX}.
7. Repeat steps 5 and 6 for all other input variables.
8. Repeat steps 4 through 7 for all other P-terms.
9. Remove V_{IX} from all input variables.

"OR" MATRIX

PROGRAM PRODUCT TERM

Program one output at the time for one P-term at the time. All P_n links in the "OR" Matrix corresponding to unused outputs and unused P-terms are not required to be fused.

1. Set GND (pin 14) to 0V, and V_{CC} (pin 28) to V_{CCS}.
2. Disable the chip by setting $\overline{CE}$ (pin 19) to V_{IH}.
3. Set inputs I_6 through I_{15} to V_{IH} or V_{IL}.
4. Address the P-term to be programmed (No. 0 47) by applying the corresponding binary code to input variables I_0 through I_5, with I_0 as LSB.

5a. If the P-term is contained in output function F_0 ($F_0 = 1$ or $F_0^* = 0$), go to step 6, (fusing cycle not required).

5b. If the P-term is **not** contained in output function F_0 ($F_0 = 0$ or $F_0^* = 1$), set to fuse the P_n link by forcing output F_0 to V_{OPF}.

6a. After t_D delay, raise FE (pin 1) from V_{FEL} to V_{FEH}.

6b. After t_D delay, pulse the $\overline{CE}$ input from V_{IH} to V_{IX} for a period t_p.

6c. After t_D delay, return FE input to V_{FEL}.

6d. After t_D delay, remove V_{OPF} from output F_0.

7. Repeat steps 5 and 6 for all other output functions.
8. Repeat steps 4 through 7 for all other P-terms.
9. Remove V_{CCS} from V_{CC}.

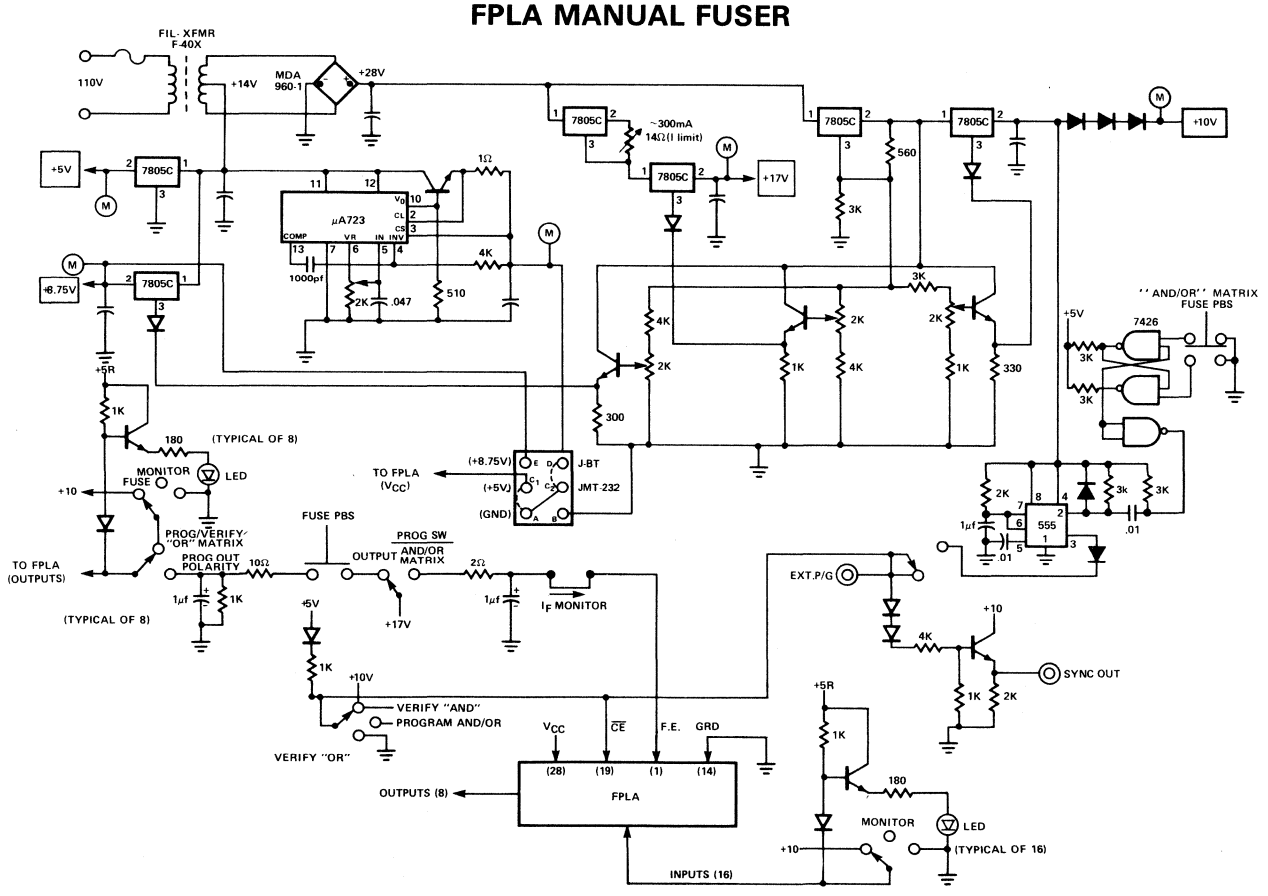
VERIFY PRODUCT TERM

1. Set GND (pin 14) to 0V, and V_{CC} (pin 28) to V_{CCS}.
2. Enable the chip by setting $\overline{CE}$ (pin 19) to V_{IL}.
3. Set inputs I_0 through I_{15} to V_{IH} or V_{IL}.
4. Address the P-term to be verified (No. 0 through 47) by applying the corresponding binary code to input variables I_0 through I_5.
5. To determine the status of the P_n link in the "OR" Matrix for each output function F_p or F_p^*, sense the state of outputs F_0 through F_7. The status of the link is given by the following truth table:

OUTPUT		P-term Link
Active HIGH (F_p)	Active LOW (F_p^*)	
0	1	FUSED
1	0	PRESENT

6. Repeat steps 4 and 5 for all other P-terms.
7. Remove V_{CCS} from V_{CC}.

FPLA MANUAL FUSER



16 X 48 X 8 FPLA PROGRAM TABLE

THIS PORTION TO BE COMPLETED BY SIGNETICS

CF (XXXX) _____

CUSTOMER SYMBOLIZED PART # _____

DATE RECEIVED _____

COMMENTS _____

CUSTOMER NAME _____

PURCHASE ORDER # _____

SIGNETICS DEVICE # _____

TOTAL NUMBER OF PARTS _____

PROGRAM TABLE # _____ *REV* _____ *DATE* _____

PROGRAM TABLE ENTRIES

INPUT VARIABLE			OUTPUT FUNCTION		OUTPUT ACTIVE LEVEL	
I_m	$\overline{I_m}$	DON'T CARE	PROD. TERM PRESENT IN F_p	PROD. TERM *NOT* PRESENT IN F_p	ACTIVE HIGH	ACTIVE LOW
H	L	– (dash)	A	● (period)	H	L
NOTE: Enter (–) for *unused* inputs of *used* P-terms.			NOTES: 1) Entries independent of output polarity. 2) Enter (A) for *unused* outputs of *used* P-terms.		NOTES: 1) Polarity programmed once only. 2) Enter (H) for all *unused* outputs.	

	PRODUCT TERM*															
	INPUT VARIABLE															
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																
1																
2																
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
16																
17																
18																
19																
20																
21																
22																
23																
24																
25																
26																
27																
28																
29																
30																
31																
32																
33																
34																
35																
36																
37																
38																
39																
40																
41																
42																
43																
44																
45																
46																
47																

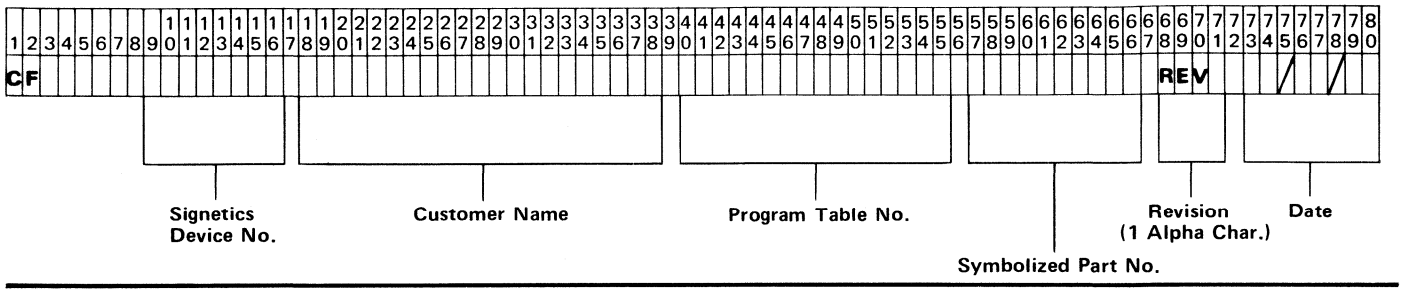
	ACTIVE LEVEL							
	OUTPUT FUNCTION*							
	7	6	5	4	3	2	1	0
0								
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								
30								
31								
32								
33								
34								
35								
36								
37								
38								
39								
40								
41								
42								
43								
44								
45								
46								
47								

*Input and Output fields of *unused* P-terms can be left blank.

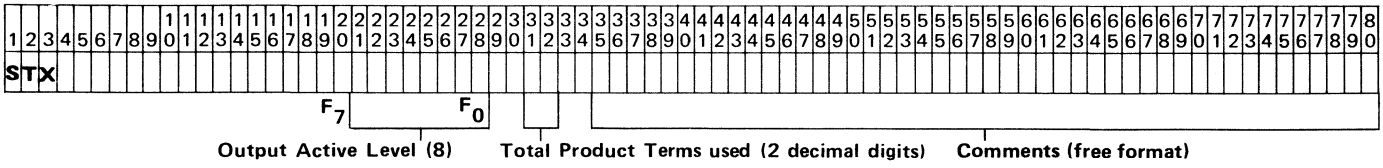
PUNCHED CARD CODING FORMAT

The FPLA Program Table can be supplied directly to Signetics in Punched Card form, using standard 80-column IBM cards. For each FPLA Program Table, the customer should prepare an input card deck in accordance with the following format. Product Term cards 3 through 50 can be in any order. Not all 48 Product Terms need to be present. Unused Product Terms require no entry cards, and will be skipped during the actual programming sequence:

CARD NO. 1-Free format within designated fields.



CARD NO. 2-

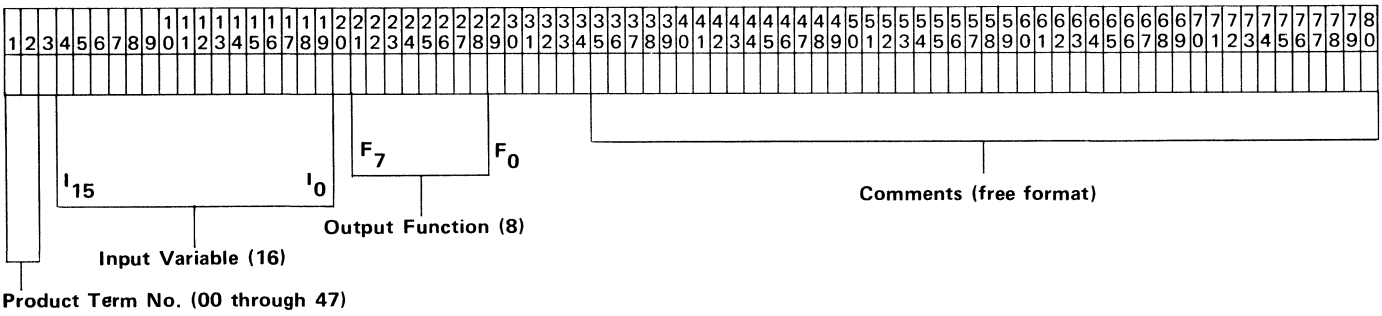


Output Active Level entries are determined in accordance with the following table:

OUTPUT ACTIVE LEVEL	
ACTIVE HIGH	ACTIVE LOW
H	L

NOTES:
1. *Polarity programmed once only.*
2. *Enter (H) for all* unused *outputs.*

CARD No. 3 through No. 50



Input Variable and Output Function entries are determined in accordance with the following table:

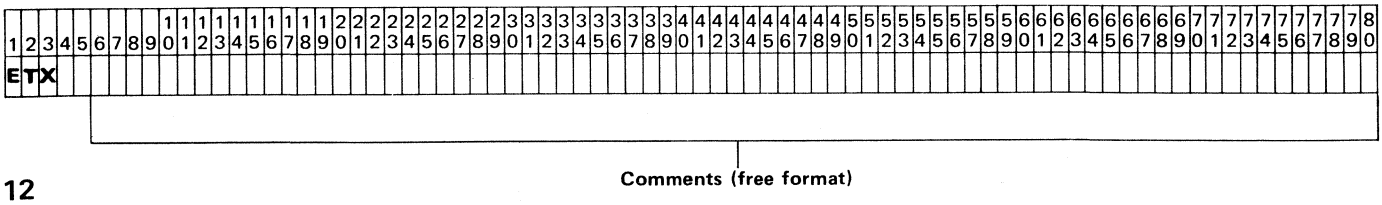
INPUT VARIABLE		
I_m	$\overline{I_m}$	DON'T CARE
H	L	– (dash)

NOTE:
Enter (–) for unused *inputs of* used *P-terms.*

OUTPUT FUNCTION	
PROD. TERM PRESENT IN F_P	PROD. TERM *NOT* PRESENT IN F_P
A	• (period)

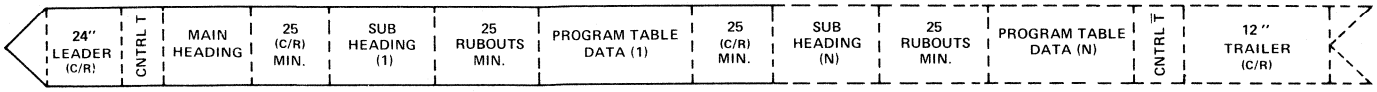
NOTES:
1. *Entries independent of output polarity.*
2. *Enter (A) for* unused *inputs of* used *P-terms.*

CARD NO. 51



TWX TAPE CODING FORMAT

The FPLA Program Table can be sent to Signetics in ASCII code format via TWX, or air mail using any type of 8-level tape (paper, mylar, fanfold, etc.) A number of Program Tables can be sequentially assembled on a continuous tape as follows, however limit tape length to a roll of 1.75 inch inside diameter, and 4.25 inch outside diameter:



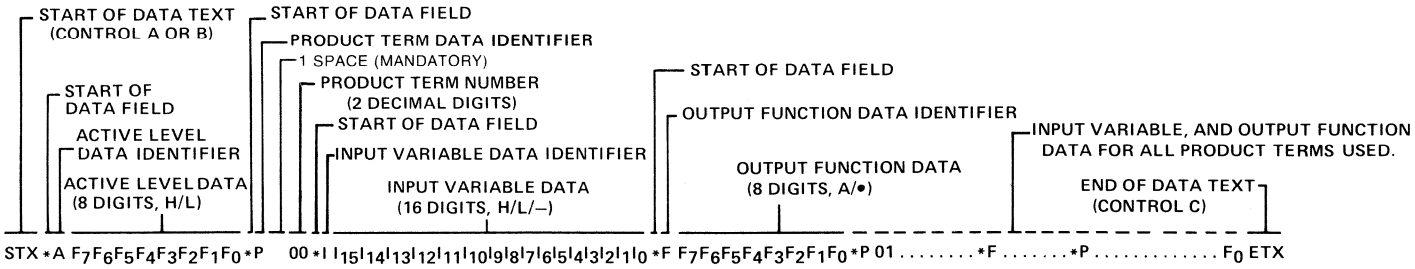
A. The MAIN HEADING at the beginning of tape includes the following information, with each entry preceded by a ($) character, whether used or not:

1. **Customer Name** _____
2. **Customer TWX No.** _____
3. **Date** _____
4. **Purchase Order No.** _____
5. **Number of Program Tables** _____
6. **Total Number of Parts** _____

B. Each SUB HEADING should contain specific information pertinent to each Program Table as follows, with each entry preceded by a ($) character, whether used or not:

1. **Signetics Device No.** _____
2. **Program Table No.** _____
3. **Revision** _____
4. **Date** _____
5. **Customer Symbolized Part No.** _____
6. **Number of Parts** _____

C. Program Table data blocks are initiated with an STX character, and terminated with an ETX character. The body of the data consist of Output Active Level, Product Term, and Output Function information separated by appropriate identifiers in accordance with the following format:



Entries for the 3 Data Fields are determined in accordance with the following Table:

INPUT VARIABLE			OUTPUT FUNCTION		OUTPUT ACTIVE LEVEL	
I_m	$\overline{I_m}$	DON'T CARE	PROD. TERM PRESENT IN F_p	PROD. TERM *NOT* PRESENT IN F_p	ACTIVE HIGH	ACTIVE LOW
H	L	– (dash)	A	• (period)	H	L
NOTE: Enter (–) for *unused* inputs of *used* P-terms.			NOTES: 1) Entries independent of output polarity. 2) Enter (A) for *unused* outputs of *used* P-terms.		NOTES: 1) Polarity programmed once only. 2) Enter (H) for all *unused* outputs.	

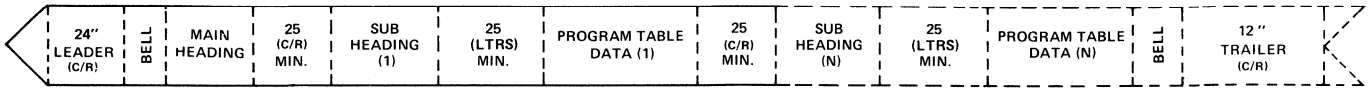
Although the Product Term data are shown entered in sequence, this is not necessary. It is possible to input only one Product Term, if desired. Unused Product Terms require no entry. ETX signalling end of Program Table may occur with less than the maximum number of Product Terms entered.

NOTES:

1) Correction of errors in P-Term *number* is not allowed. If this occurs, delete the P-Term as in (3) below, if in time. Otherwise, fill (*I) and (*F) fields, then delete.
2) Corrections to any other entry can be made by backspace and rubout. However, limit consecutive rubouts to less than 25.
3) Any P-Term can be deleted entirely by inserting the character (E) *immediately following* the P-Term *number* to be deleted, i.e., *P 25E deletes P-Term 25.
4) Carriage returns, line feeds, spaces, rubouts, etc. may be interspersed between data groups to facilitate an orderly Teletype printout.
5) Comments are allowed between data fields, provided that an asterisk (*) is *not* used in any Heading or Comment entry.

TELEX TAPE CODING FORMAT

The FPLA Program Table can be sent to Signetics in BAUDOT code format via TELEX, or air mail using any type of 5-level tape (paper, mylar, fanfold, etc.) A number of Program Tables can be sequentially assembled on a continuous tape as follows, however limit tape lenth to a roll of 1.75 inch inside diameter, and 4.25 inch outside diameter:



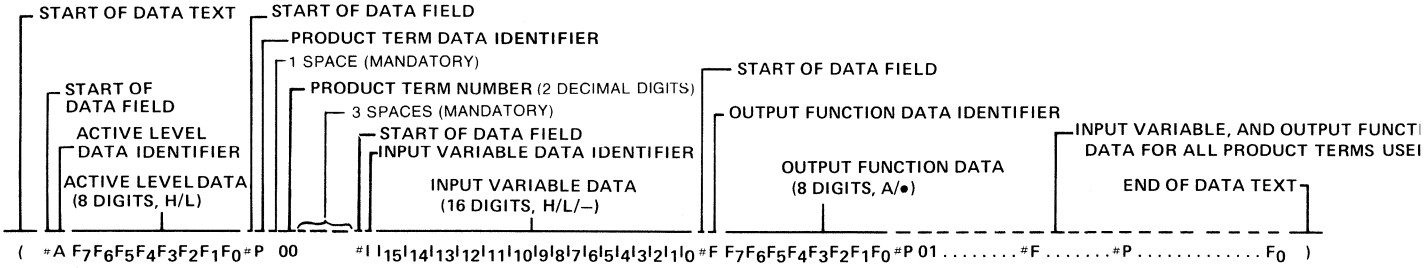
A. The MAIN HEADING at the beginning of tape includes the following information, with each entry preceded by a ($) character, whether used or not:

1. **Customer Name** _____
2. **Customer TELEX No.** _____
3. **Date** _____
4. **Purchase Order No.** _____
5. **Number of Program Tables** _____
6. **Total Number of Parts** _____

B. Each SUB HEADING should contain specific information pertinent to each Program Table as follows, with each entry preceded by a ($) character, whether used or not:

1. **Signetics Device No.** _____
2. **Program Table No.** _____
3. **Revision** _____
4. **Date** _____
5. **Customer Symbolized Part No.** _____
6. **Number of Parts** _____

C. Program Table data blocks are initiated with an "Open" parenthesis, and terminated with a "Closed" parenthesis. The body of the data consists of Output Active Level, Product Term, and Output Function information separated by appropriate identifiers in accordance with the following format:



Entries for the 3 Data Fields are determined in accordance with the following Table:

INPUT VARIABLE			OUTPUT FUNCTION		OUTPUT ACTIVE LEVEL	
I_m	$\overline{I_m}$	DON'T CARE	PROD. TERM PRESENT IN F_p	PROD. TERM *NOT* PRESENT IN F_p	ACTIVE HIGH	ACTIVE LOW
H	L	– (dash)	A	• (period)	H	L
NOTE: Enter (-) for *unused* inputs of *used* P-terms.			NOTES: 1) Entries independent of output polarity. 2) Enter (A) for *unused* outputs of *used* P-terms.		NOTES: 1) Polarity programmed once only. 2) Enter (H) for all *unused* outputs.	

Data fields for each Product Term must be entered in ascending order, beginning with Product Term (00) and ending with Product Term (47).

Active Level data must be entered prior to Product Term (00).

An entry must be made for all 48 Product Terms. For all *unused* Product Terms follow the typical entry procedure outlined below:

#P 05XX to designate P-Term 05 "UNUSED", and to be left unprogrammed.
#P 05XXX to designate P-Terms 05 through 47 "UNUSED", and to be left unprogrammed.

NOTES

1) Correction of errors in P-Term *number* is not allowed. If this occurs, delete the P-Term as in (3) below, if in time. Otherwise, fill (#I) and (#F) fields, then delete.
2) Corrections to any other entry can be made by backspace and rubout (LTRS). However, limit consecutive rubouts to less than 25.
3) An erroneous P-Term can be corrected only by *immediate* 'overlay' of CORRECT data over a deleted P-Term. A P-Term is deleted by inserting a single character (X) immediately following the P-Term *number*. Thus, the sequence: #P 10 (data error) (rest of I/O fields) #P 10X #P 10__#I (correct data), allows P-Term 10 to be corrected.
4) Carriage returns, line feeds, spaces, rubouts, etc. may be interspersed between data groups to facilitate an orderly Teletype printout.
5) Comments are allowed between data fields, provided that the character (#) is *not* used in any Heading or Comment entry.

PROGRAM TABLE OF SAMPLE DEVICE

THIS PORTION TO BE COMPLETED BY SIGNETICS

CF (XXXX) 1001

CUSTOMER SYMBOLIZED PART # WHEN YOU LIKE

DATE RECEIVED FOR ALL PURPOSE LOGIC APPLICATIONS

COMMENTS OR, AND, EX-OR, MUX, ADD

CUSTOMER NAME ADVANCED LOGIC DESIGNS

PURCHASE ORDER # NONE (FREE SAMPLE)

SIGNETICS DEVICE # 82S100/101

TOTAL NUMBER OF PARTS 1

PROGRAM TABLE # REV DATE 7/22/75

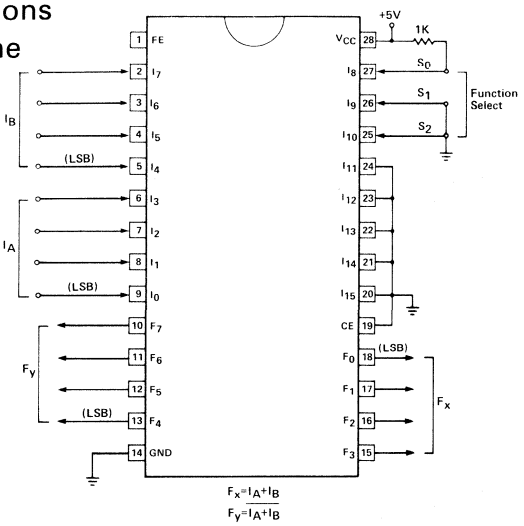
PROGRAM TABLE ENTRIES						
INPUT VARIABLE			OUTPUT FUNCTION		OUTPUT ACTIVE LEVEL	
I_m	$\overline{I}_m$	DON'T CARE	PROD. TERM PRESENT IN F_p	PROD. TERM *NOT* PRESENT IN F_p	ACTIVE HIGH	ACTIVE LOW
H	L	– (dash)	A	● (period)	H	L
NOTE: Enter (–) for *unused* inputs of *used* P-terms.			NOTES: 1) Entries independent of output polarity. 2) Enter (A) for *unused* outputs of *used* P-terms.		NOTES: 1) Polarity programmed once only. 2) Enter (H) for all *unused* outputs.	

PRODUCT TERM*																	ACTIVE LEVEL							
	INPUT VARIABLE																L	L	L	L	H	H	H	H
NO.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	OUTPUT FUNCTION*							
																	7	6	5	4	3	2	1	0
0	—	—	—	—	—	L	—	H	—	—	—	—	—	—	—	H	●	●	●	A	●	●	●	A
1	—	—	—	—	—	L	L	H	—	—	—	H	—	—	—	—	●	●	●	A	●	●	●	A
2	—	—	—	—	—	L	—	H	—	—	—	—	—	—	H	—	●	●	A	●	●	●	A	●
3	—	—	—	—	—	L	L	H	—	—	H	—	—	—	—	—	●	●	A	●	●	●	A	●
4	—	—	—	—	—	L	—	H	—	—	—	—	—	H	—	—	●	A	●	●	●	A	●	●
5	—	—	—	—	—	L	L	H	—	H	—	—	—	—	—	—	●	A	●	●	●	A	●	●
6	—	—	—	—	—	L	—	H	—	—	—	—	H	—	—	—	A	●	●	●	A	●	●	●
7	—	—	—	—	—	L	L	H	H	—	—	—	—	—	—	—	A	●	●	●	A	●	●	●
8	—	—	—	—	—	L	L	L	—	—	—	H	—	—	—	H	●	●	●	A	●	●	●	A
9	—	—	—	—	—	L	L	L	—	—	H	—	—	—	H	—	●	●	A	●	●	●	A	●
10	—	—	—	—	—	L	L	L	—	H	—	—	—	H	—	—	●	A	●	●	●	A	●	●
11	—	—	—	—	—	L	L	L	H	—	—	—	H	—	—	—	A	●	●	●	A	●	●	●
12	—	—	—	—	—	H	H	H	—	—	—	H	—	—	—	—	●	●	●	A	●	●	●	A
13	—	—	—	—	—	H	H	H	—	—	H	—	—	—	—	—	●	●	A	●	●	●	A	●
14	—	—	—	—	—	H	H	H	—	H	—	—	—	—	—	—	●	A	●	●	●	A	●	●
15	—	—	—	—	—	H	H	H	H	—	—	—	—	—	—	—	A	●	●	●	A	●	●	●
16	—	—	—	—	L	H	L	L	—	—	—	L	—	—	—	H	●	●	●	●	●	●	●	A
17	—	—	—	—	L	H	L	L	—	—	—	H	—	—	—	L	●	●	●	●	●	●	●	A
18	—	—	—	—	H	H	L	L	—	—	—	L	—	—	—	L	●	●	●	●	●	●	●	A
19	—	—	—	—	H	H	L	L	—	—	—	H	—	—	—	H	●	●	●	●	●	●	●	A
20	—	—	—	H	—	H	L	L	—	—	L	—	—	—	H	—	●	●	●	●	●	●	A	●
21	—	—	—	H	—	H	L	L	—	—	H	—	—	—	L	—	●	●	●	●	●	●	A	●
22	—	—	—	L	—	H	L	L	—	—	L	—	—	—	L	—	●	●	●	●	●	●	A	●
23	—	—	—	L	—	H	L	L	—	—	H	—	—	—	H	—	●	●	●	●	●	●	A	●
24	—	—	H	—	—	H	L	L	—	L	—	—	—	H	—	—	●	●	●	●	●	A	●	●
25	—	—	H	—	—	H	L	L	—	H	—	—	—	L	—	—	●	●	●	●	●	A	●	●
26	—	—	L	—	—	H	L	L	—	L	—	—	—	L	—	—	●	●	●	●	●	A	●	●
27	—	—	L	—	—	H	L	L	—	H	—	—	—	H	—	—	●	●	●	●	●	A	●	●
28	—	H	—	—	—	H	L	L	L	—	—	—	H	—	—	—	●	●	●	●	A	●	●	●
29	—	H	—	—	—	H	L	L	H	—	—	—	L	—	—	—	●	●	●	●	A	●	●	●
30	—	L	—	—	—	H	L	L	L	—	—	—	L	—	—	—	●	●	●	●	A	●	●	●
31	—	L	—	—	—	H	L	L	H	—	—	—	H	—	—	—	●	●	●	●	A	●	●	●
32	—	L	—	—	—	H	L	L	—	—	—	—	H	—	—	—	●	●	●	A	●	●	●	●
33	—	L	—	—	—	H	L	L	H	—	—	—	—	—	—	—	●	●	●	A	●	●	●	●
34	—	—	—	—	H	H	L	L	—	—	—	—	—	—	—	H	●	●	A	●	●	●	●	●
35	—	—	—	—	H	H	L	L	—	—	—	H	—	—	—	—	●	●	A	●	●	●	●	●
36	—	—	—	L	—	H	L	L	—	—	—	—	—	—	H	—	●	A	●	●	●	●	●	●
37	—	—	—	L	—	H	L	L	—	—	H	—	—	—	—	—	●	A	●	●	●	●	●	●
38	—	—	L	—	—	H	L	L	—	—	—	—	—	H	—	—	A	●	●	●	●	●	●	●
39	—	—	L	—	—	H	L	L	—	H	—	—	—	—	—	—	A	●	●	●	●	●	●	●
40	—	—	—	—	—	H	L	L	H	—	—	—	H	—	—	—	●	●	●	A	●	●	●	●
41	—	—	—	—	—	H	L	L	—	—	—	H	—	—	—	H	●	●	A	●	●	●	●	●
42	—	—	—	—	—	H	L	L	—	—	H	—	—	—	H	—	●	A	●	●	●	●	●	●
43	—	—	—	—	—	H	L	L	—	H	—	—	—	H	—	—	A	●	●	●	●	●	●	●
44																								
45																								
46																								
47																								

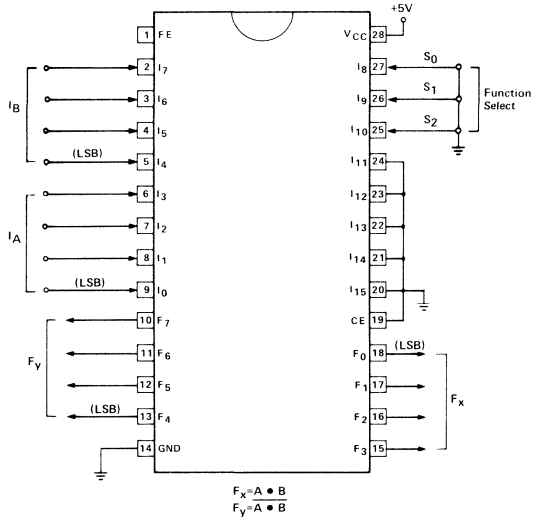
*Input and Output fields of *unused* P-terms can be left blank.

CONNECTIONS FOR SAMPLE DEVICE

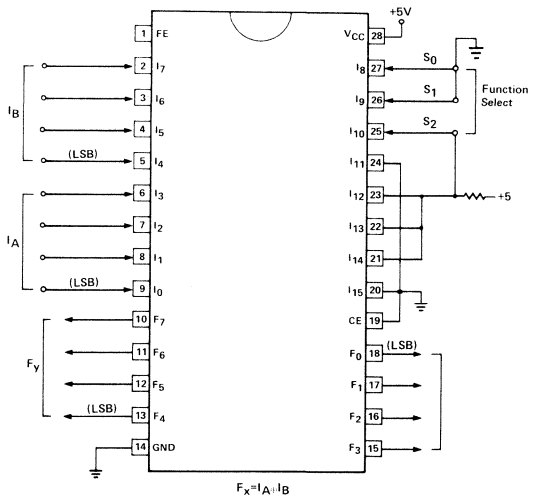
To observe the five logic functions of the sample FPLA, connect the device as follows:



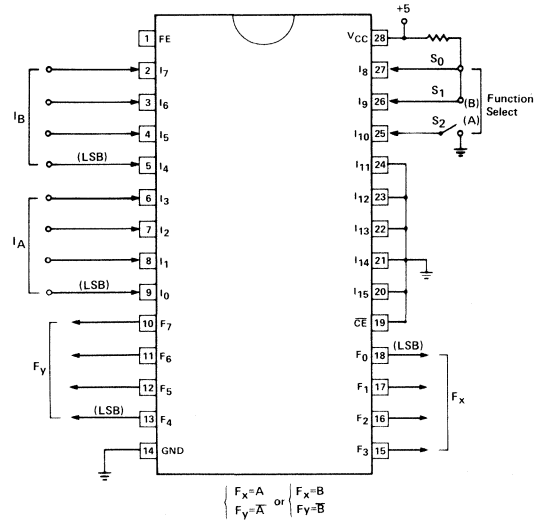
a. "OR" FUNCTION



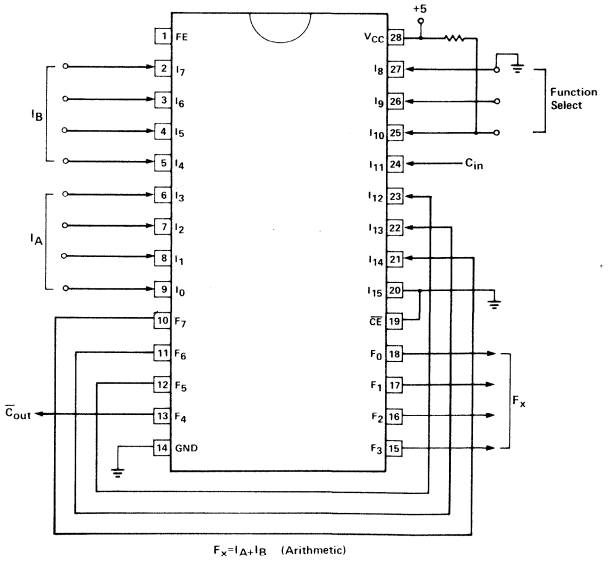
b. "AND" FUNCTION



c. "EX-OR" FUNCTION



d. "MULTIPLEX" FUNCTION



e. "ADD" FUNCTION (with Serial Carry)

NI-CR TECHNOLOGY MATURES

Nichrome was the first material to give rise to stable, low current fuses with excellent fusing characteristics, easily reproducible. However, as with all new developments, Nichrome technology had to undergo a learning curve, with each advance signaling the advent of more complex and higher performance devices, without a compromise in reliability. It soon became apparent that each incremental step in complexity implied a fuller understanding of the fusing phenomenon. Accordingly, fusible link technology has been intensively investigated by Signetics over the last six years (see Signetics' Prom Reliability page 39), giving rise to the broadest line of PROMs in the industry, and presently, the addition of a family of Field Programmable Logic Arrays (FPLAs), designed for both commercial and temperature ranges.

USER ORIENTED DEVICE CONFIGURATION

Signetics' family of Bipolar Field Programmable Logic Arrays include both Tri-state (82S100), and Open Collector devices (82S101), featuring the following characteristics:

- FIELD PROGRAMMABLE (Ni-Cr LINK)
- 16 INPUT VARIABLES
- 8 OUTPUT FUNCTIONS
- 48 PRODUCT TERMS
- 50ns MAX. ACCESS TIME (0–75°C)
- 600mW POWER DISSIPATION (TYPICAL)
- TTL COMPATIBLE
- 28 PIN PACKAGE
- $\overline{CE}$ INPUT FOR EXPANSION OR INHIBIT
- OUTPUTS INDIVIDUALLY PROGRAMMABLE ACTIVE "HIGH" OR "LOW"
- SINGLE +5V POWER SUPPLY

The above features and organization combine into an easy to use, high performance device, affording distinct user benefits:

A. 16 Input Variables

The 16 by 8 I/O configuration permits direct byte manipulations required by intelligent terminals, peripherals, microprocessor based emulators, minicomputers, and all the way up to the larger mainframes. Also, in address mapping applications, it provides the capability to scan an address field 65,536 words deep.

B. Chip Enable Input

The Chip Enable input is a major improvement over current devices:

- Eases expansion of input variables and/or product terms.
- Permits application of Tristate device in bus organized systems.
- Provides logic inhibit or preconditional decoding functions.
- Provides a unique "default" logic state for all outputs, regardless of programmed output polarity.

C. Fastest Access Time

50ns maximum over the commercial temperature range renders the replacement of random logic feasible.

D. Fully Buffered Devices

All product terms can be utilized as many times as required, without affecting device speed and power dissipation.

E. 48 Product Terms (P-Terms)

Allow the user to store in the FPLA 48 distinct words of 8 bits each. These 48 words can be addressed by a **minimum** of 48 input address combinations, chosen by the user among a total available pool of 2^{16} (65,536).

F. Polarity of All Outputs Individually Programmable Active-High or Active-Low

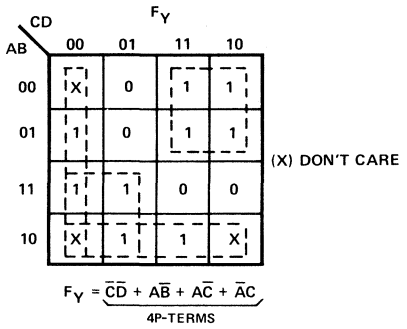
This feature is particularly useful in achieving further Product Term minimization in cases where the complement of an output function can be implemented with fewer Product Terms.

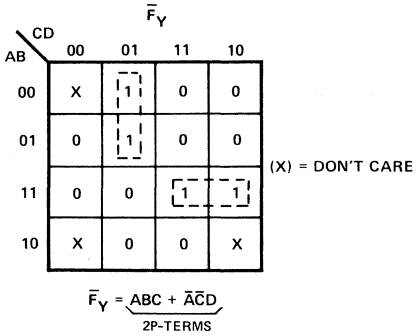
Example:

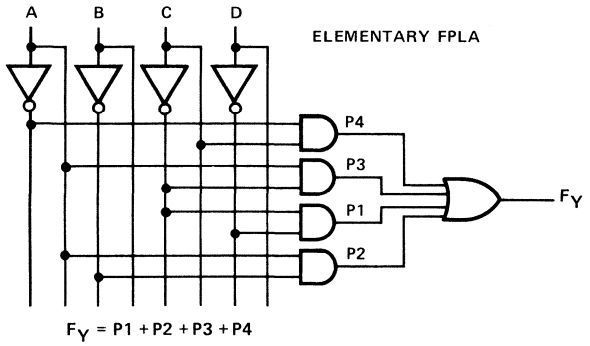
As shown on page 18, a 50% reduction in P-terms is obtained when the output of the logical structure of $\overline{F\gamma}$ is inverted by means of a gate **external** to the elementary FPLA. The desired function $F\gamma$ is realized with penalties in hardware, and circuit delays (however small). These are eliminated when using an FPLA with output polarity programmed active-LOW to realize the function "0's", rather than "1's".

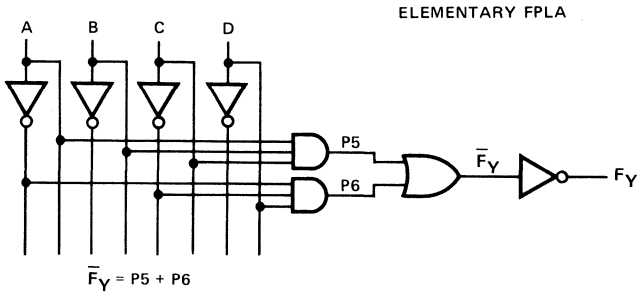
WHAT IS AN FPLA?

The structure and use of FPLAs can be more easily understood by comparing them to PROMs. In the industry we refer to PROMs as 1K, 4K, etc. These usually imply standard organizations such as 256X4, 512X8, respectively. The larger in each pair of numbers refers to the number of **words** in a PROM, and the second represents the number of **bits** in **each** word. The product of both numbers (approximately 1K, 4K) gives the total









number of storage bits contained in the PROM.

This aspect of PROMs carries over to FPLAs, such that Signetics' FPLAs can be described as 48X8, for a total **working** storage density of 384 bits. Thus, the FPLA is a relatively small PROM, but a much more useful one, due to a fundamental difference in input structure.

In a PROM (Fig. 1), all internal words are reached by a **fixed** decoder internal to the device. The size of this decoder, as well as the storage matrix, doubles for each additional address input. In a 256X8 PROM, the internal decoder selects 1 of 256 words by examining 8 address inputs. For a 512X8 PROM, 1 of 512 words are selected by a decoder twice as large by examining 9 address inputs.

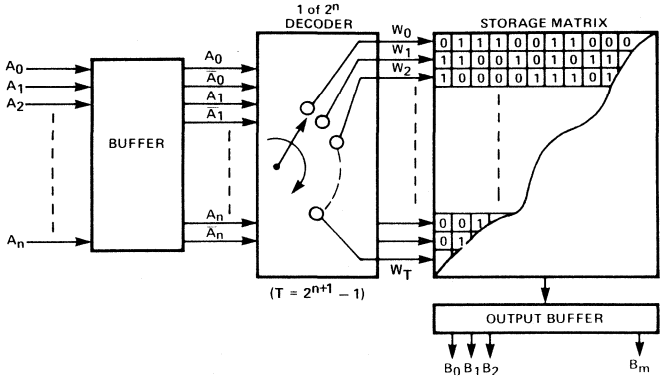


FIG. 1: Simplified PROM organization.

	ADDRESS			OUTPUTS				
M_n	A_2	A_1	A_0	O_1	O_2	O_3	O_4	
0	0	0	0	1	1	1	1	
1	0	0	1	0	0	1	1	
2	0	1	0	0	0	0	0	Inactive minterms
3	0	1	1	0	1	1	1	
4	1	0	0	0	0	0	0	
5	1	0	1	1	0	1	0	
6	1	1	0	1	1	0	0	
7	1	1	1	0	0	0	0	

FIG. 2: Typical truth-table stored in an 8X4 PROM.

The presence of a fixed decoder renders PROM addressing exhaustive. This can never be avoided, and forces the utilization of PROMs in discrete chunks. This constraint is at the root of the inefficiency of PROMs in the type of application shown in Fig. 2. Notice that if we define logic "1" as the active-True state of all output functions, it is not possible to compress the truth table by eliminating inactive minterms 2, 4, and 7. Moreover, with regard to minterms 0 and 1, it is necessary to allocate two distinct storage locations to activate output function 0_3 with a single change in input variable A_0. In this case, A_0 represents a logical Don't Care (X) which cannot be directly programmed in a PROM. Instead, separate minterms $\overline{A_2}\overline{A_1}A_0$ and $\overline{A_2}\overline{A_1}\overline{A_0}$ must be programmed.

With FPLA, both constraints are removed.

As shown in Fig. 3, the FPLA does away with a fixed decoder in favor of a **programmable Address Matrix**, which offers, in place of forced exhaustive addressing, the flexibility to choose by "linear-select" any finite subset from a large number of input states. This is possible because each column of the Address Matrix functions essentially as a logic comparator programmed to recognize the simultaneous presence of (n) inputs, each either True, False, or both (Don't Care).

As a result, storage for unused minterms is no longer required. The necessary logic output for

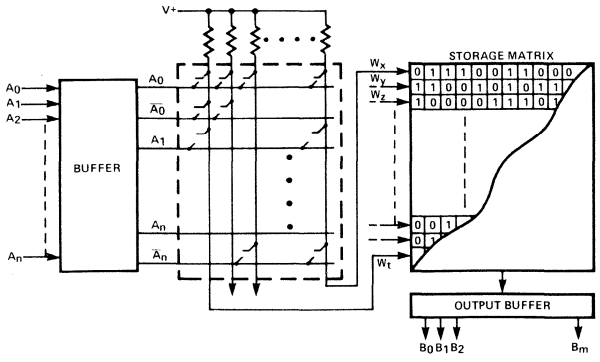


FIG. 3: Typical FPLA organization. The input buffer drives a programmable address matrix, in which anyone of 2^{n+1} input combinations can be programmed to select a stored word $W_{x, y \ldots t}$.

the inactive minterms occurs by "default". And, Don't Care states of input variables can be directly programmed in the FPLA. This allows to program the FPLA with either Minterms or the more general Product Terms (P-terms) of the input variables (addresses) to minimize logic "waste".

When any programmed logic combination is present at the FPLA inputs, the corresponding Address Matrix column (P-term) will be pulled HIGH (logically active), forcing all (B) outputs to their True logic state programmed in the Storage Matrix. Conversely, for all **unprogrammed** logic combinations present at the FPLA inputs, all columns will remain LOW (logically inactive) forcing all (B) outputs to their False logic state by **default** (the complementary logic state of their programmed Active Level polarity).

Because it is programmable, the FPLA Address Matrix is not bound in size by the number of inputs it examines. Signetics' FPLA has 16 inputs to the matrix. If it were a PROM, this address matrix would have to be large enough to decode the address of 65,536 words. For the FPLA, the matrix has to be only large enough to store the address of 48 words: the FPLA's P-terms. The advantage comes about because here we have a choice to select a minimum of any 48 input words (or more, as determined by Don't Care input variables) from a total available pool of 65,536.

Due to the unique capability of FPLAs to store directly Don't Care (X) input states, each internal word (W) in the device Storage Matrix can be addressed by several logic input combinations (minterms, given by:

$(M_n)T = 2^{m-r}$

Where m = total number of input variables

r = number of active inputs (true or complement) contained in a programmed P-term column.

Thus, if P_t = $XXXI_0$, m = 4 and r = 1, for which $(M_n)T = 8$.

SIGNETICS' FPLAs

The following diagram (Fig. 4) shows the logical structure of Signetics' FPLAs. Both devices consist of an upper AND matrix containing 48 product term columns (P-terms), and a lower OR matrix containing 8 sum term rows (S-terms), one for each output function. Each P-term in the AND matrix is initially coupled to each input variable via 2 Schottky diodes for programming the desired input state, and to each S-term in the OR matrix thru an emitter follower with an emitter fuse, for pulling the summing node to a HIGH level when the P-term is activated. Each S-term in turn is coupled to its respective output via an EX-OR gate, which has programmable transmission polarity by means of an input to ground through a fusible link.

In its initial unprogrammed state all Ni-Cr links are intact, such that:

- Each P-term contains both true and complement values of every input I_m. Hence all P-terms are in the "NULL" state (always LOW).
- Each S-term contains all 48 P-terms.
- The polarity of each output is set to active-HIGH (F_p function). Since all P-terms are inactive, all outputs will be at a LOW level when the chip is enabled ($\overline{CE}$ = LOW), regardless of input conditions.

The transmission thru the FPLA can be traced along the equivalent logic path shown in Fig. 5. (Note that for each of the 8 outputs, either the function F_p (active-High) of F_p^* (active-LOW) is available, but not both). This emphasizes the dual aspects of FPLA, in that they can be viewed as **Conditionally Addressable Memories** or as AND-OR logic blocks, depending on the application.

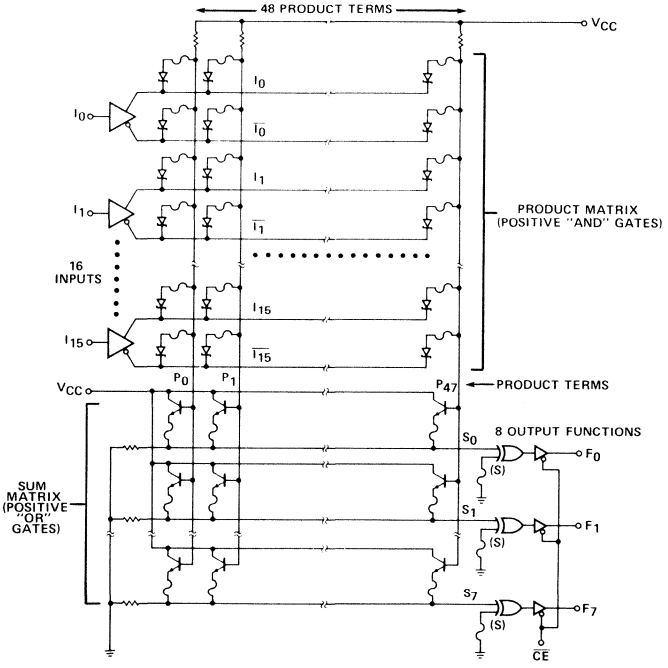


FIG. 4: Logic structure of FPLA, illustrating AND, OR, and EX-OR arrays.

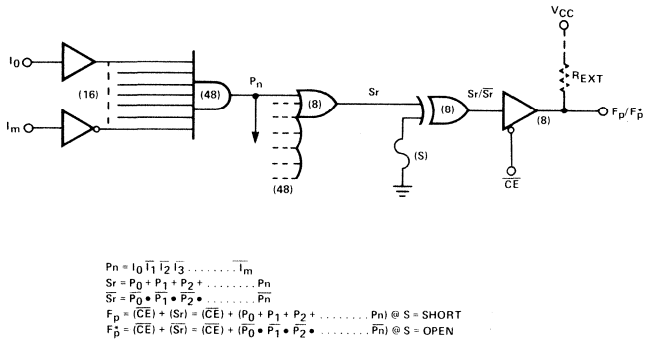


FIG. 5: Typical logic path of Signetics' FPLA. A general set of equations describing the device logic transmission is shown in the truth table on page 4.

PROGRAMMING

The FPLA is programmed by the user to contain the desired Program Table (or Truth Table) in three successive steps involving the AND matrix, the OR matrix and the transmission polarity of the output EX-OR gates. For this purpose, automatic programming equipment is currently available on the market

The peripheral fusing circuitry internal to the FPLA, and the basic, terminal requirements which must be provided to fuse the three sectors of the FPLA are shown in Figs. 6 and 7, respectively. For a more detailed fusing procedure, refer to the device data sheet.

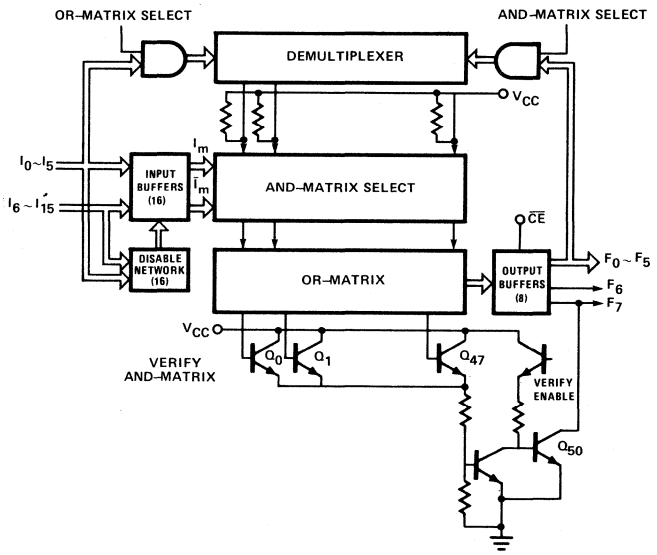


FIG. 6: Functional FPLA blocks activated during ARRAY Program/ Verify sequence.

PROGRAM →	'AND' MATRIX	'OR' MATRIX	OUTPUT ACT LEVEL
V_{CC}	+5.0 V	+8.75 V	LOW
INPUT(S) (Program) I_m	HIGH	ADDRESS P-TERM	HIGH
INPUT(S) (Program) $\overline{I_m}$	LOW	WITH I_0 ~ I_5	
OTHER INPUTS	+10.0V		
OUTPUT(S) Contains P_n	ADDRESS P-TERM	LOW	HIGH / LOW
OUTPUT(S) Excludes P_n	WITH F_0 ~ F_5	+10.0 V	LOW / 0V / 1MS / +17.0V
Fuse Enable	+17.0 V	+17.0 V	LOW
$\overline{CE}$	+10 V HIGH 1MS	+10 V HIGH 1MS	HIGH

FIG. 7: Summary of FPLA terminal requirements for programming respective areas in the device.

Each P-term P_n is programmed to contain the desired logic state of each input variable by fusing the appropriate Ni-Cr link in the pair which couples each P-term to each input variable. If P_n contains I_m, the $\overline{I_m}$ link is fused, and viceversa. If I_m is a Don't Care in P_n, both the I_m and $\overline{I_m}$ links must be fused. If fewer than 16 variables are used in any application, the unused variables represent Don't Care conditions for all used P-terms, and their corresponding I_m and $\overline{I_m}$ links **must** in general be fused (see **Editing**, below).

The response of each output function to the presence of active P-terms is programmed in the OR matrix. If any product term P_n logically Negates an output function, the link coupling that output function to the P-term(s) must be fused and viceversa. No programming is required of OR matrix links coupling used or unused P-terms to S-terms servicing any **unused** output functions. Also, since in a blank device all P-terms are in a logic "NULL" state, **unused** P-terms require no programming at all.

EDITING SIGNETICS' FPLA

In contrast with PROMs, FPLAs have inherent program editing capabilities. After programming, the user can incorporate a number of program modifications in Signetics' FPLAs. These are tabulated in Fig. 8.

ADD	P-Term To F_p/F_p^*	YES	Program desired logic function into any unused P-Term. Blow S-Term link(s) coupling P-Term to undesired output functions.
	$I_m/\overline{I_m}$ To P-Term	NO	Delete erroneous P-Term. Add new, corrected P-Term.
DELETE	P-Term From F_p/F_p^*	YES	Blow S-Term link coupling P-Term to F_p/F_p^*.
	$I_m/\overline{I_m}$ From P-Term	YES	Blow both links coupling the input variable to the P-Term.
CHANGE	$F_p \longrightarrow F_p^*$	YES	Blow EX-OR link of output to be inverted.
	$I_m/\overline{I_m} \longrightarrow X$	YES	Delete $\overline{I_m}/I_m$ from P-Term.
	$I_m \longleftrightarrow \overline{I_m}$	NO	Delete erroneous P-Term, and add a new P-Term.
	$F_p^* \longrightarrow F_p$	NO	Use spare active-HIGH output.

FIG. 8: Summary of "EDITING" features of Signetics' FPLA.

DISPOSITION OF UNUSED INPUTS

When a particular application involves less than 16 input variables, if unused *inputs are programmed as Don't Care in all* used *P-terms (M) of the FPLA, it is no longer possible to modify the logic structure of the (M) P-terms by reinstating any of the unused inputs as additional controlling variables to the FPLA.*

While it is possible to recover from this condition by deleting P-terms requiring change, and adding any of the remaining (48–M) P-terms programmed with the desired number of input variables, this method ultimately fails once all 48 P-terms are exhausted.

This method can be combined with an alternate procedure to obtain a greater degree of flexibility in adding previously unused inputs to a preprogrammed FPLA. It requires that about one half of all originally unused inputs be programmed HIGH and the remaining half LOW, in (M)P-terms only. These inputs are then normally tied to HIGH and LOW logic levels respectively.

If at anytime during function update or modification it becomes necessary to add HIGH and/or LOW control variables to (N) of the (M)P-terms, any of the properly programmed idle inputs are disconnected from their voltage clamps and connected to their corresponding logic sources. These newly activated inputs must in turn be reprogrammed as Don't Care in (M–N) of the used P-terms.

GENERATING THE FPLA PROGRAM TABLE

In a typical application as in Fig. 9, the symbolic statements, or the truth-table, of a logic problem are first reduced to a minimum set of P-terms, and then Program Table entries are derived from an "Activity Map".

$\overline{F_0} = P_0 + / + P_2$
$F_1 = / + P_1 + P_2$

a. Activity Map of Elementary FPLA. (/) = F_x excludes P_n.

$P_0 = I_2 \overline{I_1}\ \overline{I_0}$
$P_1 = I_2 \overline{I_1} I_0$
$P_2 = \overline{I_2} I_1 \overline{I_0}$

b. P-term List

FIG. 9: Elementary program to be stored in FPLA.

The standard Program Table format adopted by Signetics is shown in Fig. 10. The term "Program Table" is used in favor of Truth Table, because the former allows Don't Cares (X) as a **direct** entry, and thus is more general, and conforms to FPLA structure. The Activity Map shows those P-terms which are contained in an output function, and those which are not, designated by (P_n) or (/) in their respective positions. The presence of a (/) in the Activity Map, although not strictly necessary, is convenient in generating the Output Table section of the Program Table (especially while interacting with a pro-

FPLA PROGRAM TABLE

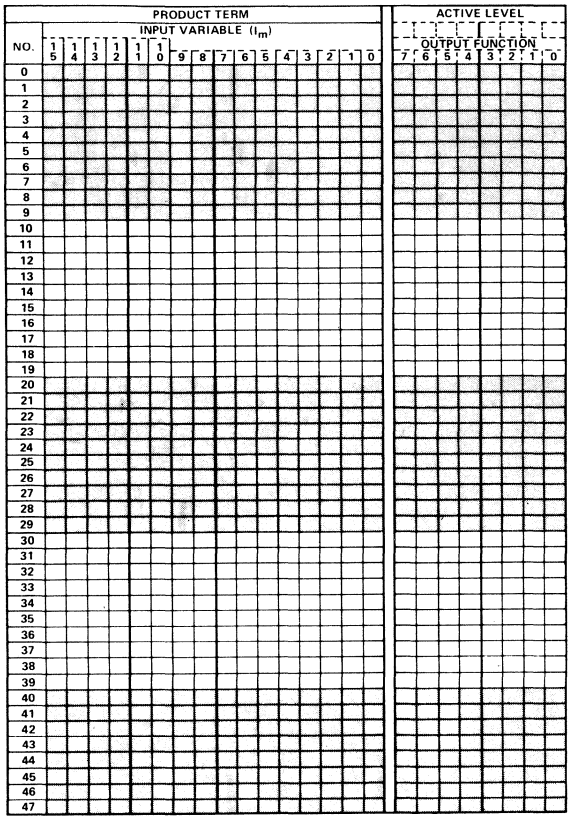


FIG. 10: Standard Signetics' Program Table. Used P-Terms can be programmed anywhere. Unused P-Terms require no programming, and can be left blank.

gramming system) because it makes it easier to enter the program by taking vertical slices through the Activity Map while the P-terms are sequentially addressed.

Ideally the FPLA Program Table should contain entries formulated with a code which not only issues unambiguous fusing commands to a programming system, but also readily displays the actual logic state of the FPLA outputs. In dealing with logical statements or truth-tables, most logic designers are used to either (1/0) or (H/L) symbols. An additional symbol (X) is generally used for Don't Care input states. Their widespread usage is a strong incentive to choose among these symbols for a suitable set to code the FPLA Program Table.

However, in many cases the Program Table will be transmitted to remote programming centers over commercial communication links, which normally employ an ASCII alpha-numeric code. Since the "distance" between the ASCII codes for "0" and "1" is only one bit, the risk of undetected transmission errors is large. Thus, the set (H, L, X) is more preferable, but it is still not ideal. Indeed, to enhance the production of low cost programming equipment, in which a 7-segment LED display is mandatory, one must forego the (X) in favor of a more realizable symbol such as a (—).

Therefore, to elicit unambiguous fusing commands from a programming system, the state of all input variables in each P-term is coded as illustrated in Fig. 11.

All entries clearly indicate the logic states of the input variables which activate a given P-term. An additional symbol for the input variables is required to code the state in which both I_m and $\overline{I_m}$ links are intact. This state, chosen as (0) again for ease of display, occurs only in a virgin device, or for variable(s) of unused or partially programmed P-terms. It is the **initial** state of all input variables, signifying their logic NULL state. If any P-term contains at least one variable in the NULL state, the P-term will never be selected by any logic input combination. Entry of a (0) in the Program Table is thus meaningless, and not allowed. However, it does require to be displayed by a programming system to indicate blank check results, or program fail conditions.

While these symbols are appropriate to code the various states of the FPLA input variables for each P-term, as well as the output Active Level polarities, they give rise to some ambiguities when used to code the FPLA outputs, because of the user choice of output Active Level. To code the outputs of the FPLA, several alternatives are available. In all cases, derivation of each entry involves scanning the Activity Map to determine whether or not an output function contains a particular P-term. Regardless of

$P_n \stackrel{?}{=} f(I_m)$	Input State	Program Table Entry	Fuse Command
Yes	I_m	Ⓗ	Fuse $\overline{I_m}$ link
	$\overline{I_m}$	Ⓛ	Fuse I_m link
No	Dont Care	⊖	Fuse both

FIG. 11: Program Table coding of input variables. Entries contained in ○.

chosen output polarity, a P-term activates F_p if it is **contained** in F_p. Accordingly, any F_p will be forced HIGH, and $\overline{F}_p$ (defined as F_p^*) will be forced LOW. Conversely, if P_n is not contained in

an output, all F_p and F_p^* functions will remain in their **default** logic state (LOW or HIGH, respectively). A particularly convenient method for coding the FPLA Output-Table is shown in Fig. 12.

$F_p \stackrel{?}{=} f(P_n)$	Activity Map	Output Table		Fuse Commands
Yes	P_n	Ⓐ	Ⓐ	Do not Fuse P_n link in OR-Matrix
No	/	⊙	⊙	Fuse P_n link in OR-Matrix
	Active Level	H	L	
		F_p	F_p*	Function Polarity

FIG. 12: Table for formulating Output Table entries for the FPLA. Entries, contained in ○, are obtained by "multiplying" the contents of the Activity Map with the Active Level.

This coding system utilizes an Ⓐ (for Active) to indicate the presence of P_n in either F_p or F_p^*, and a ⊙ (period) to indicate absence. It has the advantage that the FPLA Output Table can be constructed directly from the Activity Map. Also, when retrieving the stored Output Table from a programmed device, the presence/absence of a P-term in an output function is readily detected, yielding the easiest array verification procedure. However, in order to relate the actual logic output of the FPLA to the above entries (especially when dealing with code conversion, or address translations), reference to the following table is necessary:

FPLA Output Table	FPLA Logic Output	
Ⓐ	H	L
⊙	L	H
	Ⓗ	Ⓛ Active Level
Function Polarity	F_p	F_p*

FIG. 13: Table for calculating the FPLA logic output. The FPLA output is obtained by "multiplying" Output Table entries with the Active Level.

VERIFYING THE STORED PROGRAM

Unlike PROMs, verification of an FPLA after programming, or examination of the contents of a master device for duplication or diagnostic purposes, is no trivial task. Unique difficulties are posed by the large number of inputs to be manipulated, and by the multiple and concurrent addressing modes characteristic of FPLAs.

In general, the FPLA Program Table may bear little resemblance to the original truth-table, yet, from a black box viewpoint, the logic function of the FPLA should match entry for entry the original truth-table. This level of verification can only be obtained through a logic verification procedure, in which the logic transfer characteristic of the FPLA is exhaustively examined by exercising its inputs with a minterm generator.

But, while logic verification is the ultimate test of FPLA valid function, it is a useless tool for determining the FPLA stored program. This is readily apparent in Fig. 14 which shows the output of an elementary FPLA to be the same (LOW) for three distinct internal programmed states, when its single output is toggled between HIGH and LOW logic levels.

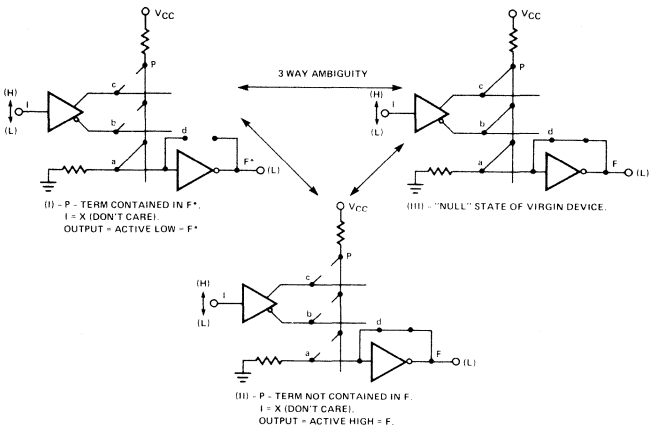


FIG. 14: Distinct FPLA programmed states resulting in identical logic function.

Therefore, a non-ambiguous map of the status of every link in the device is a most essential tool required to monitor and manipulate the stored program, especially while interacting with an FPLA programming system. When duplicating from a master device, the programming system must derive precise and unambiguous instructions for programming the slave. Signetics' devices allow such map to be obtained via an Array Verify test sequence comprising three tests for examining the links in the output EX-OR, the AND-matrix, and the OR-matrix.

ARRAY VERIFY

The peripheral fusing circuitry in Signetics' FPLAs incorporates additional networks and

dedicated paths for the Array Verify test sequence. These are shown at the bottom of the composite FPLA diagram in Fig. 6. Specifically, to sense the status of the AND-matrix links, the OR-matrix includes an extra row of non-fusible emitter followers Q_0 through Q_{47}, monitored via Q_{50} collector ORed with output F_7. This stage does not interfere with F_7 during normal operation because Q_{50} can only get base drive during verify mode. The internal map of the FPLA is obtained by performing the sequence of tests summarized in Fig. 15, during which the Fuse Enable input is maintained at ground. Verification of the active level polarity of the outputs is simply obtained by addressing a non-existent

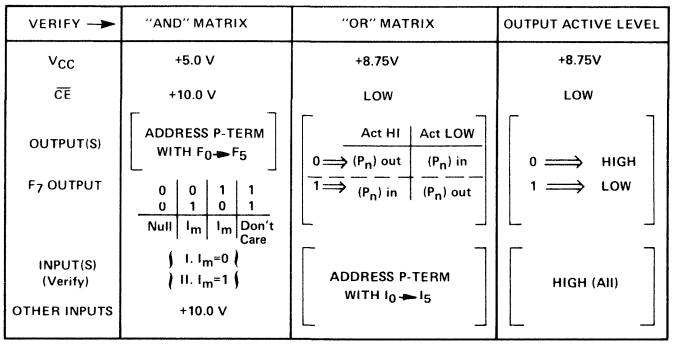


FIG. 15: Summary of FPLA terminal requirements for mapping the status of all internal links. The Output Active Level test must be performed *before* the OR-Matrix test.

I_m	F_7	Input Variable State Contained In P-term	Input Code
L H	H L	$\overline{I_m}$	L
L H	L H	I_m	H
L H	H H	Don't Care	–
L H	L L	(I_m), ($\overline{I_m}$)	0

FIG. 16: Table for determining the status of each Input Variable link in the AND-MATRIX.

P-term in the device, and thus rely on the pull-down resistors in the "OR-matrix" to yield a non-ambiguous result.

To verify the "AND-matrix" two tests are required for each input for each of the P-terms. The status of each I_m link coupling a P-term to the input buffer outputs is determined in accordance with the table in Fig. 16.

Verification of the "OR-matrix" requires prior knowledge of the output level polarities. The status of the OR-matrix links coupling each P-term to the S-term is given by the table in Fig. 17.

For a more detailed Array Verify procedure refer to the device data sheet.

Output		P-term Link
Active-HIGH (F_p)	Active-LOW (F_p*)	
L	H	FUSED
H	L	PRESENT

FIG. 17: Table for interpreting the status of OR-MATRIX links, based on Output Active Level test results.

LOGIC VERIFY

After an FPLA has been programmed, and its contents checked by Array Verify against hard-copy reference of the Program Table, there should be in most cases little reason to suspect that the device will not exhibit the correct logic function in a system environment. However, in some cases, device defects, programming equipment problems, user coding inexperience, as well as system logic races and other marginalities, may all contribute in creating a situation in which system failures are traced to an FPLA which nevertheless appears to contain the correct Program Table. In these cases, further device diagnostics are necessary to identify the source of the problem at hand, for which the actual operating system may be a slow and ineffective tool.

Also, at the end of the design cycle, some users may want to replace FPLAs with *mask* programmable PLAs for cost reduction. Since a PLA does not contain peripheral fusing circuitry, it is not possible to logically address each of is internal links to verify that the PLA contains the same Program Table as the Master FPLA. In this case the only verification possible is a full logic verify of the PLA vs. FPLA functions.

Ultimate verification of FPLA logic performance entails an exhaustive check of its logic function to compare the *expected* Truth-Table with the *stored* Truth-Table, obtained by cycling the FPLA inputs through all 2^{16} combinations with a Minterm generator. This, however, involves dealing with a hardcopy reference of a table containing about 64 thousand input entries, which is a totally impractical task in view of what may be required to generate and store such table.

A more feasible alternative consists of con-

structing a "hardwired" Logic Verify system which may be conveniently incorporated within the FPLA Programming system. The Programmer would then function as an FPLA emulator with the ability to produce and display the full Truth-Table of the FPLA, viewed just as a logic box. This is extremely useful in code conversion, map translations, or when programming directly from a truth-table.

In essence the Logic Verify system must be able to compare the actual FPLA logic output with that computed on-the-fly by composite overlay and manipulation of the Output Table stored in the Programmer, as activated by all concurrent and multiple address selections for each state of the input minterm generator.

The Logic Verify procedure presumes knowledge of the Program Table stored in the device; hence, it must necessarily follow an Array Verify operation to first scan and store in the system main memory the Program Table contained in the device under test. A comparison of the actual versus computed Output Tables, in conjunction with a direct display of the FPLA logic output for each minterm input, will reveal all discrepancies. For this, a CRT display is mandatory.

To be useful, the Logic Verify procedure must also be fast. It should be complete within 5 to 10 seconds per device, and thus dictates use of a hardwired algorithm. The block diagram of a logic subsystem which executes a suitable algorithm, outlining basic hardware, controls, and data paths is shown in Fig. 18.

The algorithm manipulates Program Table data stored in Main Memory and Active Level Register, in the format tabulated in Fig. 19. Before loading the Program Table, M/M and the ALR are reset to "O", to clear all previously stored fusing commands. A binary counter, conditionally incremented, functions as minterm (M_n) generator, for addressing the FPLA with all 2^{16} input combinations. The FPLA output for each M_n input is stored in Register B. All 48 P-terms are fetched one at a time from the program table in M/M, and examined to determine whether they *logically contain* each M_n. The criteria which logically include or exclude M_n from a P-term are tabulated in Fig. 20 for all general programmed states. If the test fails, a new P-term is fetched, and the test repeated until all 48 P-terms have been examined, and all 2^{16} minterms are exhausted. On the other hand, if the test indicates

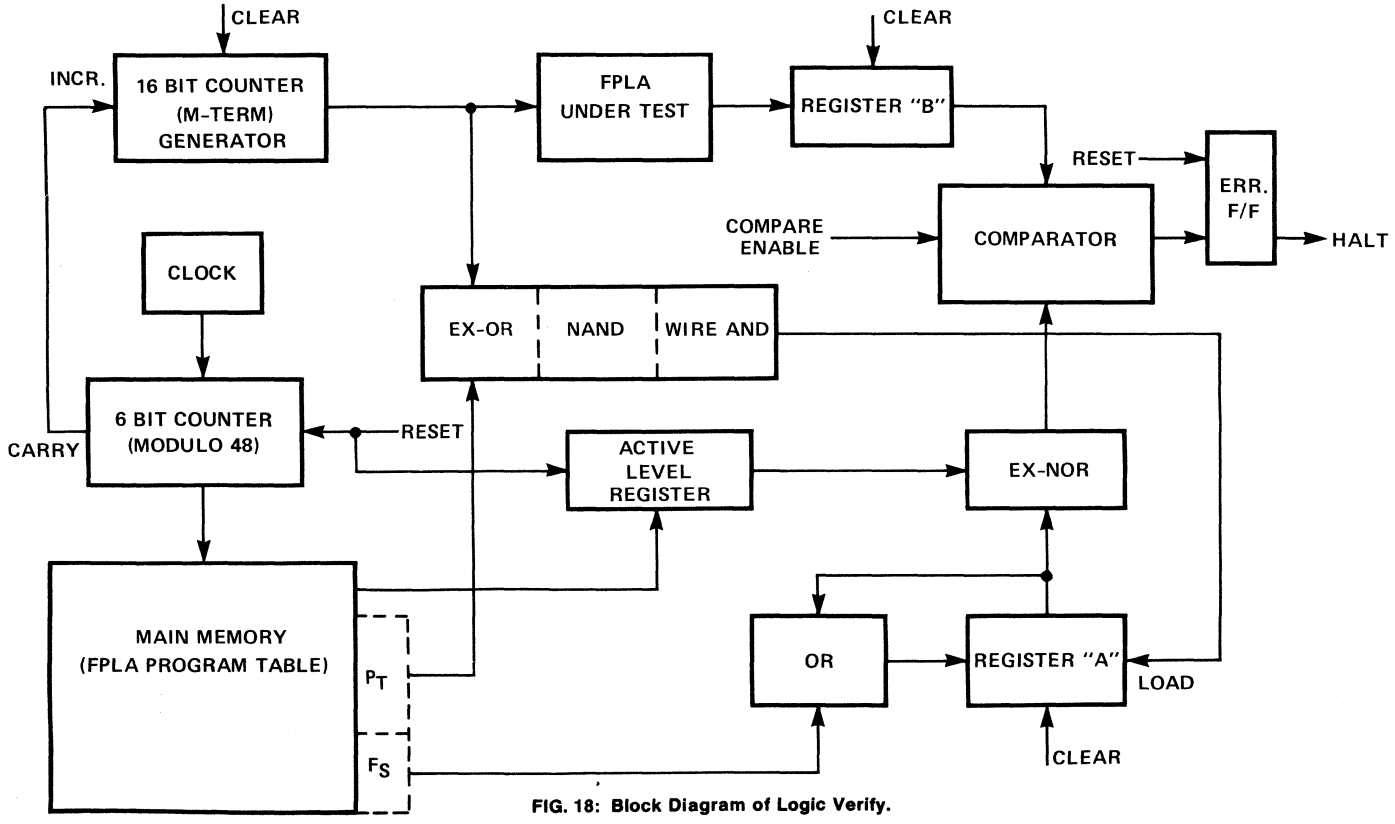


FIG. 18: Block Diagram of Logic Verify.

	Address	Data									
	P-term #	P-term Field					F-Set Field				
		I_{15}	I_{14}	.....	I_0		F_7	F_6	.....	F_0	
Stored Format	Sequential	0 1	1 0	.....	1 1		0	1	.....	0	
Typical Entry	27	H	L	.....	—		A	•	.....	A	

a. M/M binary format and typical entry.

	F_7	F_6	.....	F_0
Stored Format	0	1	.....	0
Typical Entry	H	L	.....	H

b. ALR binary format and typical entry.

FIG. 19: Binary assignment of FPLA Program Table stored in Main Memory, and Active Level Register.

	I				II					III				
M_n	H	H	L	L	H	H	H	L	L	H	H	L	L	L
P-term	H	–	L	–	L	H	–	L	–	H	–	L	–	H
	M_n contained in P-term				M_n not contained in P-term									

FIG. 20: Criteria for the logical inclusion/exclusion of a minterm in a P-term. (H↔L) preclude logical inclusion.

that M_n is contained in the P-term, the F-set field associated with the addressed P-term is *overlaid* in Register A, while the M/M address of the P-term is stored in a stack containing the *concurrent* P-term list, and a presence flag set to indicate that the P-term address is a valid member of the list.

Testing continues until all 48 P-terms have been compared to the M_n count. At this point, Register A contains a *composite* FPLA Output Table obtained when all concurrently selected P-terms are activated by M_n at the FPLA inputs. This table is merged through an EX-NOR with the contents of the ALR to produce a composite binary F-set, which is in turn compared with the contents of Register B. If they are equal, the M_n generator is incremented, and the test sequence repeated with M_{n+1} until the last minterm. (Alternately, if in manual mode, before incrementing M_n one could observe the Logic Output of the FPLA with M_n as input by calling the contents of the display buffer). If the contents of Registers A and B differ, an error flag is set, and the M_n count halted. The following housekeeping displays occur, and the system will wait until a continue command:

- The concurrent P-term list is scanned and displayed in the designated field on the CRT.
- The contents of the M_n generator are displayed in the hexadecimal M-term count field, while its binary equivalent (presented to the FPLA inputs) is displayed in the INPUT field.
- Results of the EX-NOR of Register B with the contents of the ALR are displayed in the OUTPUT field. This yields the Output Table obtained from the *device* with M_n as input.
- The contents of the ALR are displayed in the ACT LEVL field.
- The contents of Register A are displayed in the COMPUTED OUTPUT TABLE field. They indicate the composite Output Table *expected* from the FPLA with input M_n.
- The contents of Register B are displayed in the PLA OUTPUT field. They indicate the logic levels present at the FPLA outputs.

A suitable display of this information is shown in Fig. 21. All error conditions detected during Logic Verify will produce conflicting indications in the PLA OUTPUT TABLE versus the COMPUTED TABLE. From Fig. 21, the presence of Ⓐ in the PLA OUTPUT TABLE versus a ⊙ in the COMPUTED TABLE suggests an illegal concurrency in the device. Conversely, the ⊙ in F_0 and F_6 in contrast with an Ⓐ for the same bits in the COMPUTED TABLE indicates inherent concurrencies absent in the device. Knowing all concurrent P-terms and the logic input to the FPLA, we can resort either to Array Verify or hardcopy reference of the Program Table and Activity Map for further diagnostics and isolation.

```
M–TERM'     LOGIC VERIFY      'ACT LEVL
[FA76] '  ..................' HHLHHHLH
''''''''' [INPUT VARIABLE]  '''''''''
  <P>     111111              [OUTPUT]
  <L>     5432109876543210    76543210
  <A> ... HHHHHLHLLHHHLHHL    A••AA•A•
[COMPUTED OUTPUT TABLE]       AA•A••AA
[ERROR]..................      ↑  ↑  ↑
[PLA LOGIC OUTPUT] .......    HLHHHLLL
-----------------------------------
ERROR: P-TERM CONFLICT
CONCURRENT P-TERM LIST: 0, 1, 2, 3, 4
```

FIG. 21: Logic verify of FPLA, yielding device truth-table for logic input FA76 (HEX). Output bits in error indicated by arrow.

DEALING WITH DEVICE LIMITATIONS

In many applications a single FPLA cannot

accomodate the full Program Table because it exceeds device limitations arising from the finite number of inputs, outputs, and P-terms available. In many cases this can only be overcome by resorting to design intuition and ingenuity in place of complex data manipulations which tend to obscure the problem on hand, and may render troubleshooting difficult.

Borderline cases can usually be resolved by judicious inspection of the Program Table to discover ways to further compression. Nevertheless, to increase design flexibility in these situations, Signetics' FPLAs are the only ones which feature a Chip Enable input which can be used for input and P-term expansions, preconditional input decoding, and output inhibit.

The output inhibit function of $\overline{CE}$ not only permits utilization of the tristate device in bused organizations, but also provides a means to force all outputs to a unique logic state, regardless of their programmed polarity, without sacrificing FPLA inputs or entailing additional hardware. This feature is essential in a number of applications involving system initialization from a known state, exit to "idle" following sequence error, synchronous clocking, etc. For example, in the typical sequencer of Fig. 22, if an input error occurs parity fails, forcing all outputs to logic "1" ("IDLE" state, by user definition).

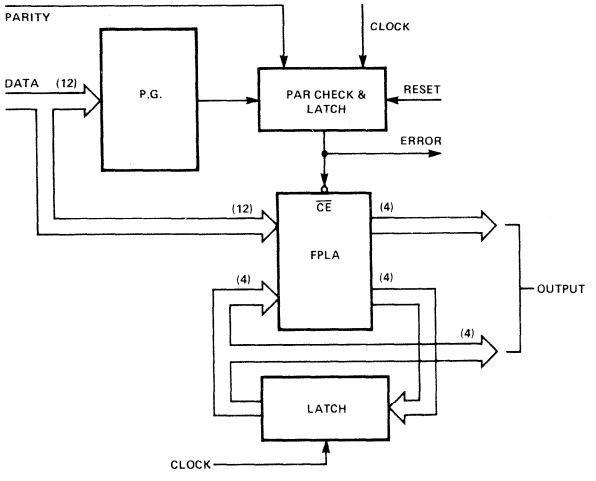


FIG. 22: Sequential Controller forced into "IDLE" state by input parity error.

PRODUCT TERM EXPANSION

Expansion of P-terms involving up to 16 input variables is easily accomplished with Open Collector devices, as shown in Fig. 23. It is only necessary to parallel respectively all inputs and outputs of several devices, operated with $\overline{CE}$ at ground (unless needed as additional control function). The composite logic output of the network is determined by P-terms activated in one or several FPLAs simultaneously.

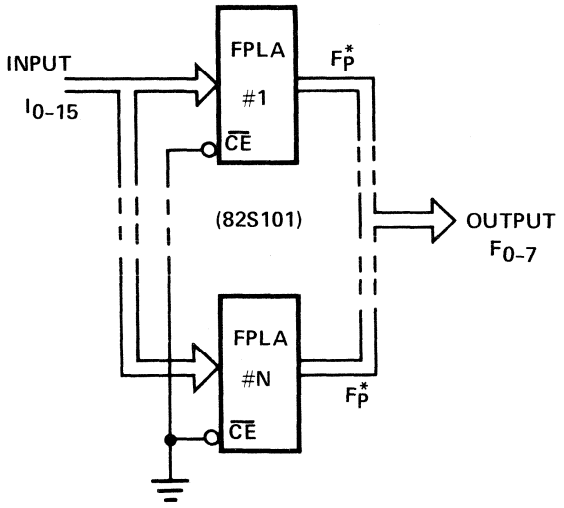


FIG. 23: P-term expansion with Open Collector FPLAs, involving up to 16 input variables. All outputs must be programmed active-LOW (F_P^*) to realize the wire-OR function. The total number of P-terms available is 48N.

When using tristate devices (82S100), P-term expansion cannot be readily achieved in the same way because of logic conflicts ensuing from the active pull-up outputs of FPLAs sharing the same output bus. To ensure enabling only one device at a time, P-term expansion *must* involve the $\overline{CE}$ input.

In most applications requiring more than 48 P-terms it should be a relatively simple task to partition the Program Table in 2 or more Subtables, each containing less than 48 P-terms which in turn can be fitted in separate FPLAs. This partitioning is achieved by segmenting the original Table about the "1's" and "0's" of suitable input variables. Since all P-terms P_n which contain a segmenting variable as Don't Care give rise to 2 P-terms P_{na} and P_{nb}, it is best to segment a Program Table about variables with the fewest Don't Care states.

The logic sources of segmenting variables are removed from the FPLA input field and made to drive instead the $\overline{CE}$ input of the required FPLAs, after proper decoding. As an example, if one were restricted to use tristate FPLAs with only 10 P-terms each to incorporate the Program Table of Fig. 32b (page 31), a segmentation of this Table about input I_2 yields the Subtables shown in Fig. 24.

Each Subtable contains less than 10 P-terms, and will fit in separate FPLAs which are operated in parallel and controlled by I_2 via their $\overline{CE}$ input, as shown in Fig. 25.

The feasibility of this procedure is strongly dependent on the contents of the original Program Table, and in some degenerate cases (too few or no "0's" at all in the input field of the Program Table) it may not work. Also, note that in general the final number of P-terms used may increase due to expansion of input Don't Cares. However, this is preferable to no solution at all.

P-terms		INPUTS				OUTPUTS							
P_n	P_n	I_3	I_2	I_1	I_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0
0a	0	X	0	X	1	0	0	0	0	0	0	0	1
1a	1	X	0	1	0	0	0	0	0	0	1	0	0
3	2	X	0	1	1	0	0	0	0	1	0	0	0
6	3	1	0	X	1	0	0	0	1	0	0	0	0
7	4	1	0	1	X	0	0	1	0	0	0	0	0
10	5	1	0	X	X	0	1	0	0	0	0	0	0
11a	6	1	0	1	X	0	1	0	0	0	0	0	0

FIG. 24a. Subtable A to be stored in FPLA #1 with I_2 removed. P_{11a} can be eliminated since it is "covered" by P_7.

P-terms		INPUTS				OUTPUTS							
P_n	P_n	I_3	I_2	I_1	I_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0
0b	0	X	1	X	1	0	0	0	0	0	0	0	1
1b	1	X	1	1	0	0	0	0	0	0	1	0	0
2	2	X	1	0	1	0	0	0	0	1	0	0	0
4	3	X	1	0	0	0	0	0	1	0	0	0	0
5	4	0	1	X	1	0	0	0	1	0	0	0	0
8	5	1	1	X	1	0	0	1	0	0	0	0	0
9	6	0	1	1	X	0	0	1	0	0	0	0	0
11b	7	1	1	1	X	0	1	0	0	0	0	0	0
12	8	1	1	X	X	1	0	0	0	0	0	0	0

FIG. 24b. Subtable B to be stored in FPLA #2, with I_2 also removed.

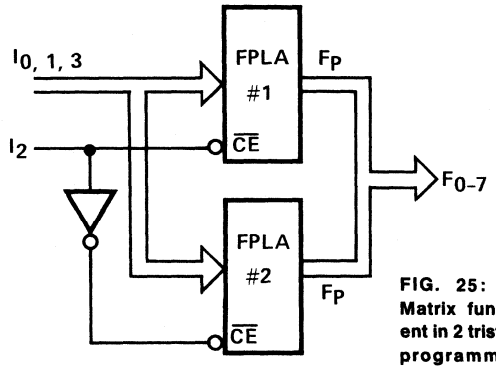


FIG. 25: Squaring Matrix function resident in 2 tristate FPLAs programmed respectively with Subtables A and B. Note that the inhibit function of $\overline{CE}$ renders unnecessary to program the outputs active-LOW.

INPUT VARIABLE EXPANSION

This is the most difficult and cumbersome task with FPLAs. When the Program Table involves more than 16 inputs, the above partitioning technique by Subtables segmented about any suitable variables will work as well with tristate or open collector devices. This technique is shown applied to 18 input variables in Fig. 26. In this case several devices are necessary, even though not all FPLA P-terms are used.

Note that the expansion capability provided by $\overline{CE}$ input limits the total number of FPLAs required to 2^n, where (n) is the number of segmenting variables. Without $\overline{CE}$, the total number of FPLAs required would be 2^{n+1}.

With more than 20 or so inputs this approach may become too costly, and thus it may make more sense to review the Program Table in conjunction with the problem at hand for ways to multiplex the FPLA inputs. This also involves a sort of segmentation of the Program Table for grouping P-terms about input variables which are mutually exclusive.

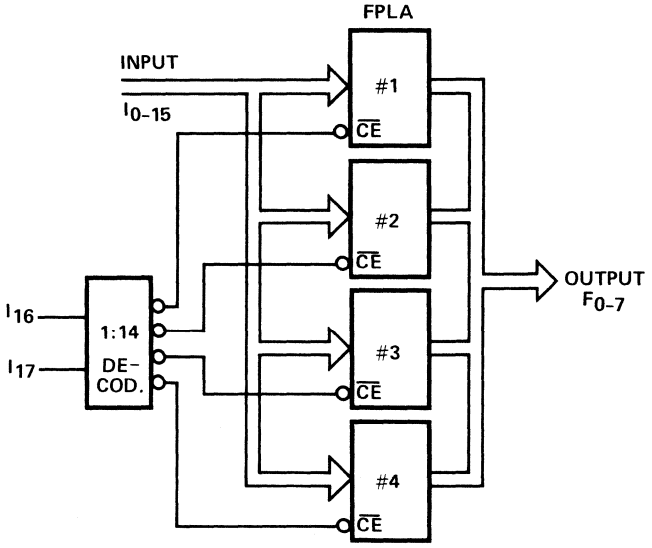


FIG. 26: Direct manipulation of 18 input variables using $\overline{CE}$ with either 82S100 or 82S101 FPLAs. Note that here it is not necessary to program all output functions active-LOW (F_p^*) because of the disabling function of $\overline{CE}$.

The principle is illustrated in Fig. 27a when dealying with only 17 input variables and 5 P-terms, for simplicity.

The original Program Table in Fig. 27a has been segmented about the "0's" and "1's" of variable I_n, and the P-terms regrouped as in Fig. 27b. Note that it was necessary to create new P-terms 4a and 4b to expand the Don't Care for I_n in P-

term 4. Also, it is readily apparent that when $I_n = 0$, the outputs are independent of I_{n-1}, and when $I_n = 1$ the outputs are independent of I_{n+1}. These inputs can be multiplexed in an FPLA with I_n as the steering condition, as shown in Fig. 28.

The FPLA Program Table contains Upper P-terms with I_{n-1} variable removed, and Lower P-terms with I_{n+1} variable removed.

When this technique fails too, it may still be possible to factor out of the logic equation of each FPLA output common expressions involving the variables in excess. These can be externally combined with simple gating, or another FPLA, into first level P-terms generating *dummy* variables to be applied to a second-level FPLA.

P_n	I_{16} . . .	I_{n+1}	I_n	I_{n-1} . . .	I_0	F_x	F_y
0	0 . . .	X	1	0 . . .	1	1	0
1	1 . . .	1	0	X . . .	1	1	1
2	X . . .	0	0	X . . .	0	0	1
3	0 . . .	X	1	X . . .	X	1	0
4	1 . . .	X	X	X . . .	0	0	1
5	1 . . .	X	1	1 . . .	0	1	0

FIG. 27a. Initial Program Table involving 17 input variables, which cannot be directly examined by a single FPLA.

	P_n	I_{16} . . .	I_{n+1}	I_n	I_{n-1} . . .	I_0	F_x	F_y
UPPER	1	1 . . .	1	0→X	. . .	1	1	1
	2	X . . .	0	0→X	. . .	0	0	1
	4a	1 . . .	X	0→X	. . .	0	0	1
LOWER	0	0 . . .	X←1	0	. . .	1	1	0
	3	0 . . .	X←1	X	. . .	X	1	0
	4b	1 . . .	X←1	X	. . .	0	0	1
	5	1 . . .	X←1	1	. . .	0	1	0

FIG. 27b. Variable I_{n+1} and I_{n-1} can be multiplexed on a single FPLA input because they are mutually exclusive "about" I_n (selector).

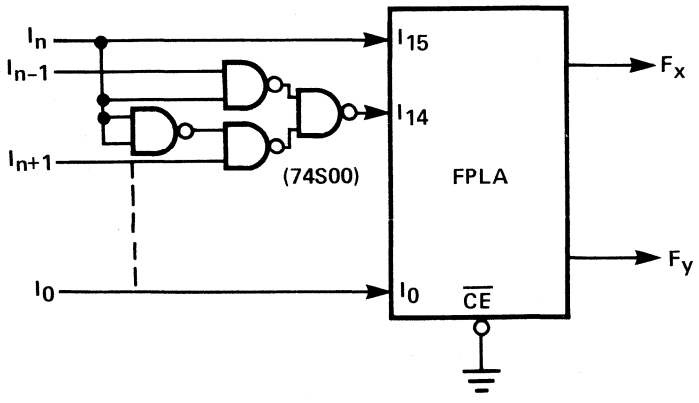


FIG. 28: Multiplexing of inputs I_{n+1} and I_{n-1} with selector input I_n allows 17 inputs to be handled with one 16-input FPLA.

OUTPUT EXPANSION

If an application requires more than 8 outputs, several FPLAs can be used with parallel inputs and separate outputs. In other cases, it may be more cost effective to encode the Output Table stored in a single device, and then unscramble the desired output states via a 32X8 PROM, or 1/N decoder as required. Both methods are shown in Fig. 29.

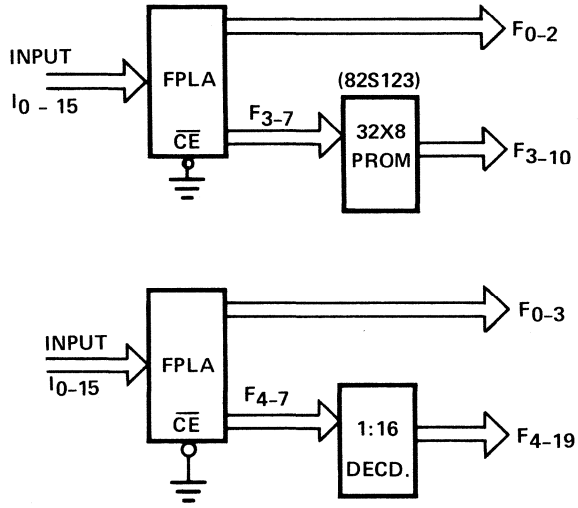


FIG. 29a. Output expansion by decoding outputs previously encoded in the FPLA Program Table.

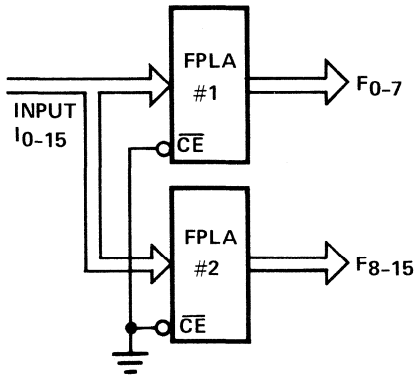


FIG. 29b. Output expansion by utilizing additional FPLAs.

The recent surge in design activity involving microprocessors and microprogramming techniques reflects the growing trend to replace hardwired logic with microcode for gaining system flexibility at lower cost. In this respect, designers have come to rely on ever larger and denser PROMs to fit the demands of their applications, and today PROMs as large as 4K-bits, organized as 512X8 or 1KX4, are readily available. However, a PROM solution in general forces the user to allocate storage for all possible logic combinations of the input variables, whether needed or not. As a result, when dealing with the type of problem requiring the manipulation of more than about 10 logic input variables (or Addresses), several IC packages are usually necessary. This quickly renders a PROM solution marginal at best, in terms of speed, power, and cost, and in most cases impractical. Fortunately, many combinational and sequential logic designs involve logic functions which are True for only a small subset of the total logic states generated by the controlling variables. Typical examples are the 96 graphic characters, out of 2^{12} coding states, of a 12-bit Hollerith code; or the 50 (or so) subroutine-start addresses, from a total of 2^{16}, in a typical 16-bit microprogrammed machine. It is here that we step in the basic domain of Field Programmable Logic Arrays which, when viewed as Associative memories, exhibit Selective, Concurrent, and Multiple addressing modes that enable compressing a set of logic functions to the minimum required states, at substantial savings in hardware.

The areas of application in which FPLAs provide a more efficient design alternative span the whole spectrum of logic design. Many applications based on mask-programmable devices have been well documented [1,2,3,4,5]. The typical design applications described in the following pages emphasize the conceptual aspects of FPLA usage, in order to focus the reader on the basic roles of FPLAs in logic design, and ease the transfer of these basic ideas to a variety of other practical applications.

Since FPLAs can be readily programmed in the field by the user, they are more economical and easier to use, and should find their way quickly in a wider variety of design situations specifically suited to FPLAs. An estimate of the savings and design advantages obtainable by using FPLAs can be gleaned by examining the recent experience of a Signetics' customer who used FPLAs in the design of parts of an Automatic Landing System for aircrafts. By using a different design approach, he was able to replace 49 IC packages with one FPLA. The tradeoff in both design alternatives is shown in Fig. 30. In the discrete approach, $1 is about what it takes today to place one IC on a PC board. The FPLA cost is based on the projected high volume price in 1976.

QUANTITY	TYPE
12	7400(Quad 2-NAND)
9	7402(Quad 2-NOR)
8	7427(Triple 3-NOR)
5	7442(BCD/DEC Decoder)
2	74175(Quad D-FLOP)
4	7404(Hex Inverter)
2	7430(8-Input NAND)
7	7408(Quad 2-AND)

COMPARISON		
	Random Logic	FPLA
ICs	49	1
Power	3.3 W	0.6 W
Speed	65 NS	50 NS
Cost	$49	$21
Pins	700	28
Space	50 in^2	2 in^2

FIG. 30: The economics of logic replacement with FPLAs. One FPLA replaces 49 ICs at less than half the cost.

LOGIC COMPRESSION

A concise illustration of the logic compression capabilities inherent in FPLAs is provided by the functional truth-table of a squaring matrix, shown in Fig. 31a. The Boolean form of each output function, including Don't Care states (for clarity), is shown in Fig. 22b. Although this table can be directly programmed in a Signetics' FPLA as it is, for sake of illustration all functions have been *individually* minimized by means of Karnaugh maps, and expressed as a sum of product terms (P-terms). In contrast with a minterm, a P-term of (n) variables may contain Don't Care input states, represented explicitly by (X) or implicitly by default. It is easily shown that this representation of the function set is a compressed version of its canonical-P form expressed by the original truth table. Since an FPLA allows internal programming of all three logic states of an input variable, this formal logic compression can be readily translated into hardware.

INPUTS				OUTPUTS							
I3	I2	I1	I0	F7	F6	F5	F4	F3	F2	F1	F0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0	0	1
0	1	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	1	1	0	0	1
0	1	1	0	0	0	1	0	0	1	0	0
0	1	1	1	0	0	1	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	1	0	0	0	1
1	0	1	0	0	1	1	0	0	1	0	0
1	0	1	1	0	1	1	1	1	0	0	1
1	1	0	0	1	0	0	1	0	0	0	0
1	1	0	1	1	0	1	0	1	0	0	1
1	1	1	0	1	1	0	0	0	1	0	0
1	1	1	1	1	1	1	0	0	0	0	1

FIG. 31a. Squaring Matrix for which the output is the square of all 16 input minterms.

$F_0 = XXXI_0$

$F_1 = 0$

$F_2 = XXI_1\bar{I}_0$

$F_3 = XI_2\bar{I}_1I_0 + X\bar{I}_2I_1I_0$

$F_4 = XI_2\bar{I}_1\bar{I}_0 + \bar{I}_3I_2XI_0 + I_3\bar{I}_2XI_0$

$F_5 = I_3\bar{I}_2I_1X + I_3I_2XI_0 + \bar{I}_3I_2I_1X$

$F_6 = I_3\bar{I}_2XX + I_3XI_1X$

$F_7 = I_3I_2XX$

FIG. 31b. Minimized output function set of Squaring Matrix in Boolean form.

To program an FPLA with the minimized function set, we first generate an "ACTIVITY" map of the function set involving the ordered P-terms (Fig. 32a), and then generate from it a Program Table as the direct source of programmable entries (Fig. 32b) in terms of symbols 1, 0, X for clarity.

By comparing the Program Table stored in the FPLA with the original truth table it can be seen that the squaring matrix has been compressed from 16 minterms to 13 P-terms. Truth table compression has been obtained by the "Concurrent", "Selective", and "Multiple" addressing modes characteristic of FPLAs. These can be observed by listing the FPLA output function set while executing an exhaustive logical scan at its inputs (Fig. 33). By viewing each row of the Program Table as an FPLA word selected by the corresponding P-term address, Concurrent addressing is shown by the simultaneous selection of words P_0, P_3, P_6, P_7, P_{10} and P_{11} which occurs with a binary 1011 input to the FPLA (Fig. 34a). Similarly, Multiple addressing is readily apparent by observing that word P_0 is selected by 8 different input combinations, in a manner reminiscent of virtual memory storage (Fig. 34b).

$F_0 = P_0 + / + / + / + / + / + / + / + / + / + / + / + /$

$F_1 = / + / + / + / + / + / + / + / + / + / + / + / + /$

$F_2 = / + P_1 + / + / + / + / + / + / + / + / + / + / + /$

$F_3 = / + / + P_2 + P_3 + / + / + / + / + / + / + / + / +$

$F_4 = / + / + / + / + P_4 + P_5 + P_6 + / + / + / + / + /$

$F_5 = / + / + / + / + / + / + / + P_7 + P_8 + P_9 + / + / +$

$F_6 = / + / + / + / + / + / + / + / + / + / + P_{10} + P_{11} + /$

$F_7 = / + / + / + / + / + / + / + / + / + / + / + / + P_{12}$

FIG. 32a. "ACTIVITY" map of function set of Fig. 22b.
P_n = Function activated by P-term
/ = Function ignores P-term

P_n	INPUTS				OUTPUTS							
	I3	I2	I1	I0	F7	F6	F5	F4	F3	F2	F1	F0
0	X	X	X	1	0	0	0	0	0	0	0	1
1	X	X	1	0	0	0	0	0	0	1	0	0
2	X	1	0	1	0	0	0	0	1	0	0	0
3	X	0	1	1	0	0	0	0	1	0	0	0
4	X	1	0	0	0	0	0	1	0	0	0	0
5	0	1	X	1	0	0	0	1	0	0	0	0
6	1	0	X	1	0	0	0	1	0	0	0	0
7	1	0	1	X	0	0	1	0	0	0	0	0
8	1	1	X	1	0	0	1	0	0	0	0	0
9	0	1	1	X	0	0	1	0	0	0	0	0
10	1	0	X	X	0	1	0	0	0	0	0	0
11	1	X	1	X	0	1	0	0	0	0	0	0
12	1	1	X	X	1	0	0	0	0	0	0	0

FIG. 32b. FPLA Program Table for programming purposes.

M_n	INPUTS				CONCURRENT P-terms
	I3	I2	I1	I0	
0	0	0	0	0	None (Default state term)
1	0	0	0	1	P0
2	0	0	1	0	P1
3	0	0	1	1	P0, P3
4	0	1	0	0	P4
5	0	1	0	1	P0, P2, P5
6	0	1	1	0	P1, P9
7	0	1	1	1	P0, P5, P9
8	1	0	0	0	P10
9	1	0	0	1	P0, P6, P10
10	1	0	1	0	P1, P7, P10, P11
11	1	0	1	1	P0, P3, P6, P7, P10, P11
12	1	1	0	0	P4, P12
13	1	1	0	1	P0, P2, P8, P12
14	1	1	1	0	P1, P11, P12
15	1	1	1	1	P0, P8, P11, P12

FIG. 33: Simultaneously selected P-terms obtained by exercising the FPLA with a minterm generator.

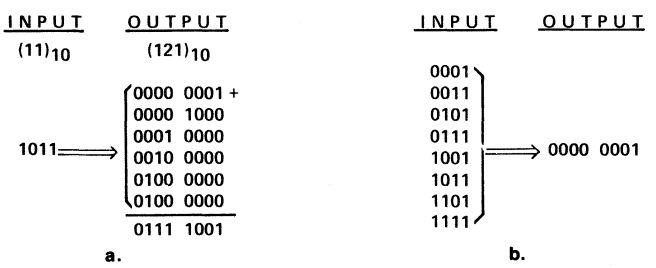


FIG. 34: a) Concurrent address selection by $M_n = (11)_{10}$. Internal OR performed by the FPLA. b) Multiple address selection of P_0.

Selective addressing occurs when Minterm "0" is presented at the FPLA input, but does not activate any of the programmed P-terms 0 thru 12.

At this point it is worth noting that the above implementation is not unique, since the Program Table is not unique. This results from the individual, rather than the *simultaneous* minimization of the output function set. For example:

$$F_4 = \bar{I}_3 I_2 X I_0 + I_3 \bar{I}_2 X I_0 + \bar{I}_3 I_2 \bar{I}_1 X + I_3 I_2 \bar{I}_1 \bar{I}_0$$
$$F_5 = \bar{I}_3 I_2 I_1 X + I_3 \bar{I}_2 I_1 X + I_3 I_2 X I_0$$
$$F_6 = I_3 \bar{I}_2 X X + I_3 I_2 I_1 X$$

is an equivalent form for $F_{4,5,6}$. This choice of expression, although it introduces an additional P-term in F_4, eliminates P_{12} for realizing F_7, since:

$$F_7 = I_3 I_2 X X = \boxed{I_3 I_2 I_1 X + I_3 I_2 X I_0 + I_3 I_2 \bar{I}_1 \bar{I}_0}$$
(contained in $F_{4,5,6}$)

In this case no net reduction in number of P-terms is obtained. However, the method is at the root of the search for a minimum set of P-terms which will implement the desired logic function. Indeed, the reduction of a set of logic functions of several variables to a minimum set of prime implicants (P-terms) requires a simultaneous minimization process for which suitable algorithms have already been developed. *Considerable efforts are currently underway at Signetics for translating such an algorithm in an efficient software program for execution at any of the major timeshare service organizations.*

MEMORY OVERLAYS

The storage and software efficiency of a computer can be improved by overlaying Read/Write memory with (P)ROM memory in blocks of various sizes, including overlay on an individual word basis. A typical memory overlay application is shown in Fig. 35 in which a flag is used to conditionally transfer (P)ROM or R/W data in the MDR. Since (P)ROMs are available in discrete chunks confined in standard IC configurations, a lot of storage can be wasted when the application requires overlay of many blocks of few words each, scattered throughout the address range of R/W main memory. All unused (P)ROM locations servicing a sparsely overlaid sector are forever inhibited access, and are therefore wasted. By using an FPLA instead of (P)ROM, the FPLA address matrix is programmed to recognize only the address of the RAM memory locations to be overlaid. The contents of the overlaid locations (the RAM modifier) are programmed in the FPLA storage matrix. This way, total PROM storage is compressed to the actual words used. Also, because of the large number of inputs to the FPLA, the overlaid locations can be scattered anywhere within a 64K address range. The chip enable feature readily extends this range to any practical size by allowing several FPLAs in parallel to examine a larger number of address inputs.

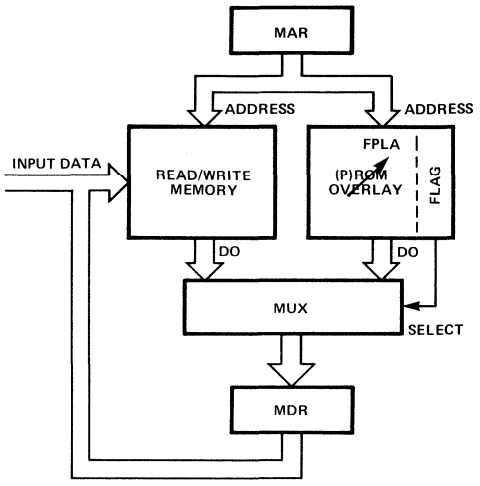


FIG. 35: Typical memory overlay system. (P)ROM word is jammed in MDR when address "present" flag bit is true.

CORE MEMORY PATCH

The use of partially functional random access memory devices is a well known technique employed by manufacturers of Add-On and other large memory systems to reduce overall memory cost per bit. This technique now can be extended to core memory systems by means of an FPLA. Modern core planes are available in many sizes, up to 16K X 18 or 32K X 9. A 64K X 9 memory would require two planes, each containing about 300K cores, in which it is not unusual to find as many as 100 broken or improperly tested cores. Currently, cores are replaced by a hand "restringing" operation, at a cost of about $2/core. A better alternative to core replacement would be a dynamic repair routine, in which memory addresses containing bad bits are patched by an auxiliary memory. However, since bad cores can be scattered anywhere in the plane, this approach would in general be not cost effective because a large auxiliary memory is required to cover an address field equal in size to the original memory. But, by means of the address selectivity of an

FPLA, this constraint is removed. The FPLA renders this technique economically feasible by providing an address "locator" function by virtue of its programmable address characteristics. The core memory addresses containing bad bits are mapped in the "AND" matrix of an FPLA, whose output "OR" matrix is programmed in turn with sequential address pointers to a small auxiliary RAM containing correct data. This scheme is shown in Fig. 36. A 16 input FPLA is used as an address map, and a 64 X 9 RAM as auxiliary memory, chosen to simplify control and to allow several bad core bits/word. The 48 P-terms of the FPLA allow dynamic repair of 48 core memory addresses scattered anywhere in core. Correct data stored in the 82S09 is addressed by 6 FPLA outputs programmed as a binary table. Memory select control is provided by the F_6 output from the FPLA to jam the contents of auxiliary memory in the MDR when a faulty core location is addressed, and to enable writing in auxiliary memory only in the patched locations.

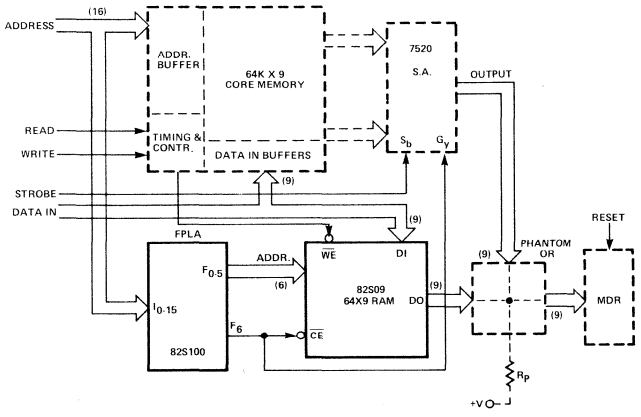


FIG. 36: Core memory "Patch" with FPLA requires only two ICs.

The core memory system normally contains sockets and connections for both FPLA and auxiliary RAM. The sockets are filled only with partially functional planes. The FPLA Input Table is programmed immediately following final test with the addresses of core failures.

This technique could also be applied, with suitable modifications, to memory systems implemented with partially functional Bipolar or MOS memory devices. It could also be extended to patch modifications in ROM memory systems, or utilize spare locations in PROM memory systems to avoid replacing several packages because of random or repeated changes.

SUBROUTINE ADDRESS MAP AND BRANCH LOGIC

In the design of microprogrammed computers considerable design flexibility is gained by complete freedom in allocating microprogram subroutines throughout microcontrol store, and by the utilization of variable formats in the Instruction Register op-code field.

To satisfy these requirements in an economical manner, an efficient means of address translation is mandatory. FPLAs are ideally suited for this application as shown in a typical system in Fig. 37. The first FPLA translates the current op-code from a 16-bit Instruction Register into 48 subroutine-start addresses in microcontrol store. Variable op-code formats are easily handled by judicious programming of Don't Care states in the FPLA Input Table. The second FPLA is used to generate branch conditions based on the current microinstruction, as well as jump and status conditions in the machine. In particular, the utilization of tri-state FPLAs (82S100) saves a multiplexer in the address path of the ROM Address Register, and their 50 ns access time minimizes overhead time in the instruction execution loop.

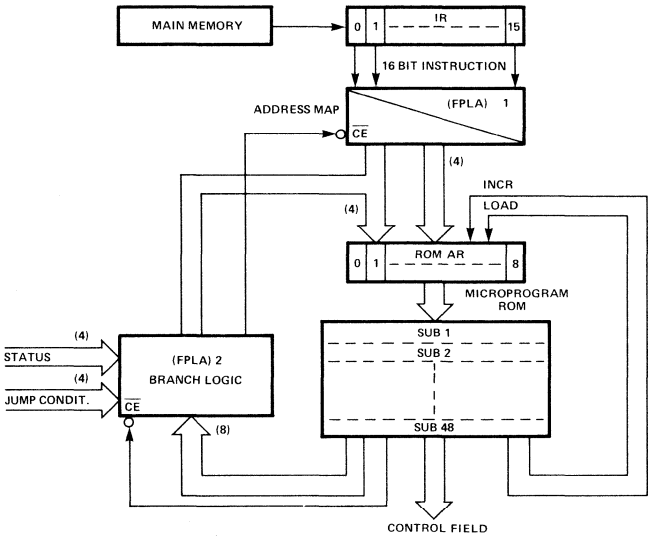


FIG 37: Subroutine Address Map and Branch Logic with tri-state FPLAs.

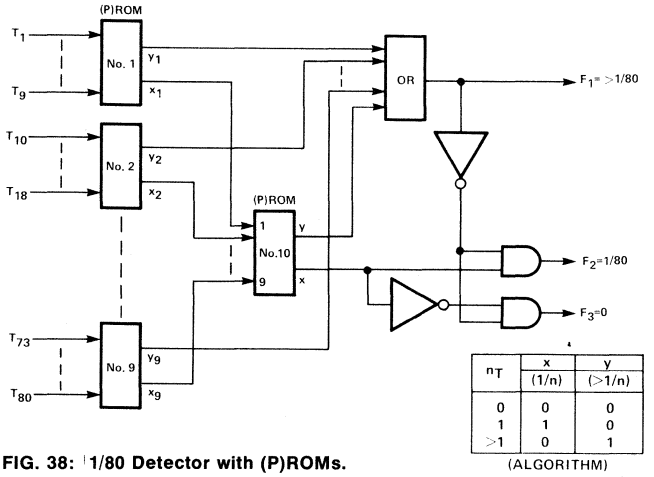
FAULT MONITOR NETWORKS

The dramatic savings in hardware which can be obtained by using FPLAs to manipulate a large number of logic variables is again apparent when building 1/N detectors, as a special case of m/N decoder networks. These are useful in a

variety of applications in computers, data communications, and fault monitor systems. For example, in a data multiplexing system it is not uncommon to find 80 or more channels time-division multiplexed onto a single transmission line. If a fault occurs in the multiplexer-control network, multiple or no connections on the line give rise to invalid transmission. These type of faults can be readily detected by using a 1/80 detector to monitor the normal selection status of only one multiplexer channel at a time. A 1/N detector could be implemented by using logic gates. Excluding inversion and ORing of partial results, the number of gates required is given by the number of logic states to be detected. For 80 status monitor terminals (one for each data channel):

$$\#\ \text{of Gates}_{(AND)} = N!/(N-1)! = 80$$

This approach, when complicated by the fact that each gate also requires 80 inputs, becomes quickly impractical. A more practical alternative involves partitioning the number of terminals in equal subsets which are applied to PROMs whose truthtables yield outputs x = 1/n and y > 1/n [6]. Each PROM is used as a basic building block in a cascaded array, to implement a general algorithm for detecting 0,1 or more True states (logic "1" = channel selected) of n variables. This is shown in Fig. 38 for N = 80, using a 512 X 8 PROM organization. It requires 10 PROMs, plus some gating circuitry for status indication.



nT	x (1/n)	y (>1/n)
0	0	0
1	1	0
>1	0	1

(ALGORITHM)

FIG. 38: 1/80 Detector with (P)ROMs.

However, with FPLAs a more efficient solution is possible as shown in Fig. 39. It requires only 6 devices.

Since each FPLA can examine 16 terminals, five are sufficient to service all 80 terminals. Each FPLA utilizes 17 P-Terms to detect the presence of zero, 1/16, or >1/16 via outputs (x) and (y) as defined in the Program Table of Fig. 40. An additional FPLA is necessary to examine a total of 10 partial x and y results from the first level devices, and to give final indication of the number of selected terminals. The Program Table of the last FPLA is contained in Fig. 40b.

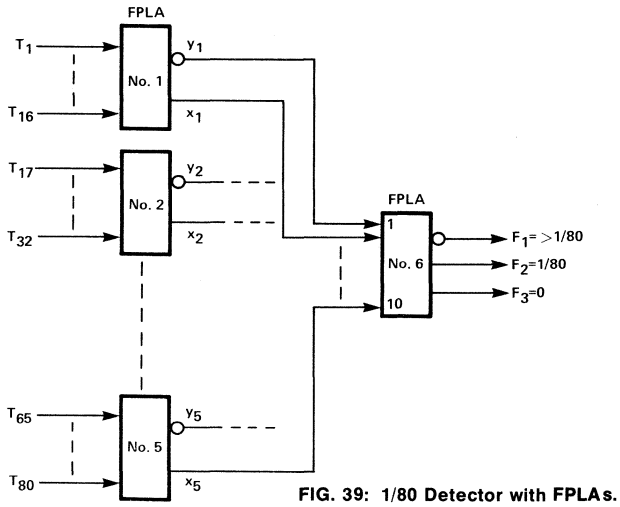


FIG. 39: 1/80 Detector with FPLAs.

INPUT																	OUTPUT	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		F_1 x	F_2 y
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1		1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0		1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0		1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0		1	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0		1	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0		1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0		1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0		1	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0		1	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0		1	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0		1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0		1	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0

a. Program Table stored in FPLAs 1 through 5.

INPUT										OUTPUT		
–x=1/N–					–y>1/N–					>1/N	=1/N	0
10	9	8	7	6	5	4	3	2	1	F_1^*	F_2	F_3
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	1	0

b. Program Table stored in FPLA 6.

FIG. 40: Program Tables of FPLAs used in 1/80 Detector. Starred (*) outputs are programmed active-LOW.

FAST MULTIBIT SHIFTER

Computer performance can be greatly increased by incorporating hardware capabilities

to execute fast multibit shifts. This results in a considerable reduction in execution time for algorithms that involve a large number of arithmetic, logic, or circular shifts, such as divide/multiply, floating point operations, etc.

A multibit shifter implemented with two FPLAs is shown in Fig. 41a. It provides Arithmetic or Logic shift of an 8-bit byte either Left or Right up to 7 places within one clock cycle. Two FPLAs are necessary, for a total of 71 P-terms.

The Program Table to be stored in the devices is derived from the set of output equations tabulated in Fig. 41b. The table entries represent output functions F_0 through F_7, which are True (1) at coordinate points $(I_m \bullet S_{LN})$ or $(I_m \bullet S_{RN})$. These are respectively the *ordered input data bits*, and the number of right or left shifts. A further subdivision of the table is given by I_{12} for Arithmetic or Logic shifts.

For example, for a Logic Shift of the input data, the P-terms which must be programmed in the FPLAs for say output bit 5 are:

$$F_5 = I_5S_{RO} + I_6S_{R1} + I_7S_{R2} + I_5S_{L0} + I_4S_{L1} + I_3S_{L2} + I_2S_{L3} + I_1S_{L4} + I_0S_{L5}$$

The P-terms in the equation are in turn converted in Program Table format, typically as follows:

	LOG/AR	L/R	S_N			I_m							F*
	I_{12}	I_{11}	I_{10}	I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	5
I_5S_{R0} ---	X	1	0	0	0	X	X	1	X	X	X	X	X	0
I_5S_{L0} ---	X	0	0	0	0	X	X	1	X	X	X	X	X	0

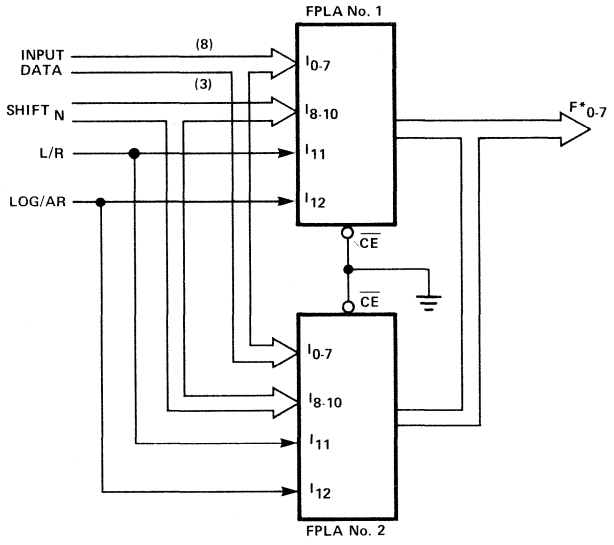
The wire-AND of the two FPLAs requires F_0 through F_7 to be programmed active-LOW (each designated as F*). Therefore, the Shifter outputs the complement of the shifted input word, which must be in turn complemented if this inversion cannot be buried in the system. Both P-terms involving S_{RO} and S_{LO} can be combined as I_5S_{X0}, denoting a Don't Care for right or left shift. All 16 such terms appearing in F_0 through F_7 can be combined into 8 P-terms. It can be readily shown that all 64 P-terms implicit in the upper half of the table are needed for both Arithmetic and Logic shift, and require I_{12} = X (Don't Care) as conditional input. For the Arithmetic shift selected by I_{12} = "1", seven additional P-terms are necessary to ensure propagation of the sign bit to the right in a right shift, and retention of the sign bit in F_7 during a left shift. These additional P-terms are implicit in the equations obtained from the bottom half of the table. For example, for an Arithmetic shift output F_6 is given by:

$$F_6 = I_6S_{LO} + I_5S_{L1} + I_4S_{L2} + I_3S_{L3} + I_2S_{L4} + I_1S_{L5} + I_0S_{L6} + I_6S_{RO} + I_7(S_{R1} + S_{R2} + S_{R3} + S_{R4} + S_{R5} + S_{R6} + S_{R7})$$

This application can be readily expanded to detect overflow, or to execute circular shifts.

The capability for circular shifts is obtained by using and additional FPLA, for a total of 124 P-terms.

Note that here we can obtain a shift of 7 bit positions in 35 ns, typical.



a. SHIFTER shifts Left/Right, Arithmetic or Logic up to 7 places in 35 ns.

I_m	LEFT SHIFT S_{LN}: I_{11} = 0 0 1 2 3 4 5 6 7	RIGHT SHIFT S_{RN}: I_{11} = 1 0 1 2 3 4 5 6 7	
0	0 1 2 3 4 5 6 7	0	
1	1 2 3 4 5 6 7	1 0	
2	2 3 4 5 6 7	2 1 0	
3	3 4 5 6 7	3 2 1 0	LOGIC/ARITHMETIC
4	4 5 6 7	4 3 2 1 0	I_{12} = X
5	5 6 7	5 4 3 2 1 0	
6	6 7	6 5 4 3 2 1 0	
7	7	7 6 5 4 3 2 1 0	
7	7 7 7 7 7 7 7 7	7 7 6 5 4 3 2 1	
7		7 6 5 4 3 2	
7		7 6 5 4 3	ARITHMETIC
7		7 6 5 4	I_{12} = X
7		7 6 5	
7		7 6	

b. Logic equation set of SHIFTER to be programmed in FPLAs.

FIG. 41: Fast multibit SHIFTER with FPLAs.

PRIORITY RESOLVER AND LATCH

FPLAs can perform the dual function of detecting and latching tristate-bus data, on a priority basis. By using only 24 P-terms in a single FPLA, three priority functions can be selected via inputs $S_{0,1,2}$ as shown in Fig. 42.

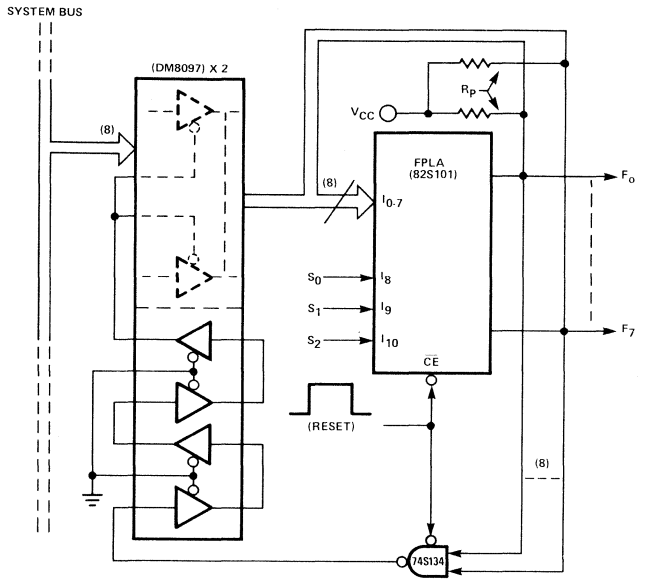


FIG 42: Priority Resolver and Latch with FPLA. The FPLA latched state must be reset prior to sampling new data.

INPUTS										OUTPUTS								
S2	S1	S0	17	16	15	14	13	12	11	F7	F6	F5	F4	F3	F2	F1	F0	
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1st of 8 Priority
			1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	
			1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	
			1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	
			1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	
			1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	
			1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	
0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	
0	1	0	X	X	X	X	X	X	X	0	1	1	1	1	1	1	0	1 of 8 Priority (Ascending rank)
			X	X	X	X	X	X	0	1	1	1	1	1	1	0	1	
			X	X	X	X	X	0	1	1	1	1	1	1	0	1	1	
			X	X	X	X	0	1	1	1	1	1	1	0	1	1	1	
			X	X	X	0	1	1	1	1	1	1	0	1	1	1	1	
			X	X	0	1	1	1	1	1	1	0	1	1	1	1	1	
			X	0	1	1	1	1	1	1	0	1	1	1	1	1	1	
0	1	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	
1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1 of 8 Priority (Descending rank)
			1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	
			1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	
			1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	
			1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	
			1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	
			1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	
1	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	

FIG. 43: FPLA Program Table for Priority Resolver. F_{0-7} must be programmed active-LOW. Unused inputs are programmed as Don't Care.

The Reset pulse clears any previously latched priority, and must be at least 30 ns wide to compensate for FPLA delay. Sampling of the system bus begins with the trailing edge of Reset, and ends about 50 ns after the detection of an input request (H→L transition). This delay is provided by the feedback chain of spare gates in the DM8097 buffers, and is required to allow the FPLA to latch the incoming request before releasing the bus. It is also the circuit's resolving time of nearly simultaneous requests. The FPLA Program Table is shown in Fig. 43. The function selected by S_0 provides a 1 of 8 priority in *time* by latching the first of eight signals occurring on the bus, and is useful in many polling applications in which a 50 ns resolution is adequate. The functions selected by S_1 and S_2 provide 1 of 8 complementary priorities in *space* by latching the highest ranked signal on the bus.

Both functions are particularly useful in asynchronous multiport systems for transferring control of the main system bus. The concept illustrated is readily expanded with additional output circuitry to monitor up to 16 inputs with any assigned rank, or to implement a clocked revolving priority of N signals.

The primary advantage provided by the FPLA is that the reassignment of priority rank is facilitated by combining the external selection with FPLA programmability, without resorting to system *wire changes*.

"VECTORED" PRIORITY INTERRUPT SYSTEM

Since FPLAs can store input Don't Care states directly, a simple ranked priority among N signals can be resolved with just N P-terms. With 16 inpus available, in most applications of this type most FPLA P-terms would remain unused. In such cases, a more efficient utilization is possible by time-sharing the FPLA to perform separate functions.

This technique can be applied to the design of a "Vectored" priority interrupt system for the Signetics 2650 Microprocessor.

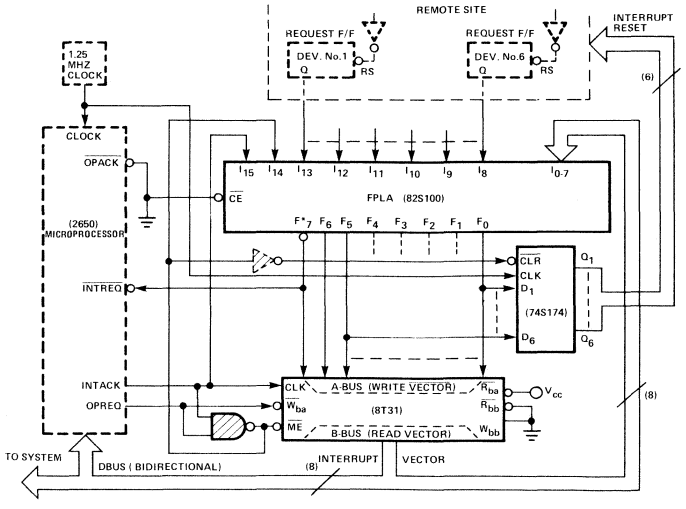


FIG. 44: "Vectored" Priority Interrupt system for the 2650 μP requires 3 ICs, and 2 spare gates. Starred (*) FPLA outputs are programmed active-LOW.

The circuit in Fig. 44 is all that is required to service six I/0 Devices via the conventional, single level, address vectoring interrupt mechanism of the 2650.

When one, or more devices request service the CPU receives an $\overline{\text{INTREQ}}$ signal on its single interrupt pin. Program control is transferred to any of 128 possible memory locations as determined by an 8-bit vector supplied by the FPLA on the CPU data bus, in accordance with a preprogrammed priority. Since memory locations are expressed in 2's complement, the vector can point anywhere within −63 to +64 bytes of page zero, byte zero of memory. Also, both Direct or Relative Indirect addressing modes can be specified by the vector (bit D_7 = 0/1), hence program execution can be directed anywhere within addressable memory.

During the execution of the asynchronous CPU handshake the FPLA supplies at various times three distinct functions:

1. *Interrupt Request to the CPU, triggered by one or more service requests from Devices 1 through 6.*
2. *Priority Resolution of simultaneous requests by placing on the CPU data bus the vector of the highest ranked interrupting Device.*
3. *Issue a Request Reset signal to 1 of 6 selected Devices to acknowledge servicing its interrupt.*

The six I/O Devices have been assigned the arbitrary vectors tabulated in Fig. 45.

The Program Table of Fig. 46 shows the FPLA P-terms necessary to execute the above functions, with inputs $I_{15,14}$ used as function selectors under CPU control. Note that it was necessary to program the FPLA outputs with the *complement* of the vector, to compensate for the inversion with the 8T31.

The timing diagram of the CPU handshake and FPLA response is shown in Fig. 47.

In order to be immediately serviced, an $\overline{\text{INTREQ}}$ must be received by the CPU before the last cycle of the current instruction. When this occurs, the CPU finishes executing the current instruction, and in its last cycle, rather than fetching the

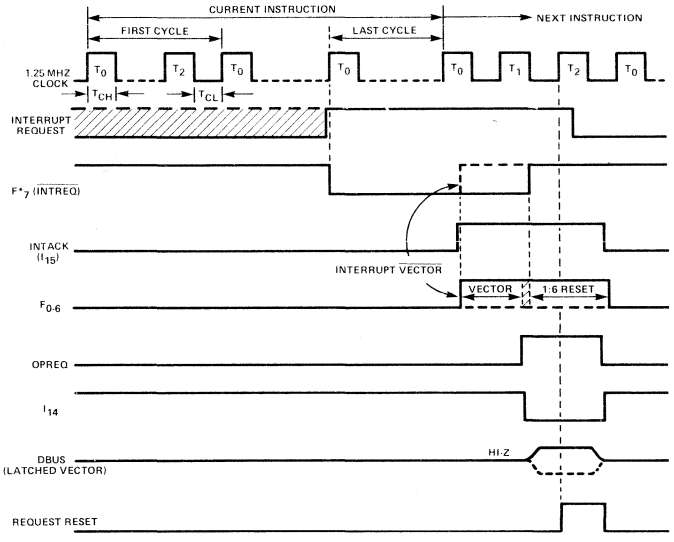


FIG. 47: Timing diagram of I/O service request interrupting program execution.

2's COMPLEMENT VECTOR	μP DBUS							
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
	D/I	+/−	32	16	8	4	2	1
+25 Direct	0	0	0	1	1	0	0	1
−39 "	0	1	0	1	1	0	0	1
+25 Indirect	1	0	0	1	1	0	0	1
−39 "	1	1	0	1	1	0	0	1
+55 Direct	0	0	1	1	0	1	1	1
+38 "	0	0	1	0	0	1	1	0

FIG. 45: Vectors pointing to memory locations containing instructions for servicing interrupting Devices.

FUNCTION	FUNCTION SELECTOR		PRIORITY/REQUEST GENERATOR						RESET GENERATOR								FPLA OUTPUT							
	I_{15}	I_{14}	I_{13}	I_{12}	I_{11}	I_{10}	I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0
INTERRUPT REQUEST TO μP	0	1	X	X	X	X	X	1	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
	0	1	X	X	X	X	1	X	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
	0	1	X	X	X	1	X	X	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
	0	1	X	X	1	X	X	X	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
	0	1	X	1	X	X	X	X	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	0	1	1	1	1	1	1	1
PRIORITY RESOLVER	1	1	X	X	X	X	X	1	X	X	X	X	X	X	X	X	1	1	1	0	0	1	1	0
	1	1	X	X	X	X	1	0	X	X	X	X	X	X	X	X	1	0	1	0	0	1	1	0
	1	1	X	X	X	1	0	0	X	X	X	X	X	X	X	X	0	1	1	0	0	1	1	0
	1	1	X	X	1	0	0	0	X	X	X	X	X	X	X	X	0	0	1	0	0	1	1	0
	1	1	X	1	0	0	0	0	X	X	X	X	X	X	X	X	1	1	0	0	1	0	0	0
	1	1	1	0	0	0	0	0	X	X	X	X	X	X	X	X	1	1	0	1	1	0	0	1
RESET REQUEST	1	0	X	X	X	X	X	X	0	0	0	1	1	0	0	1	1	0	1	0	0	0	0	0
	1	0	X	X	X	X	X	X	0	1	0	1	1	0	0	1	1	0	0	1	0	0	0	0
	1	0	X	X	X	X	X	X	1	0	0	1	1	0	0	1	1	0	0	0	1	0	0	0
	1	0	X	X	X	X	X	X	1	1	0	1	1	0	0	1	1	0	0	0	0	1	0	0
	1	0	X	X	X	X	X	X	0	0	1	1	0	1	1	1	1	0	0	0	0	0	1	0
	1	0	X	X	X	X	X	X	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	1

FIG. 46: FPLA Program Table. Only 18 P-terms are necessary to perform three time-shared functions.

next sequential instruction, it 1) sets the Interrupt Inhibit bit in the Program Status Word to inhibit further interrupts, and 2) inserts the first byte of the "Zero Branch-to-Subroutine, Relative" instruction in the IR.

In the next cycle the CPU gets ready to access the data bus to fetch the interrupt vector as the second byte of the ZBSR instruction, hence it generates the INTACK signal which is used to jam on the FPLA outputs the complement of the vector associated with the highest ranked Device requesting service. The vector is latched, and placed on the CPU data bus following the leading edge of OPACK, after which the 8T31 A-Bus is locked out. The CPU reads the DBUS on the trailing edge of T_2, and begins executing the interrupt routine. When the routine is completed, a return instruction clears the Interrupt Inhibit bit and links execution back to the interrupted program. Meanwhile, in order to communicate with the Device being serviced by the interrupt routine, it is necessary to flag the Device that its request has been acknowledged. This is done by issuing to the Device a Reset signal generated by the FPLA. The latched vector is fed back in the FPLA and decoded to issue a unique Reset signal, which is in turn latched in the 74S174 on the leading edge of T_2 clock phase.

Several variants of this basic approach have been investigated. In particular, in a case where one needs to service 12 I/O Devices and can tolerate to point the vector within a narrower memory address range, it is possible to substitute the 8T31 with 4 tristate buffers, and use the FPLA in a wrap-around connection to latch the vector. The generation of the $\overline{\text{INTREQ}}$ and Reset signals must however be reallocated outside the FPLA.

REFERENCES

1. D. Mrazek, and M. Morris, "How to design with Programmable Logic Arrays", National Semiconductor Corp., app. note AN-89, 1973.
2. G. Reyling, "PLAs enhance digital processor speed and cut component count", Electronics, August 1974.
3. J. Maggiore, "PLA—A universal logic element", Electronic Products Magazine, April 1974.
4. W. N. Carr, and J. P. Mize, "MOS/LSI Design and Application", pp. 229–258, T.I. Electronics Series, McGraw-Hill Co., 1972.
5. J. C. Logue et al, "Hardware implementation of a small system in PLAs", IBM J. Res. Develop., March 1975.
6. A. W. Kobylar et al, "ROMs cut cost, response time of m/N detectors", Electronics, February 1973.

RELIABILITY OF Ni-Cr LINK FUSING SYSTEM

One dominant fact with regards to reliability has emerged from the work done at Signetics and elsewhere. This is that reliability is a function of not only the nichrome link but also of the circuit design (Ref. 2) i.e., both the nichrome process and the circuit conditions must be considered during fusing and subsequent steady state operating conditions.

The requirements for a reliable fusible link system can be considered under the following categories:

- Process Control
- Reliability of Unfused Links
- Reliability of Fused Links

PROCESS CONTROLS

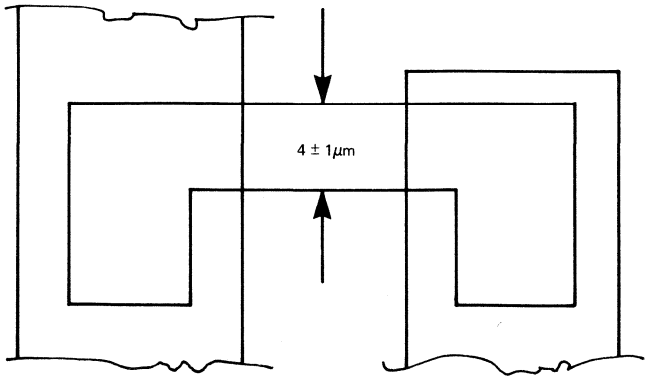
Nichrome is used for the fusible link because of the long experience with it as a thin film resistor material. It can be incorporated into the product without any compromises having to be made to the other processing steps.

The nichrome film is evaporated under carefully controlled conditions with the Nickel: Chrome ratio being monitored on a continual basis by an independent Quality Control group.

Sheet resistance is used as a thickness monitor, the two being monotonically related (Ref. 3).

The nominal film thickness of approximately 350 Angstroms ensures that the film is a continuous layer having aging characteristics superior to the "island" type structure seen with thinner films and is confirmed by the measured T.C.R. of 80 p.p.m. /°C (Ref. 3, 4, 5).

Ease of fuse control, both geometry and resistance, is greatly improved by use of a "bar" as opposed to "notch" geometry used previously because the alignment of fuse to aluminum leads is much less critical.



As a check of fuse characteristics all wafers are individually measured for fuse width and resistance before being passed to device electrical sort.

The fuses are covered with approximately 1μm of deposited passivation glass ensuring no redeposition of evaporated nichrome in the package following fusing.

RELIABILITY OF UNBLOWN LINKS

The main concern with regards to unblown links is the possibility of an unblown link opening under normal circuit operating conditions.

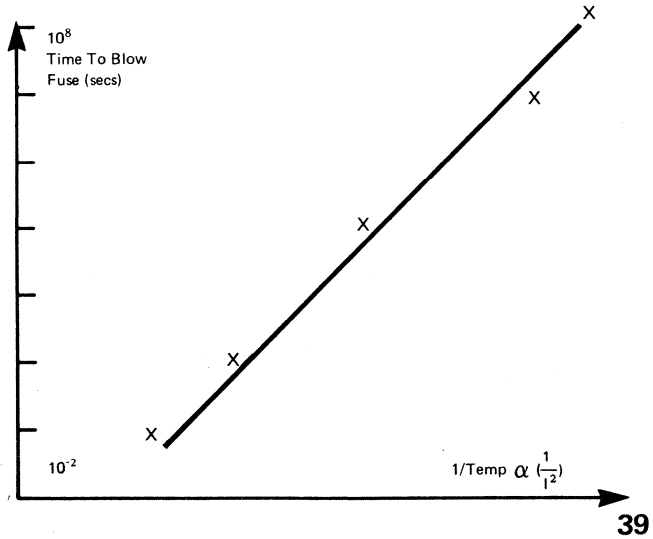
All Signetics PROMs are designed such that the link fusing current to operating current ratio is a minimum of 20:1 which ensures low current stress under operating conditions. With 350 A° films, the normal operating current levels of 1—2mA are well within safe limits.

To verify this, fuse links have been stressed, at Signetics, at 5 times normal current (7.5mA = 5 X 10^5 Amps / cm^2) at 200°C chip temperature with no failures recorded and a resistance change of less than 2% for a total of 2 million fuse hours (1080 fuses for 2016 hours).

In one study of the behavior of these fuses (Ref. 6), an Arrhenius type relationship was established between the time to blow (tens of milliseconds to thousands of hours) and the calculated fuse temperature (proportional to square of current).

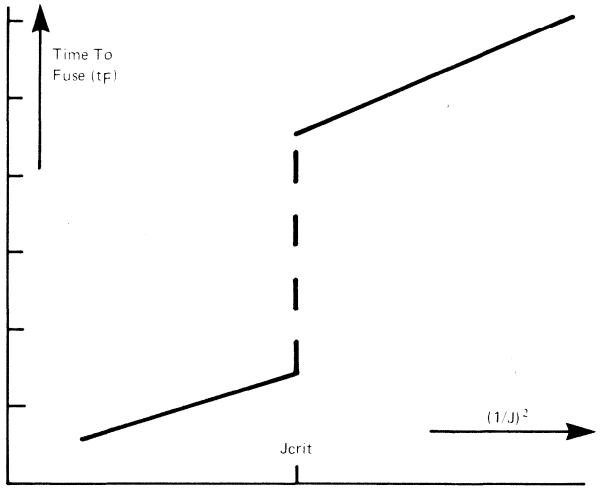
An activation energy of 1.6eV was obtained leading to an extrapolated 50% lifetime of 3.3 X 10^5 years at 110°C.

These results have been duplicated at Signetics with an E_A = 1.65eV.



RELIABILITY OF BLOWN LINKS

As fusing current is increased beyond the levels considered above, a discontinuity in fusing times is observed (see diagram).



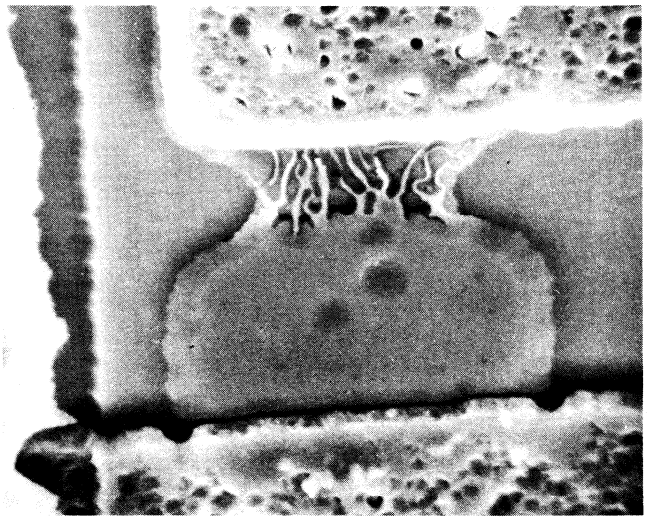


FIG. 1: Fuse blown under limited current condition resulting in characteristic "whisker" formation.

This discontinuity is seen as a sudden decrease in fusing time (from milliseconds to microseconds) for a relatively small change in current. The current density through the fuse at this point (Jcrit) is approximately 2 X 10^7 Amps / cm^2.

The fusing behaviour of the fuses above and below Jcrit is very different and **so is the tendency towards fuse regrowth.**

1) J $\leq$2 X 10^7 A / cm^2, t_F >tens of milliseconds:

Here, the fusing is thought to be a local oxidation phenomenon.

After fusing, the nichrome is seen to have small separation gaps with a distinctive "whisker" shape characteristic (Fig. 1).

Fuses blown under these conditions have been observed, here and elsewhere, to exhibit regrowth under applied bias.

(The electric field across these small gaps is extremely high.)

Several investigators have experimentally derived critical fusing times (t_{crit}) above which this phenomenon can occur. Published figures of t_{crit} range from ~2 msecs (Ref. 6) to ~20 msecs (Ref. 8, 9).

The conditions that lead to regrowth are limited availability or slow rise time of the programming current (Ref. 6, 8). These limitations can be due to external programming conditions or to **poor circuit design internally.**

Fuse regrowth behaviour under these conditions can be summarized as follows:

Healing is accelerated by operational testing at maximum V_{CC} and temperature.

Heals show an early mortality type behaviour with virtually all failures occuring in the first several hours.

Resistance of fuses after regrowth is typically around 1000 ohms.

Healing is very dependent on the voltage seen across the blown fuse under normal operation. The lower the electric field the less the chance or regrowth.

2) J >> 2 X 10^7 A / cm^2, t_F << 1 millisecond:

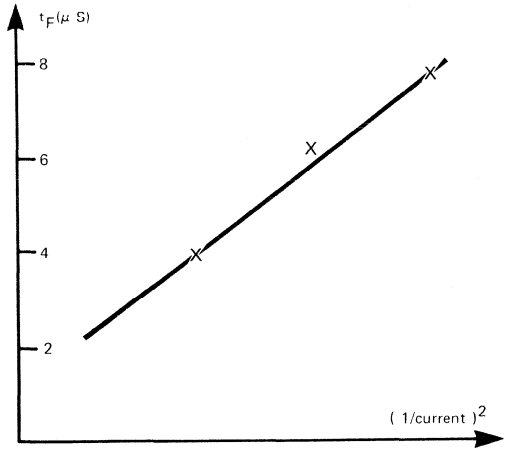
The fusing behaviour in this region is believed to be a very fast melting with surface tension pulling back material from the center of the fuse. (A quick calculation of energy considerations confirm this.)

This condition is presumed to be the time region where heat loss to surroundings is insignificant (adiabatic), the blow time being less than the thermal time constant of the system. The time to fuse under these conditions should be inversely proportional to the ratio of energy supply (Ref. 8) i.e., $t_F \propto (1/\text{current})^2$.

This was experimentally determined to be the case.

Fuses programmed to these conditions exhibit

a distinctively clean separation of the link (Fig. 2). The potential for regrowth of these fuses under the subsequest normal circuit bias (~2 volts) is essentially zero and none have indeed been seen to occur in controlled experiments (Ref. 6).



Signetics has redesigned its complete family of TTL PROMs with special emphasis on increasing current density during fusing.

This family of circuits now uses Schottky diode fusing matrix, taking full advantage of our dual level metal process to almost eliminate diode series resistance and to give identical device behaviour throughout the matrix i.e., **no worst case bits at the end of words lines.**

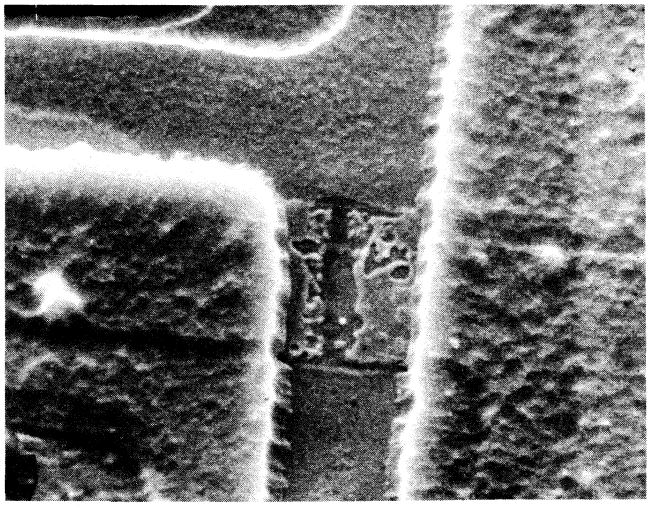
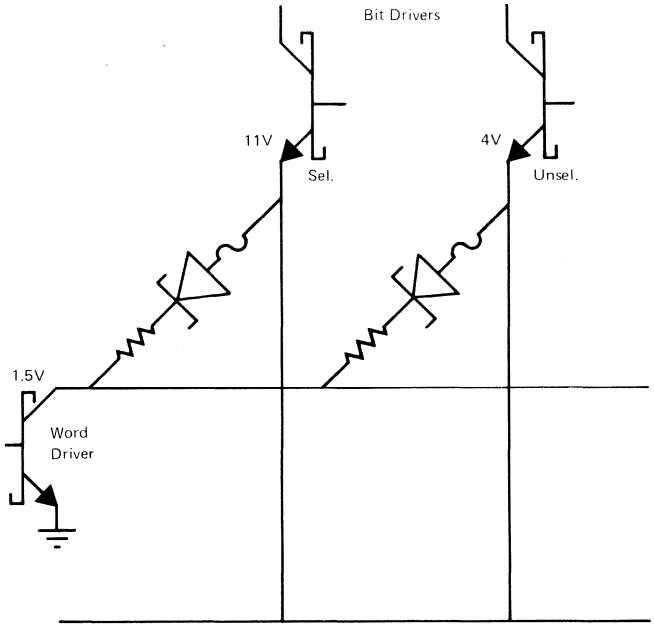


FIG. 2: Bar fuse (82S126) blown to standard programming conditions (note cleanly blown gap).

The diode approach also allows higher reverse voltages to be safely used as compared to the conventional multi-emitter structure with its inherent emitter—emitter breakdown limitation (~6 volts.) The higher breakdown ensures much lower risk of reverse biased **unselected** fuses seeing large leakage currents during programming.

The resultant voltage across the fuse is approximately 7 volts.

For a minimum width fuse (3μm) giving a maximum resistance of 150 ohms (max. allowed).



current density is:

$$J = \frac{7}{150 \times 3 \times 10^{-4} \times 350 \times 10^{-8}}$$

$$J \sim 4.4 \times 10^{7} \text{ Amps / cm}^2$$

For a maximum width fuse (5μm)giving a resistance of say 60 ohms (typical value), the current density becomes:

$$J = \frac{7}{60 \times 5 \times 10^{-4} \times 350 \times 10^{-8}}$$

$$J \sim 4 \times 10^{7} \text{ Amps / cm}^2$$

All Signetics PROM circuits are now designed to supply a minimum current density of 4×10^7 Amps / cm^2 to the fuses during the programming. Driving transistors are designed to easily carry these currents.

Because of these design techniques, Signetics PROM fuses program extremely fast and cleanly.

Using a fast storage scope (Tektronix 7623) to monitor current supplied to selected output pin during programming, it has been determined that **typical fusing times range from 0.3 to 2 microseconds (Fig. 3).**

Signetics recommends a maximum fusing current pulse width of 2 milliseconds to screen out slow fusing links.

Life tests of these redesigned parts has been started and millions of fuse hours have been successfully accumulated at the time of writing (see Attachment).

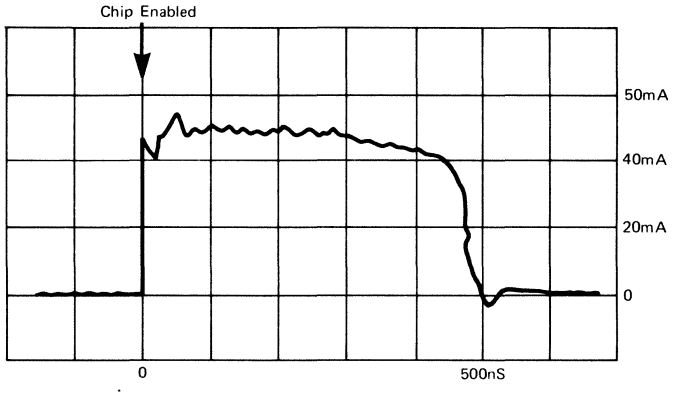


FIG. 3: Fusing current monitored at output pin of 82S129. Blow times typically measured at 0.3 to 1.5 microsecs.

PROGRAMMING YIELD

FACTORY TESTING

In order to guarantee a high programming yield the following tests are carried out at the factory.

a) Each wafer is individually measured for both fuse width and resistance before being passed on to wafer electrical sort.

b) At wafer sort, an extra row and column of the fuse matrix is programmed to standard specification (2 msec pulse width) to ensure both programmability and address uniqueness, etc. Die are also rejected for multiple (unselected) blows in the extra row and column as well as in the matrix proper.

PROGRAMMER REQUIREMENTS

The necessary conditions for good, reliable fusing puts quite a demand on the programming current source.

In particular, the source needs excellent transient response, since some devices will, during the normal course of fusing, require more than 150ma to charge up internal capacitances to their proper levels. Although this never lasts more than a few hundred nanoseconds, fusing performance can be seriously degraded if the fusing supply does not return to the specified voltage quickly.

Because of the intimate relationship between the programming equipment and the device to be programmed, the PROM and the programmer manufacturer must work very closely together to ensure a reliable programming system. For example, an adjustment of one machine to meet the above transient requirements resulted in increase of programming yield from a low of 52% up to 96%.

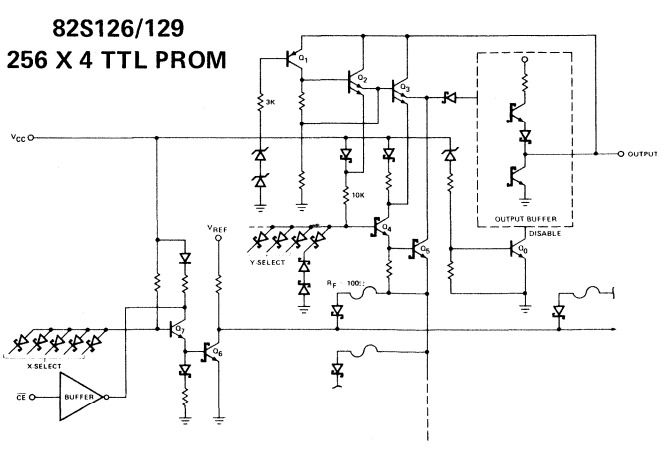
PROGRAMMING EQUIPMENT LIST

PROM	DATA I/O	CURTIS		ADAR SPECTRUM		PRO LOG
	PROG. MODULE	MANUAL	DUPLICATOR	MANUAL[1]	PROG. MODULE[2]	PROGRAM MODULE
82S23 82S123	909-1051-7A	PR-1369A	PR-2300S	434-555	434-551	PM9010
82S27	909-1055-8	PR-27	PR-2700S			
82S126 82S129 82S130 82S131	909-1055-10	PR-1369A	PR-2600SA	434-572	434-571	PM9008
82S114	909-1145-1	PR-145	PR-1145	434-580[4]	434-574	PM9021[3]
82S115	909-1145-2					
10139	909-1051-2	PR-10139		434-582[4]	434-581[4]	
10149	909-1144-1	–	–	434-584[4]	434-583[4]	

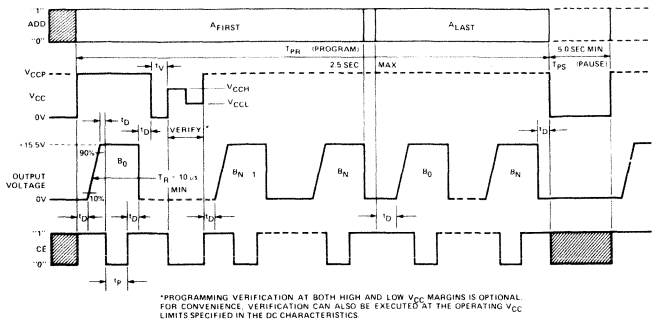
(1) FOR MODEL 300 ONLY
(2) FOR MODEL 550 ONLY
(3) PRELIMINARY, AWAITING SIGNETICS QUALIFICATION
(4) IN DEVELOPMENT

Signetics has subsequently adopted a policy of testing and giving approval of all commerically available programmers at each of the various manufacturers.

82S126/129
256 X 4 TTL PROM



82S126/129
256 X 4 TTL PROM
TYPICAL FUSING PROCEDURE



PROGRAMMING PROCEDURE

1. Disable Chip
2. Select Address
3. Raise V_{CC} to 8.75 ± ±.25, 300mA min.
4. Apply 17 ±1V to appropriate output with 200mA limits.
5. Pulse $\overline{CE}$ low (logic "O") for 1—2ms.
6. Remove 17V from output
7. Select new address.

Commercially available programmers for Signetics PROMS are shown above, including the programming procedure for the 82S126/129 with timing diagrams and a representational model of its fusing circuitry.

DEVICE PERFORMANCE

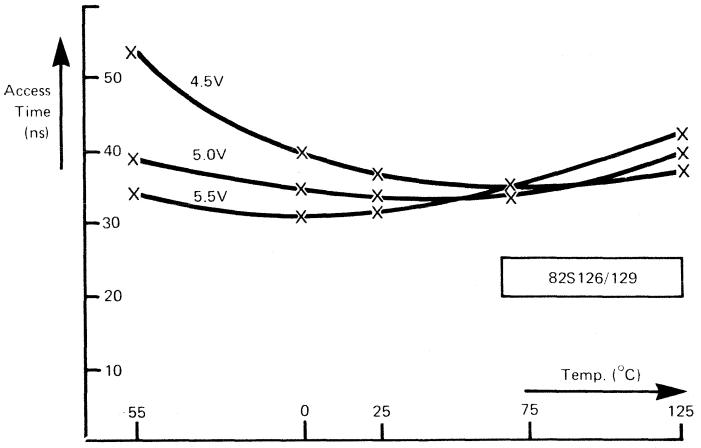
TTL Family	PROM	ROM
32 X 8 Bits	82S23/123	
256 X 4 Bits	82S126/129	82S226/229
512 X 4 Bits	82S130/131	82S230/231
256 X 8 Bits	82S114	82S214
512 X 8 Bits	82S115	82S215

All of these devices are made on Signetics' standard thin epitaxy Schottky process. ROMs and PROMs are pin for pin interchangeable and all use the same Schottky diode matrix, addressing schemes and output circuitry. **They are the highest performance devices available today.**

The use of Signetics' highly reliable dual layer metal process (Fig. 4) as an integral design tool has led to several extremely significant device performance improvements:

a) No emitter diffusion bridges—obtain perfect diode matching throughout circuit leading to excellent thermal stability. It is not necessary therefore to adopt a screening process to select out military temperature range product. The absence of diffusion bridges also reduces the dependency on V_{CC}. The devices are extremely fast as can be seen by the diagram below:

b) Ensures that all diodes in matrix are identical with respect to series resistance. Along with improved fusing this allows expansion of the matrix to any size desired and also gives extremely little speed variations with respect to various truth table patterns.



c) Allows very high current (40—80mA) Schottky diode array by essentially limiting series resistance and enabling an almost perfect "ground" system to be established across the chip (prevents parasitic transistors being turned on).

d) Allows low current densities (and low IR drops) for the main power bus system.

PNP inputs are used giving the usual advantage of a dramatic reduction in input buffering requirements compared to TTL.

Outputs are normally low thus enabling direct V_{OL} measurements.

ECL FAMILY

32 X 8	10139
256 X 4	10149

These devices are compatible with the ECL 10,000 Series circuits and are extremely fast. The use of dual layer metal has a further advantage in ECL in severely reducing the RC time constants associated with the use of diffusion bridges.

A multiple transistor matrix is used, the use of driving current sources preventing excessive currents flowing down reverse biased unselected bases.

All major signals are run on second level metal to reduce capacitances and provide a better 50 ohm environment.

R_{bb} is reduced significantly by direct metal connection to the base of each transistor in the matrix.

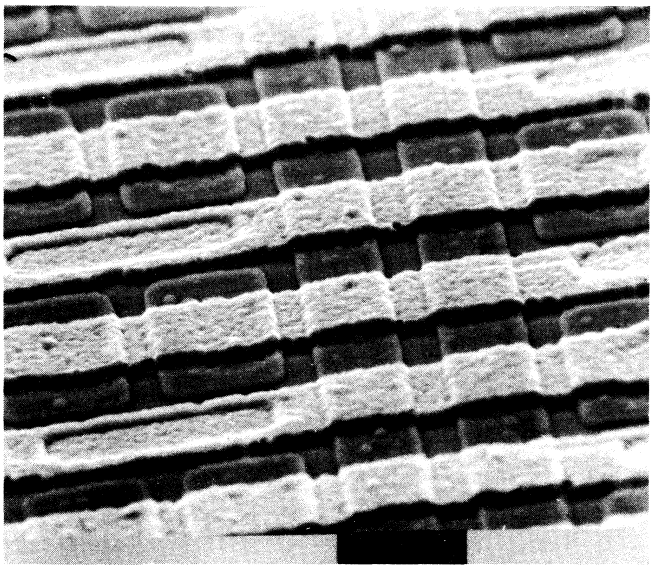
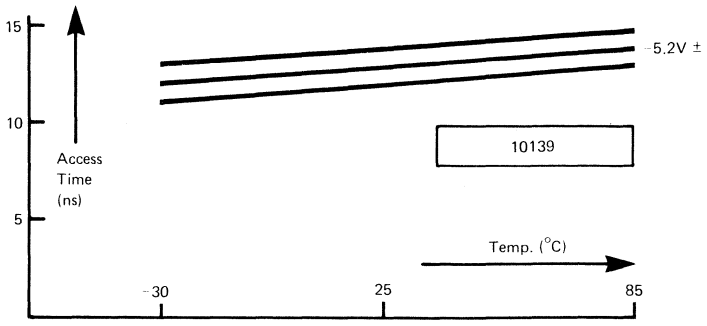


FIG. 4: PROM matris showing first and second layer aluminum interconnects.

The use of lower resistance fuses allowed by the 350 A° nichrome layer film also serves to further reduce RC time constants.

Access Time Versus V_{CC} and Temperature for Typical 10139



10149 Preliminary Data

MEASURED PROPAGATION DELAY (ns)

V_{EE}	−30°C	25°	85°
−4.68V	9.3	10.3	12.1
−5.2V	9.3	10.0	12.0
−5.72V	9.1	10.0	12.0

The high levels of performance achieved can be seen from the diagram and table above.

REFERENCES

1. Shields et al U.S. Patent No. 3, 778,886.
2. Eisenberg, P. J. and Nover, R., 1974, 12th Annual Proc. Rel. Phys., p. 87.
3. Campbell, D. S. and Hendy, B., 1965, Brit. J. Appl. Phys., Vol. 16, p. 1719.
4. Degenhart, H. J. and Pratt, I. H., 1963, Trans. 10th National Physics Symposium, p. 480.
5. Swanson, J. G. and Campbell, D. S., 1967, Thin Solid Films, p. 183.
6. Mo, R. S. and Gilbert, D. M., 1973, J. Electrochem. Soc., Vol. 120, No. 7, p 1001.
7. Gurev, H., 1974, Extended Extracts, Spring Meeting Electro. Chem. Soc., Vol. 74-1, p. 199.
8. Franklin, P. and Burgess, D., 1974, 12th Annual Proc. Rel. Phys. p. 82.
9. Barnes, D. E. and Thomas, J. E., 1974, 12th Annual Proc. Rel. Phys., p. 74.

GENERIC RELIABILITY DATA FOR SIGNETICS PROMS

An extensive Signetics Reliability Engineering data-gathering effort between 1962 and 1975 resulted in the publication of the June 1975 "Signetics Product Reliability Report—R363." The comprehensive reliability report describes the derivation of the Signetics "Failure Rate Acceleration Factor vs Temperature Curve" (a 0.41 eV activation energy curve) in 1970 based on life test data since 1962. The report also describes the technical approach to categorizing life test data which Signetics started using from 1970 onward to monitor the reliability of the many new die process technologies which began to emerge in 1969. In essence, Signetics recognizes that products within a product line can be made with several die process technologies and several assembly-package configurations. Therefore, Signetics concentrates on die process family studies (design rules apply to die process families) and assembly-package material studies. Assembly-package material studies are not covered herein as the results apply equally to the various die process families which share the assembly-package materials.

Life tests are primarily used for die process family studies. Calculation of failure rates are based upon die process families. Such "family" failure rates can be extended to cover specific devices using that "family" process. As a reference, data generated since 1970 via accelerated life tests shows that failure rates for bipolar die process families range from 0.021% to 0.00049% per 1000 hours. Similarly the MOS die process family failure rates range from 0.031% to 0.0035% per 1000 hours. Both failure rate ranges are calculated at 25°C ambient and at 60% confidence.

Periodically, failure rates continue to be calculated for all current die process families based upon accelerated life test data generated during reliability engineering programs (primarily SURE II data). Because of widespread interest among PROM users about reliability—especially nichrome fuse reliability—Signetics has placed special emphasis on PROM testing. This report presents the latest (Febuary, 1976) failure rate data on PROMs, based upon 100 million 25°C device hours and 136 *billion* nichrome fuse 25°C hours.

TTL PROM RELIABILITY TESTING HISTORY

Signetics performed reliability studies of fused and unfused nichrome links in 1971, subjecting unfused links to 5 times normal current and fused links to 12 volts. In late 1971, Field Programmable Read Only Memories (PROMs) employing the non-Schottky TTL multi-emitter fuse matrix design were offered for sale. Reliability tests were performed during 1971 to 1973 on "single notch" nichrome link product. A few program rejects were detected during the first measurement timepoint (168 hours) of operating life stress with no additional program rejects observed from 168 hours on out to 2000 hours. During that period, program yield (and probably product reliability) was found to be highly influenced by the programming procedure used.

During 1974, reliability tests were started on "second generation" TTL Schottky PROM Products. The "second generation" products use standard Schottky product processing with the addition of nichrome fuses. These products have a diode fusing matrix, use bar fuses, and are designed to supply increased current densities during fusing to enhance fuse reliability and increase programming yields. The reliability data to date is summarized in the table "PROM Life Test Summary: Std. Aluminum Schottky, DLM, Plus Nichrome Fuses." Recent distributions show a 92% AVG. programming yield (based on a 50% blow pattern) for these new Schottky PROM Products. All Signetics PROM products introduced since February 1974 (including the Field Programmable Logic Array) are the "second generation" TTL Schottky type.

TTL SCHOTTKY PROM FAILURE RATES/MTBF

The failure rates and MTBF (mean time between failures) for PROM devices, for fused links, and for unfused links are calculated from the data in the life test summary.

PROCESS	PARAMETER	FAILURE RATE(1)	MTBF =1/F.R.
Std Aluminum Schottky DLM Plus Nichrome Fuses	Device	.000893	$1.12X10^{8}$ hours
	Fused Link	.00000137	$7.30X10^{10}$ hours
	Unfused Link	.00000132	$7.58X10^{10}$ hours

(1) The failure rate (F.R.) calculations are at 60% confidence and values are shown in % per 1000 hours. Calculations are based on the 25°C equivalent device hours (combined HTOL and HTSL).

The failure rate for any given Signetics PROM can be approximated by multiplying the fuse failure rate by the number of fuses in the device and adding the result to the basic device reliability. For example, the FPLA has 1920 nichrome links. Assuming that 50% (960) are blown, we get

$$960 \times .00000137 + 960 \times .00000132 + .000893 = .00347\%/1000 \text{ Hours}$$

PROM LIFE TEST SUMMARY: Standard Aluminum Schottky, DLM, Plus Nichrome Fuses

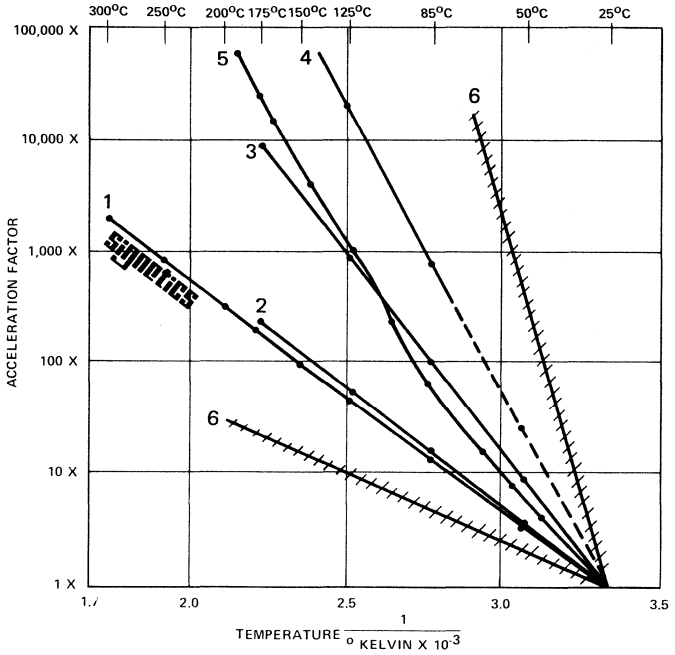
TEST NUMBER	PRODUCT	STRESS	AMBIENT TEMPERATURE	DEVICE QUANTITY TESTED	POWER (MW) TYPICAL	PACKAGE THERMAL RESISTANCE (°C/W)	RISE IN CHIP TEMPERATURE	JUNCTION TEMPERATURE (T_j)	ACCEL ATIC FACTO
SQ74044	82S23B 32X8 bit	Dynamic HTOL[3]	85°C	36	358	165	59°C	144°C	87
DT74134A	82S126I 256X4 bit	Dynamic HTOL[3]	125°C	55	525	83	44°C	169°C	180
DT74134B	82S126I 256X4 bit	Storage	150°C	64	NA	NA	N/A	150°C	100
DT74134C	82S129I 256X4 bit	Dynamic HTOL[3]	125°C	41	525	83	44°C	169°C	180
DT74134D	82X129I 256X4 bit	Storage	150°C	43	NA	NA	NA	150°C	1[illegible]
SQ74069	82S129I 256X4 bit	Dynamic HTOL[3]	125°C	45	525	83	44°C	169°C	180
SQ75045A	82S115I 512X8 bit	Dynamic HTOL[3]	85°C	46	675	50	34°C	119°C	45
SQ75045B	82S115I 512X8 bit	Storage	150°C	45	NA	NA	NA	150°C	100
SQ75065A	82S126F 256X4 bit	Dynamic HTOL[3]	125°C	46	525	90	47°C	172°C	185
SQ75065B	82S126F 256X4 bit	Storage	150°C	46	NA	NA	NA	150°C	100
SQ75046A	82S130F 512X4 bit	Dynamic HTOL[3]	125°C	46	600	90	54°C	179°C	225
SQ75046B	82S130F 512X4 bit	Storage	150°C	46	NA	NA	NA	150°C	100
DT75037B	82S126/129F 256X4 bit	Dynamic HTOL[3] (2000 Hr)	125°C	47	525	90	47°C	172°C	185
DT75037A	82S126/129F 256X4 bit	Storage (2000 Hr)	150°C	47	NA	NA	NA	150°C	100

(1) Subtracted from the quantity tested and the number of failures were all rejects for which subsequent failure analysis showed the cause of failure to be "electrical overstress/mishandling"
(2) Catastrophic failures are opens, shorts, or non-functional parts
(3) High temperature operating life included dynamic exercising of all addresses.
(4) Acceleration factor applied to T_j from the acceleration curve #1, shown below.

HOURS TRESS	FUSE HOURS ON STRESS		25°C EQUIVALENT DEVICE HOURS	25°C EQUIVALENT FUSE HOURS		NUMBER OF DEVICE FAILURES (1), (2)	NUMBER OF FUSE FAILURES	
	ZEROS (UNFUSED) (10^6 Hrs)	ONES (FUSED)		ZEROS (UNFUSED) (10^6 Hrs)	ONES (FUSED)		ZERO	ONE
000	9.216	0	3.13	801.79	0	0	0	0
,000	28.16	28.16	9.90	5068.80	5068.80	0	0	0
,000	32.77	32.77	6.40	3277.00	3277.00	0	0	0
,000	20.99	20.99	7.38	3778.20	3778.20	0	0	0
,00	22.02	22.02	4.30	2202.00	2202.00	0	0	0
,000	23.04	23.04	8.10	4147.20	4147.20	0	0	0
,000	94.21	94.21	2.07	4239.45	4239.45	0	0	0
,000	92.16	92.16	4.50	9216.00	9216.00	0	0	0
,000	23.55	23.55	8.51	4356.75	4356.75	0	0	0
,000	23.55	23.55	4.60	2355.00	2355.00	0	0	0
,000	47.10	47.10	10.35	10597.50	10597.50	0	0	0
,000	47.10	47.10	4.60	4710.00	4710.00	0	0	0
000	48.13	48.13	17.39	8904.05	8904.05	0	0	0
,000	48.13	48.13	9.40	4813.00	4813.00	0	0	0
TOTAL			1.006×10^8 hours	6.847×10^{10} hours	6.767×10^{10} hours	0	0	0

FAILURE RATE ACCELERATION FACTOR vs. TEMPERATURE CURVE

(from Signetics Product Reliability Report R363)



NOTES:

(1) Calculated from the Signetics Failure Rate vs Temperature Graph of Figure 3.2. Signetics uses acceleration factors of 15 (for 85°C), 50 (for 125°C), 100 (for 150°C), 200 (for 175°C), 350 (for 200°C), 970 (for 250°C) and 2100 (for 300°C) to relate to 25°C equivalent ambient temperature. The 25°C to 125°C segment of the graph is based primarily on operating life data. The segment of the graph above 125°C is based on high temperature storage data. The graph equates to an "activation energy" Ea = 0.41 eV.

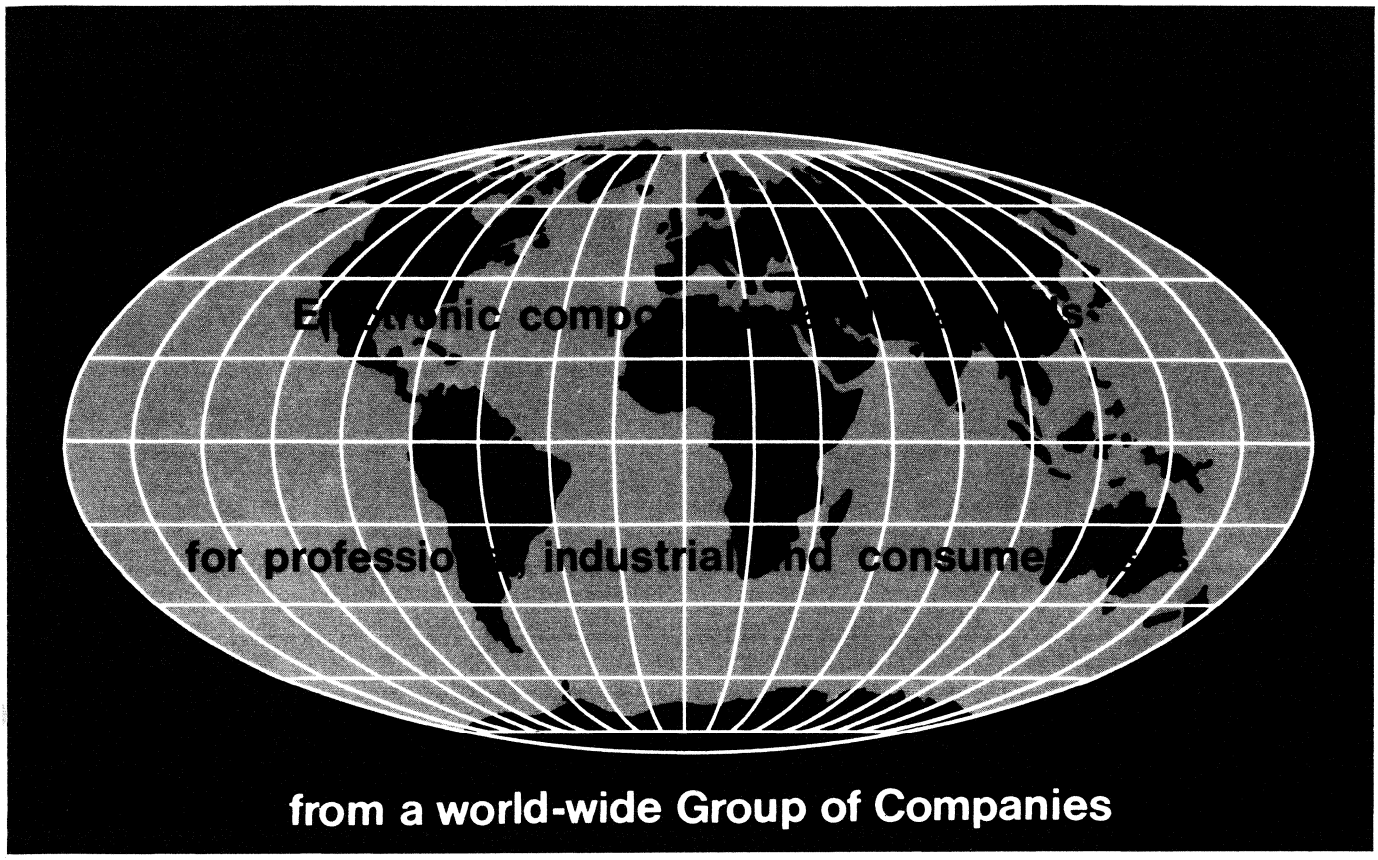
(2) Calculated from MIL-HDBK-217B, 20 September, 1974, Table 2.1.5-4 for ΠT_1 vs T_J values. The graph equates to an "activation energy" $E_A = 0.41$ eV and is applicable to all bipolar digital (excépt ECL) in the normal mode of operation.

(3) Calculated from MIL-HDBK-217B, 20 September, 1974, Table 2.1.5-4 for ΠT_2 vs T_J values. The graph equates to an "activation energy" $E_A = 0.70$ eV and is applicable to all MOS, all Linear, and bipolar ECL devices in the normal mode of operation.

(4) Calculated from MIL-STD-883A, 15 November 1974, Figures 1005-4 and 1015-1 by extrapolating the time temperature regression graph from 78°C back to 25°C. The MIL-STD-883A graph is the Bell Telephone Laboratories Graph (Specification A-B-689143, 16 January 1974 etc.) and as such applies to storage and operating T_J values and primarily surface inversion failure mechanisms. The graph equates to an "activation energy" $E_A = 1.02$ eV.

(5) This curved graph is the result of plotting the "rule of thumb" that failure rates (hence acceleration factors) double for every + Δ 10°C.

(6) All competitor data (available to Signetics) produced graphs falling within these two boundaries. The two boundaries equate to "activation energies" of $E_A = 0.23$ eV (for lower graph) and $E_A = 1.92$ eV.



EUROPEAN SALES OFFICES

Austria: Österreichische Philips, Bauelemente Industrie G.m.b.H., Zieglergasse 6, Tel. 93 26 22, A-1072 WIEN.
Belgium: M.B.L.E., 80, rue des Deux Gares, Tel. 523 00 00, B-1070 BRUXELLES.
Denmark: Miniwatt A/S, Emdrupvej 115A, Tel. (01) 69 16 22, DK-2400 KØBENHAVN NV.
Finland: Oy Philips Ab, Elcoma Division, Kaivokatu 8, Tel. 1 72 71, SF-00100 HELSINKI 10.
France: R.T.C., La Radiotechnique-Compelec, 130 Avenue Ledru Rollin, Tel. 355-44-99, F-75540 PARIS 11.
Germany: Valvo, UB Bauelemente der Philips G.m.b.H., Valvo Haus, Burchardstrasse 19, Tel. (040) 3296-1, D-2 HAMBURG 1.
Greece: Philips S.A. Hellénique, Elcoma Division, 52, Av. Syngrou, Tel. 915 311, ATHENS.
Ireland: Philips Electrical (Ireland) Ltd., Newstead, Clonskeagh, Tel. 69 33 55, DUBLIN 14.
Italy: Philips S.p.A., Sezione Elcoma, Piazza IV Novembre 3, Tel. 2-6994, I-20124 MILANO.
Netherlands: Philips Nederland B.V., Afd. Elonco, Boschdijk 525, Tel. (040) 79 33 33, NL-4510 EINDHOVEN.
Norway: Electronica A.S., Vitaminveien 11, Tel. (02) 15 05 90, P. O. Box 29, Grefsen, OSLO 4.
Portugal: Philips Portuguesa S.A.R.L., Av. Eng. Duharte Pacheco 6, Tel. 68 31 21, LISBOA 1.
Spain: COPRESA S.A., Balmes 22, Tel. 301 63 12, BARCELONA 7.
Sweden: ELCOMA A.B., Lidingövägen 50, Tel. 08/67 97 80, S-10 250 STOCKHOLM 27.
Switzerland: Philips A.G., Elcoma Dept., Edenstrasse 20, Tel. 01/44 22 11, CH-8027 ZÜRICH.
Turkey: Türk Philips Ticaret A.S., EMET Department, Gümüssuyu Cad. 78-80, Tel. 45.32.50, Beyoglü, ISTANBUL.
United Kingdom: Mullard Ltd., Mullard House, Torrington Place, Tel. 01-580 6633, LONDON WC1E 7HD.

Printed in The Netherlands

9399 509 53061

signetics

a subsidiary of **U.S. Philips Corporation**

Signetics Corporation
811 East Arques Avenue
Sunnyvale, California 94086
Telephone 408/739-7700