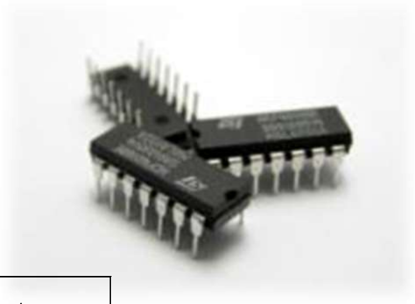
University Of Diyala
College Of Engineering
Computer Engineering Department

Introduction of Combinational Logic Design

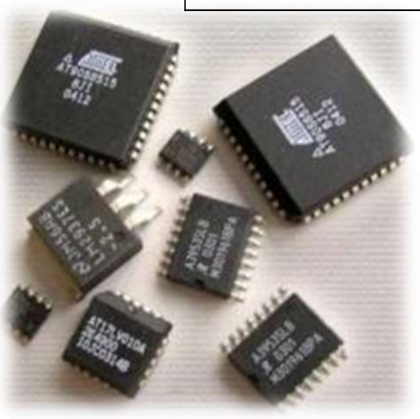
Dr. Yasir Amer Abbas
Third stage
2020

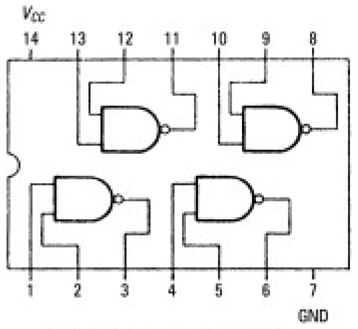
IMPLEMENTING LOGIC FUNCTIONS USING (MSI) AND PROGRAMMABLE DEVICES

Type of Circuits



Type of circuits	Number of gates
Small-scale integration (SSI)	1-10
Medium-scale integration (MSI)	10-100
Large-scale integration (LSI)	100-1,000
Very-large-scale integration (VLSI)	1,000 up
Ultra-large-scale integration (ULSI)	1,000,000





7400 Quadruple 2-input NAND gates

Implementation of Logic Circuits

Implementation considerations

- What should an engineer consider when selecting a implementation medium?

  ⇨ Power consumption limits.

  ⇨ Speed performance required.

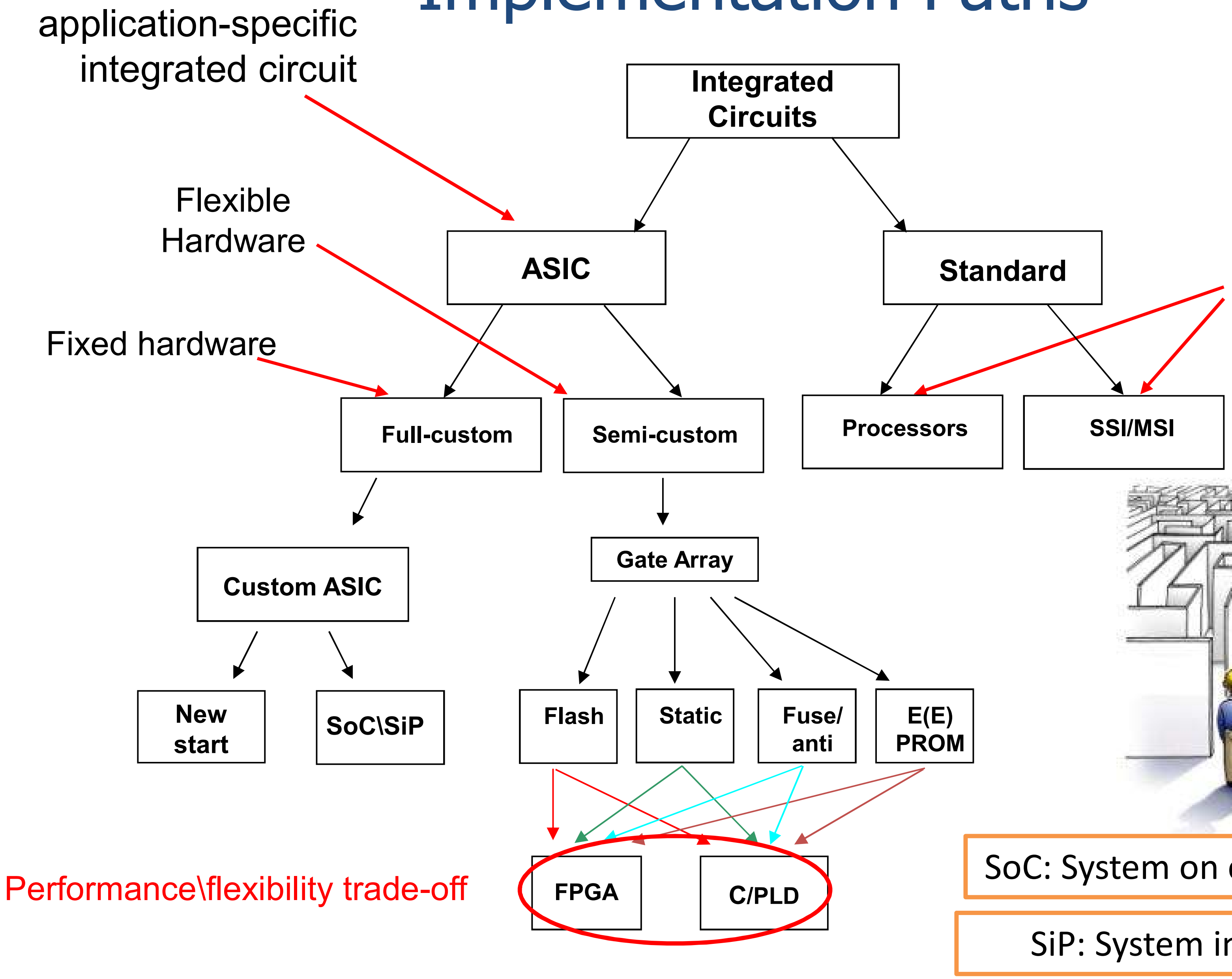
  ⇨ Device area\size limits.

  ⇨ Flexibility\re-use required.

  ⇨ Domain specific features required

  ⇨ Financial cost constraints (can depend on sales volume).

Implementation Paths



Performance\flexibility trade-off

SoC: System on chip

SiP: System in package

Implementation Trade-offs

- Provides definite implementation paths for designers with estimated costs and design-times.

- Particular devices can provide speed and low-power but without post-fabrication design flexibility.

- ASICs (semi) provide flexibility and low power for designs, but with larger financial costs and substantial design-times.

- Standard provide lower cost off the shelve computing component and shorter design-times, but possibly providing lower performance and higher power consumption.

- Financial cost plays a major role in deciding which path a initial design implementation will follow.

Standard Device Technology

- Standard devices are typically off-the-shelve computing components
- Provide low cost solutions
- Power consumption and performance vary between devices
- Devices re-programmed by software, i.e. limited flexibility
- Inherently sequential, limited parallelism exploited
- Short design time
- Several different device technologies available including:
  - ⇨ Microprocessor
  - ⇨ Microcontroller (Harvard, 8-bit to 64-bit)
  - ⇨ DSPs, (32-bit)

Full-custom Technology

- Full-custom designs can provide optimal implementations, i.e. lowest power consumption, fastest execution speeds
- Expensive due to cost of mask designs and small volume for production.
- Designs cannot be altered after fabrication, i.e. no re-programmability or design flexibility
- Design times can be substantial up to 18 months
- Designs re-spins are often required increasing design time
- Example ASIC chip application include digital TV and VoIP

Semi-custom Technology

- Semi-custom designs provide sub-optimal implementations, i.e. power consumption can be modest compared to full-custom and execution speeds are lower

- Non-expensive compared to full-custom as cost of mask design can be amortized over large volumes for production

- Designs can be altered after fabrication, i.e. re-programmed

- Design times can be less as standard cell and gate arrays are pre-defined components that can be easily incorporated, 9-12 months

- The trade-off of performance to obtain increased flexibility and lower costs has seen the creation of programmable logic

Shannon's Expansion Theorem

T11a:

$$F(X1,\ldots,Xn) = F(0, X2,\ldots,Xn) \bullet \overline{X1} + F(1, X2,\ldots,Xn) \bullet X1$$

T11b:

$$F(X1,\ldots,Xn) = [F(1, X2,\ldots,Xn) + \overline{X1}] \bullet [F(0, X2,\ldots,Xn) + X1]$$

Shannon's Expansion Theorem

Let X1=0 in T11a

$$F(X1,\ldots,Xn) = F(0,X2,\ldots,Xn)\bullet\overline{0} + F(1,X2,\ldots,Xn)\bullet 0$$

$$F(X1,\ldots,Xn) = F(0,X2,\ldots,Xn)\bullet 1 + F(1,X2,\ldots,Xn)\bullet 0$$

$$F(X1,\ldots,Xn) = F(0,X2,\ldots,Xn)$$

Let X1=1 in T11a

$$F(X1,\ldots,Xn) = F(0,X2,\ldots,Xn)\bullet\overline{1} + F(1,X2,\ldots,Xn)\bullet 1$$

$$F(X1,\ldots,Xn) = F(0,X2,\ldots,Xn)\bullet 0 + F(1,X2,\ldots,Xn)\bullet 1$$

$$F(X1,\ldots,Xn) = F(1,X2,\ldots,Xn)$$

Shannon's Expansion Theorem

$$F(X1,\ldots,Xn) = F(0,0,X3,\ldots,Xn)\bullet\overline{X1}\bullet\overline{X2} + F(0,1,X3,\ldots,Xn)\bullet\overline{X1}\bullet X2$$

$$+F(1,0,X3,\ldots,Xn)\bullet X1\bullet\overline{X2} + F(1,1,X3,\ldots,Xn)\bullet X1\bullet X2$$

$$= I0\bullet\overline{X1}\bullet\overline{X2} + I1\bullet\overline{X1}\bullet X2 + I2\bullet X1\bullet\overline{X2} + I3\bullet X1\bullet X2$$

$$= I0\bullet m0 + I1\bullet m1 + I2\bullet m2 + I3\bullet m3$$

$$= \sum_{k=0}^{2^n-1} Ki\bullet m_i \qquad \text{Where } m_i = m_i(X1, X2)$$

Example 7-1/p333

- Design a circuit using MUX to implement the following function by applying Shannon's Expansion Theorem T11a with respect to A and B

$$F(A,B,C) = A + B.\overline{C}$$

Solution

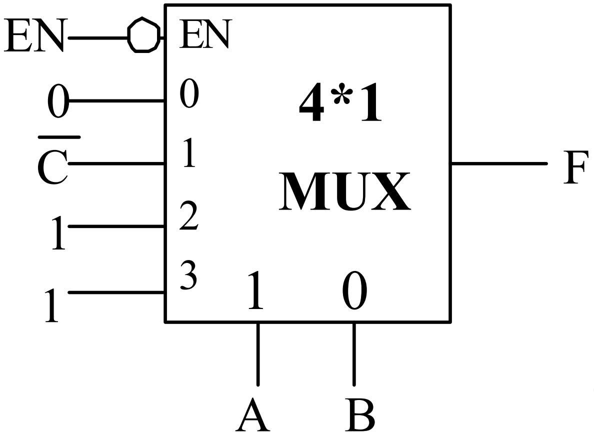
$$F(A,B,C) = F(0,0,C).\overline{A}.\overline{B} + F(0,1,C).\overline{A}.B$$
$$+ F(1,0,C).A.\overline{B} + F(1,1,C).A.B$$

$$F(0,0,C) = 0 + 0.\overline{C} = 0$$

$$F(0,1,C) = 0 + 1.\overline{C} = \overline{C}$$

$$F(1,0,C) = 1 + 0.\overline{C} = 1$$

$$F(1,1,C) = 1 + 1.\overline{C} = 1$$



Analyzing a Multiplexer Design

$$F(A,B,C)=\sum_{k=0}^{2^n-1} Ki\bullet m_i \qquad \text{Where } m_i = m_i(A,B)$$

$$=I0\bullet m0+I1\bullet m1+I2\bullet m2+I3\bullet m3$$

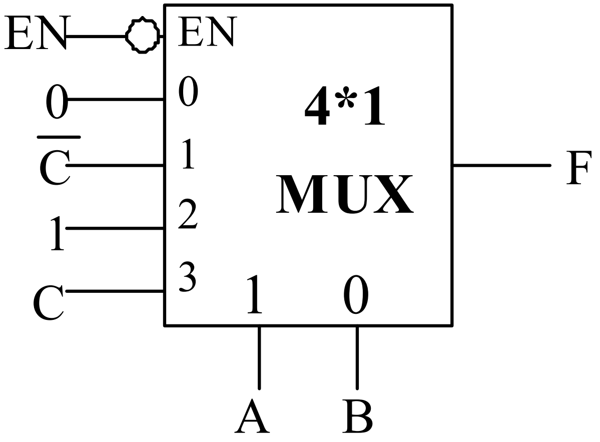
$$=0\bullet m0+\overline{C}\bullet m1+1\bullet m2+C\bullet m3$$

$$=0.\overline{\mathrm{A}}.\overline{\mathrm{B}}+\overline{\mathrm{C}}.\overline{\mathrm{A}}.\mathrm{B}+1.\mathrm{A}.\overline{\mathrm{B}}+\mathrm{C.A.B}$$

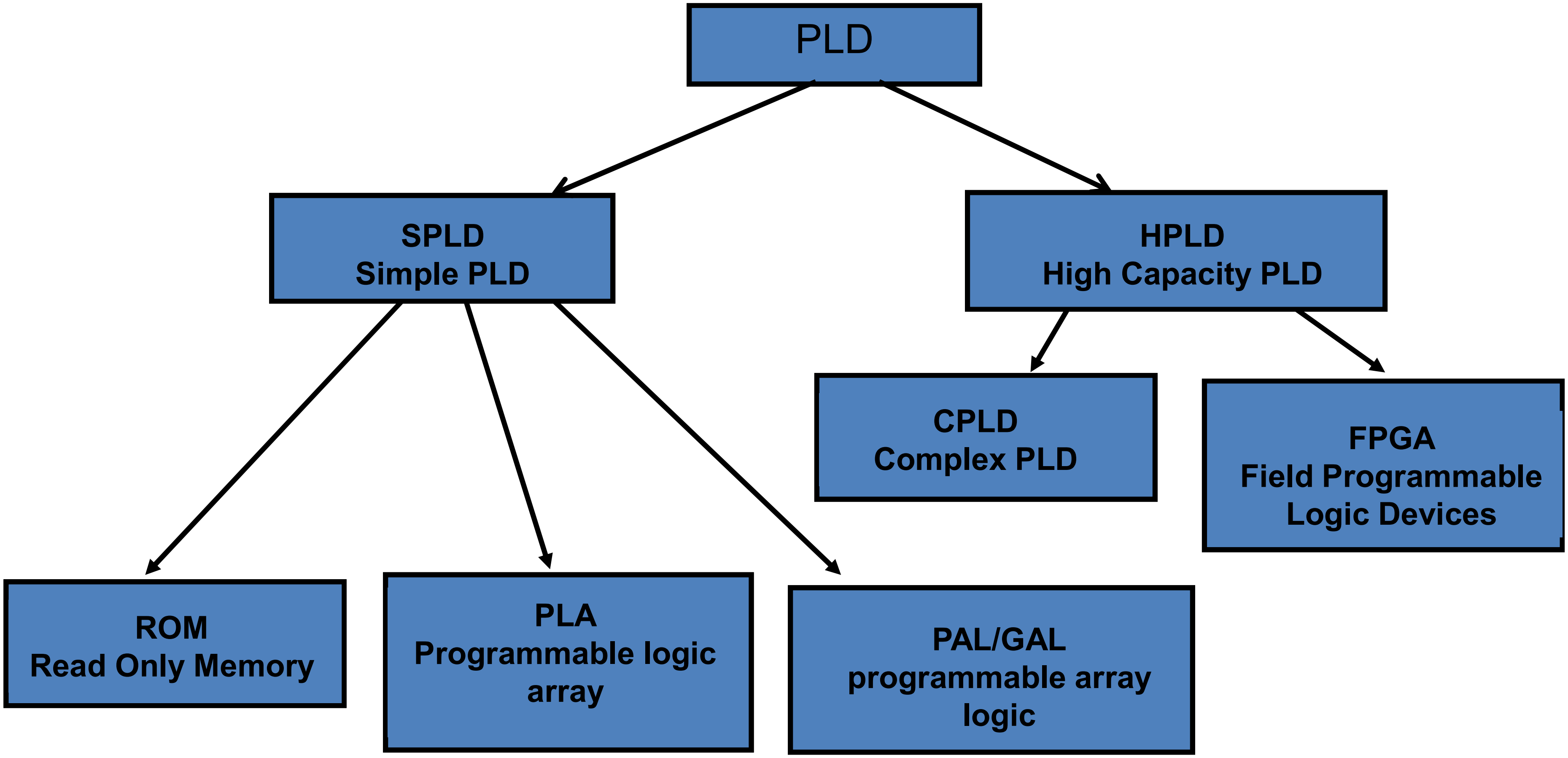
$$=\overline{\mathrm{A}}.\mathrm{B}.\overline{\mathrm{C}}+\mathrm{A}.\overline{\mathrm{B}}.(C+\overline{C})+\mathrm{A.B.C}$$

$$=m_2+m_4+m_5+m_7$$

$$=\sum m(2,4,5,7)$$



Programmable Logic Devices (PLD)



PLD

SPLD
Simple PLD

HPLD
High Capacity PLD

ROM
Read Only Memory

PLA
Programmable logic array

PAL/GAL
programmable array logic

CPLD
Complex PLD

FPGA
Field Programmable Logic Devices

Programmable Logic Devices – Programmable Technologies

- Programmable logic devices are available using several different programmable technologies:

⇨ FUSE
⇨ ANTI-FUSE
⇨ VOLATILE
⇨ NON-VOLATILE
⇨ SWITCH

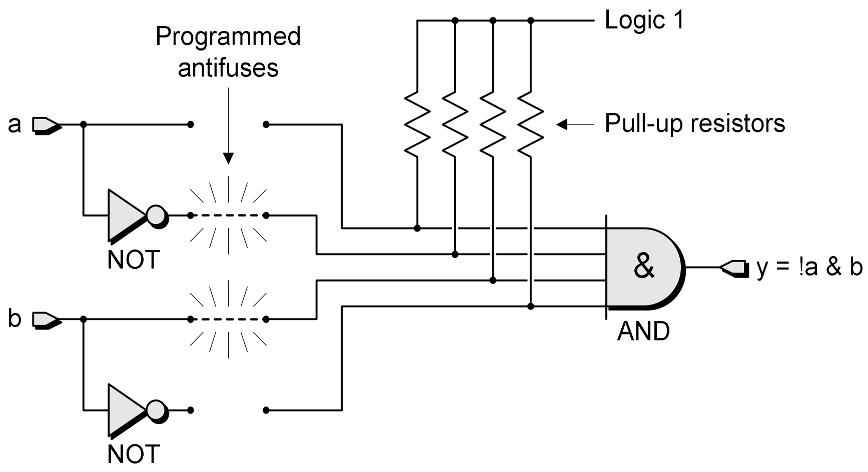
Programmable Logic Devices - programmable technologies

- **Fuse** - This is a two-terminal programmable element that is normally a low resistive element and is programmed or "blown" resulting in an open or high impedance.

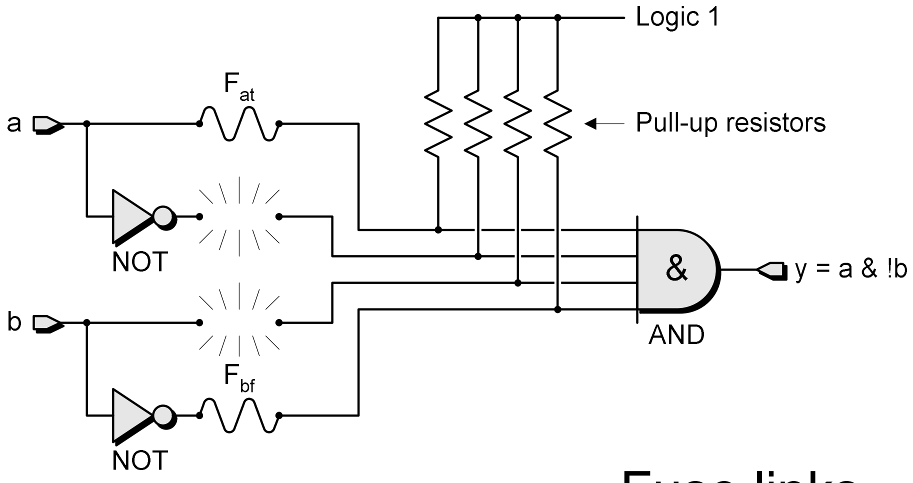
- **Anti-fuse** - This is a two-terminal element that is normally a high resistive element and is programmed to a low impedance.

- **Volatile memory** – (uses actual memory) The memory elements lose their contents when power is removed from the device. SRAM-based devices are volatile and require another device to store their configuration program.

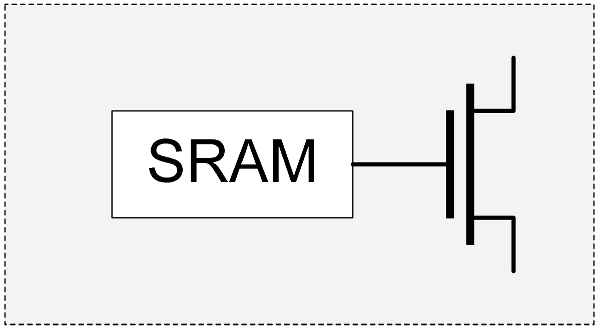
Programmable Logic Devices - programmable technologies



Anti-fuse links

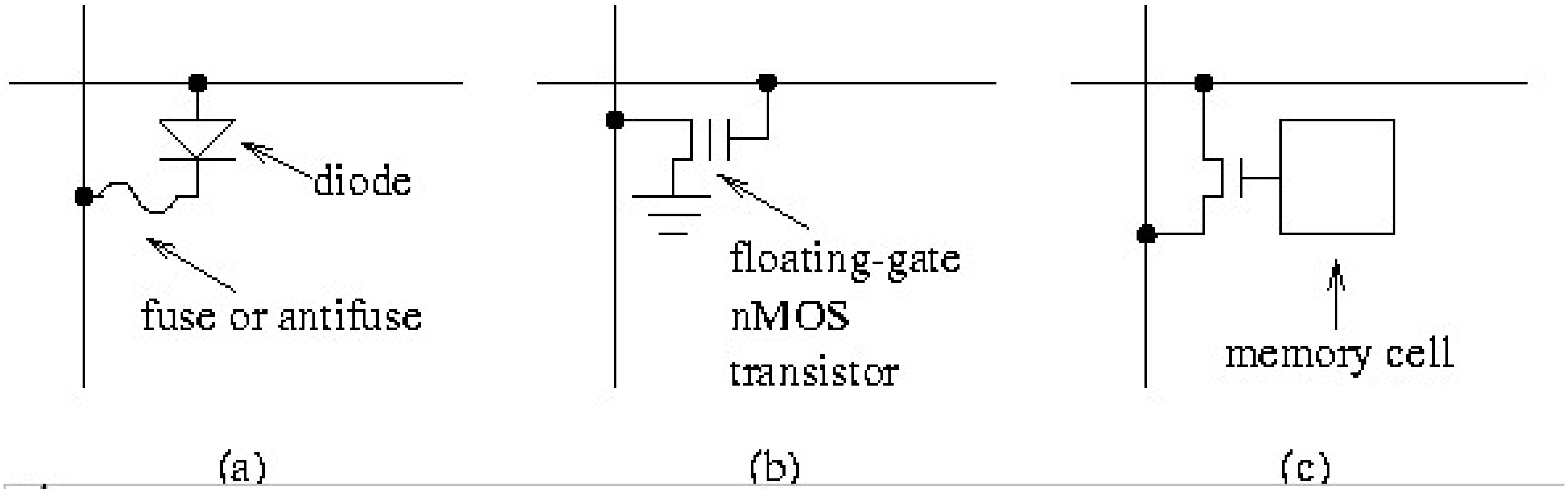


Fuse links



Volatile memory – SRAM programmable

Programmable Technologies



Programmable Logic Devices - programmable elements

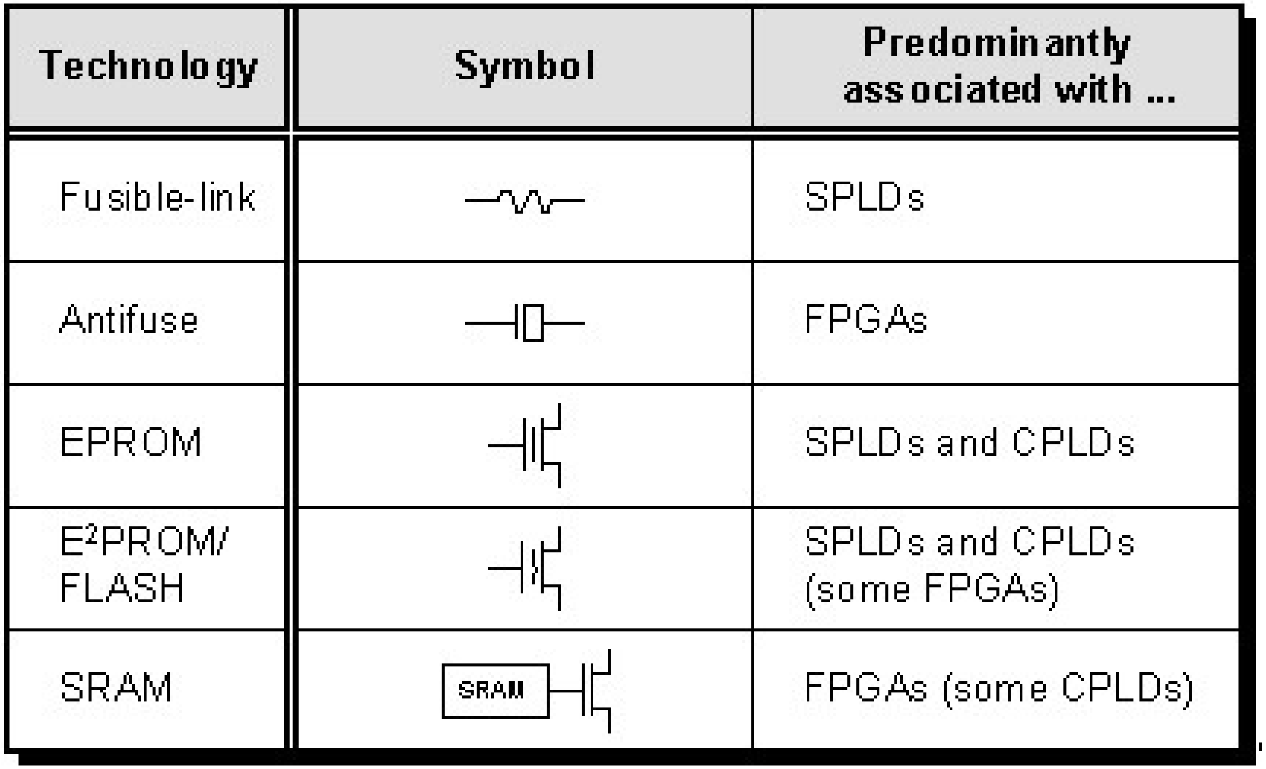
- **Non-volatile memory** - The memory elements keep their contents when power is removed from the device, e.g. Flash, EEPROM, EPROM

⇨ The memory element may be one-time programmable or re-programmable.

⇨ Programmable devices can be both non-volatile and re-programmable.

- **Switch** - This device consists of a memory element (either volatile or non-volatile) which controls a switch. Volatile SRAM-based memory elements are commonly used today.

Programmable Technologies



Technology	Symbol	Predominantly associated with ...
Fusible-link		SPLDs
Antifuse		FPGAs
EPROM		SPLDs and CPLDs
E²PROM/ FLASH		SPLDs and CPLDs (some FPGAs)
SRAM	SRAM	FPGAs (some CPLDs)

Classifying Devices

Device can be classed based on their level of programmability

➡ One Time Programmable: devices can be programmed only once; it's contents can not be changed. While typically these devices are fuse or anti-fuse based, they can also be low-cost EPROM devices.

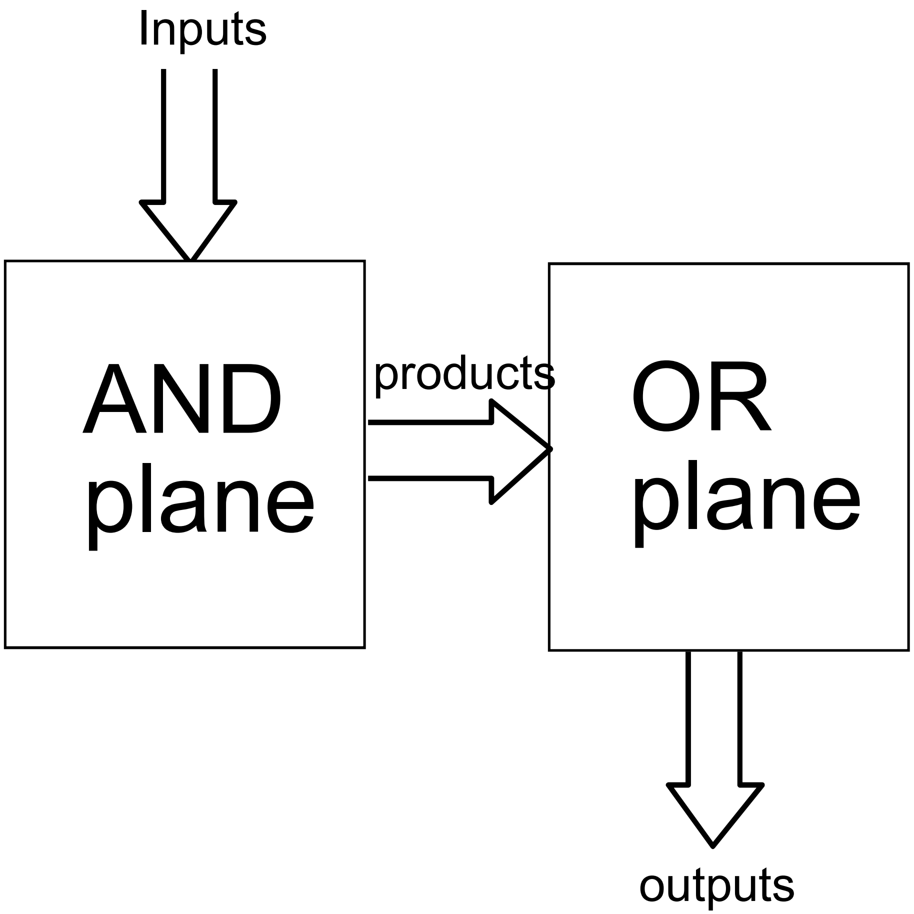
➡ Re-programmable: These devices can have their configuration loaded more than once. SRAM-based and Flash-based devices may be reloaded without restriction.

Advantages of Using PLDs

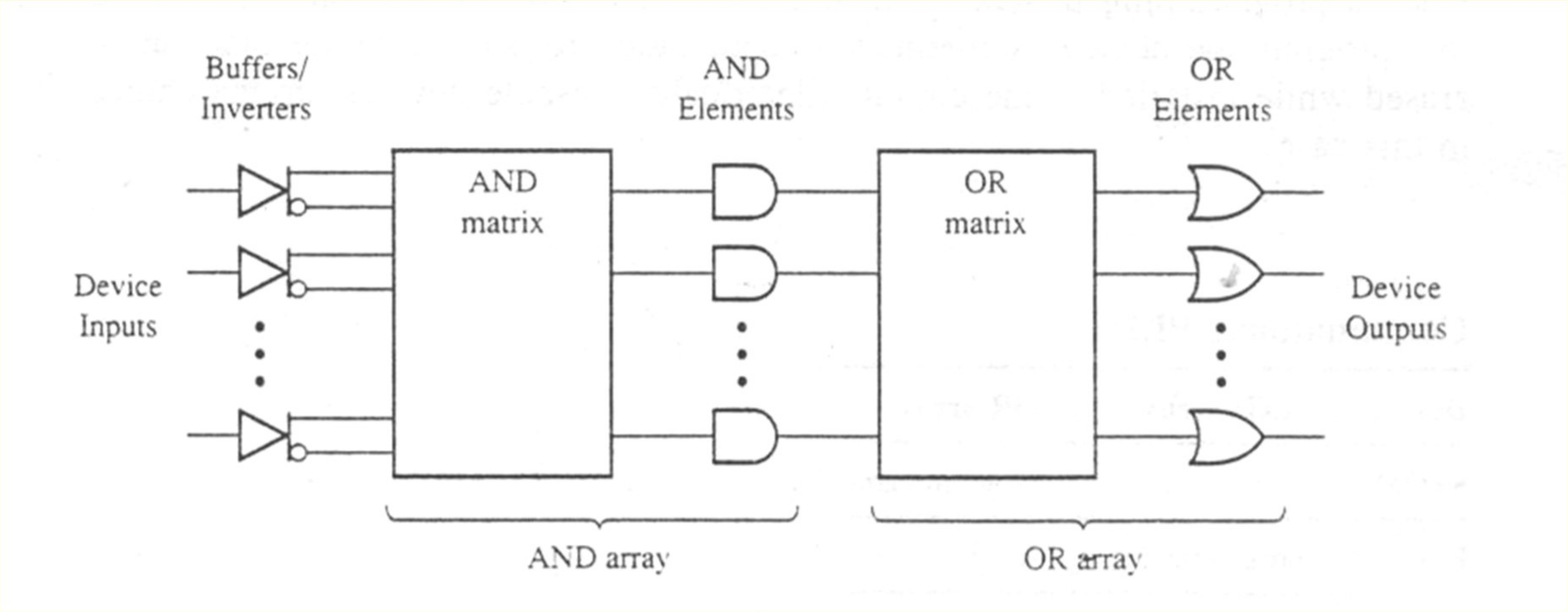
- Less board space.
- Fewer printed circuit boards.
- Smaller enclosures.
- Lower power requirements (i.e., smaller power supplies).
- Faster and less costly assembly processes.
- Higher reliability (fewer ICs and circuit connections & easier troubleshooting).
- Allow design change (availability of design software).

Programmable Logic Device (PLD's)

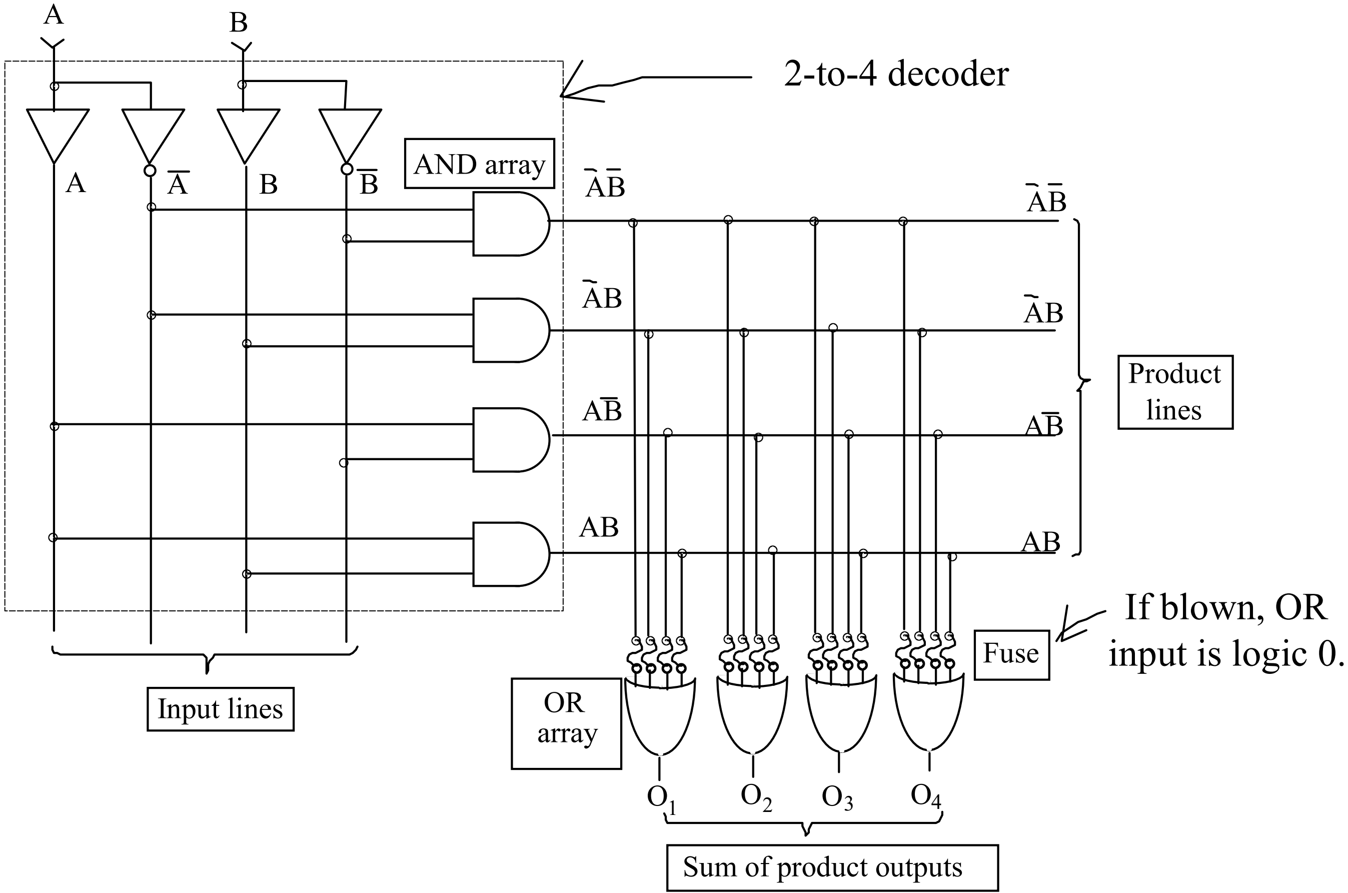
Most of these devices are based on a two level structure (sum of products form).



Programmable Logic Devices



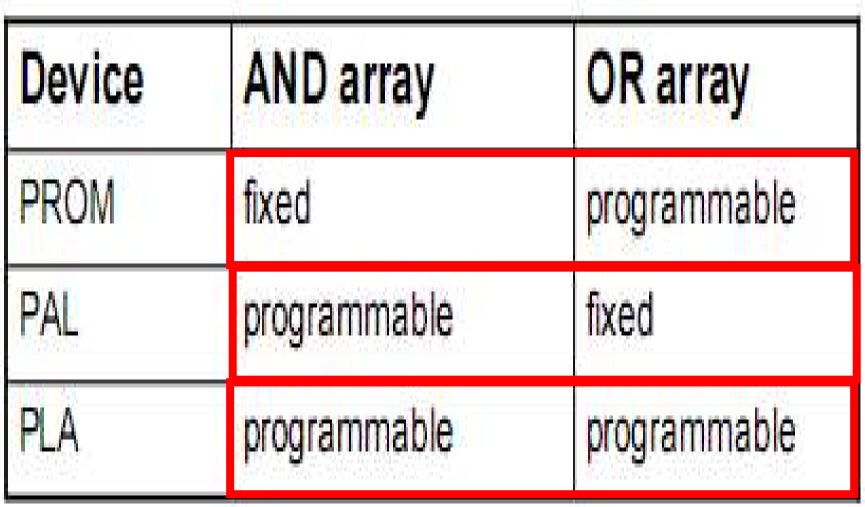
Internal Structures of PLD



Example of a programmable logic device

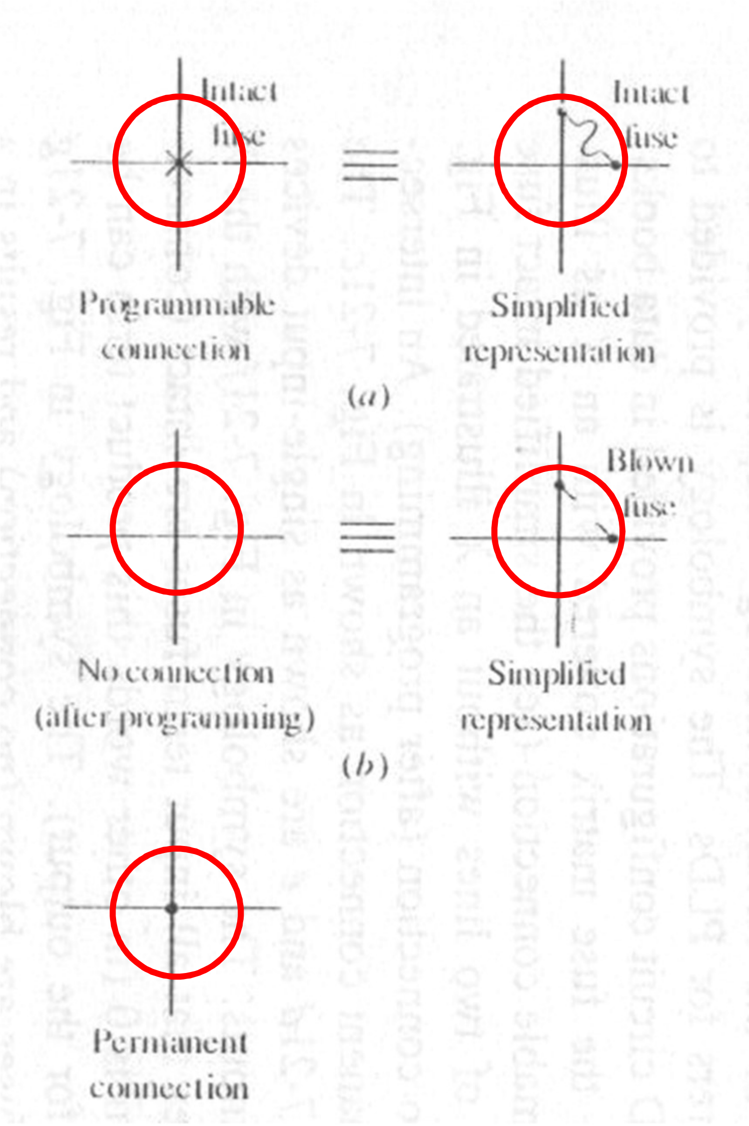
The Main Types of PLD Include:

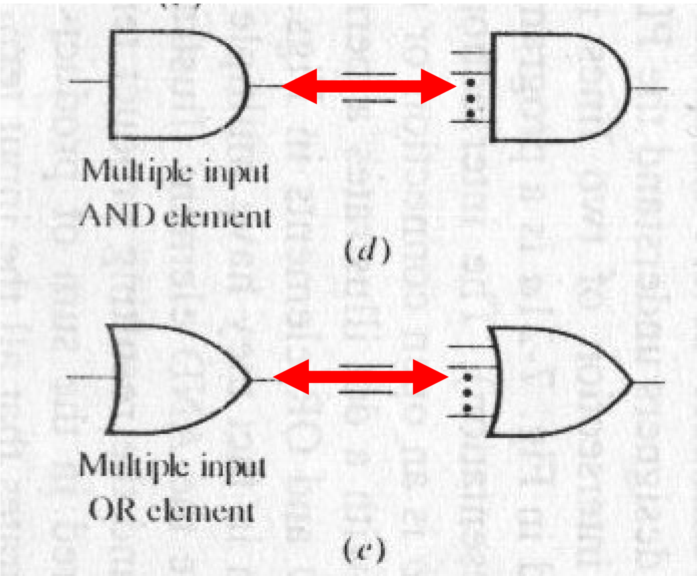
- PROM’s (programmable read only memory)
- PAL’s (programmable array logic)
- PLA’s (programmable logic arrays)

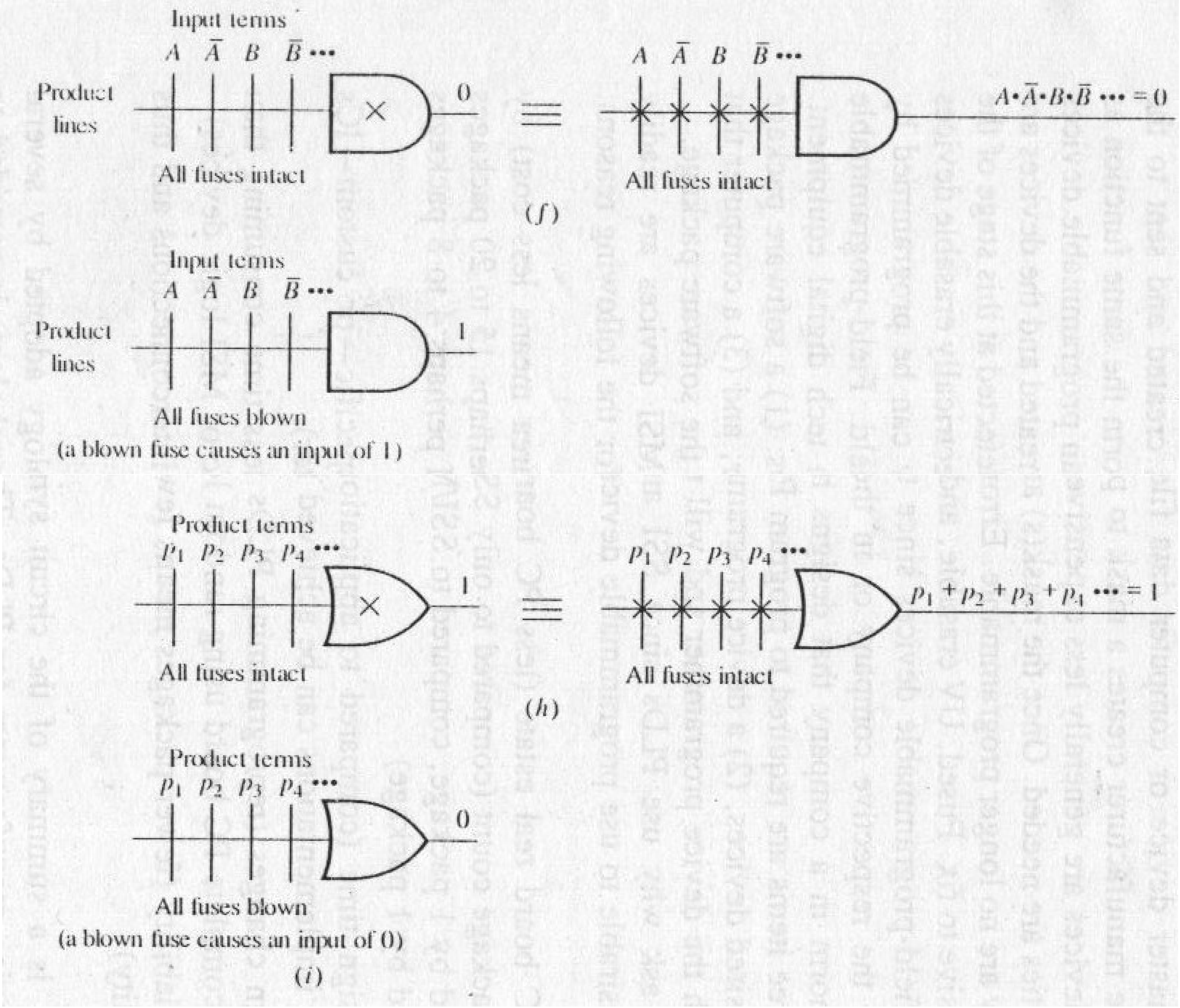


Device	AND array	OR array
PROM	fixed	programmable
PAL	programmable	fixed
PLA	programmable	programmable

PLD Basics


Intact
fuse
≡
Intact
fuse
Programmable
connection
Simplified
representation
(a)
Blown
fuse
≡
No connection
(after programming)
Simplified
representation
(b)
Permanent
connection


≡
Multiple input
AND element
(d)
≡
Multiple input
OR element
(e)


Input terms
A $\bar{A}$ B $\bar{B}$ •••
Product lines
0
All fuses intact
≡
A $\bar{A}$ B $\bar{B}$ •••
$A \cdot \bar{A} \cdot B \cdot \bar{B}$ ••• = 0
All fuses intact
(f)
Input terms
A $\bar{A}$ B $\bar{B}$ •••
Product lines
1
All fuses blown
(a blown fuse causes an input of 1)
Product terms
p_1 p_2 p_3 p_4 •••
1
All fuses intact
≡
p_1 p_2 p_3 p_4 •••
$p_1 + p_2 + p_3 + p_4$ ••• = 1
All fuses intact
(h)
Product terms
p_1 p_2 p_3 p_4 •••
0
All fuses blown
(a blown fuse causes an input of 0)
(i)

1-Read Only Memory (ROM)

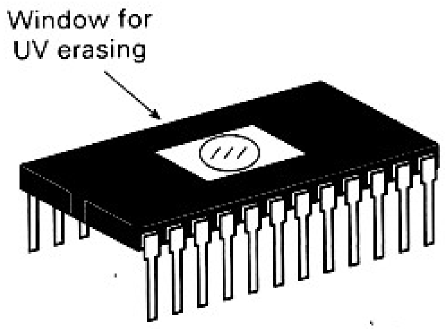
Types of ROM

1. Programmable Read Only Memory (PROM)

- Empty of data when manufactured
- May be permanently programmed by the user

2. Erasable Programmable Read Only Memory (EPROM)

- Can be programmed, erased and reprogrammed
- The EPROM chip has a small window on top allowing it to be erased by shining ultra-violet light on it
- After reprogramming the window is covered to prevent new contents being erased
- Access time is around 45 – 90 nanoseconds



Types of ROM

3. Electrically Erasable Programmable Read Only Memory (EEPROM)

- Reprogrammed electrically **without** using ultraviolet light
- Must be removed from the computer and placed in a special machine to do this
- Access times between 45 and 200 nanoseconds

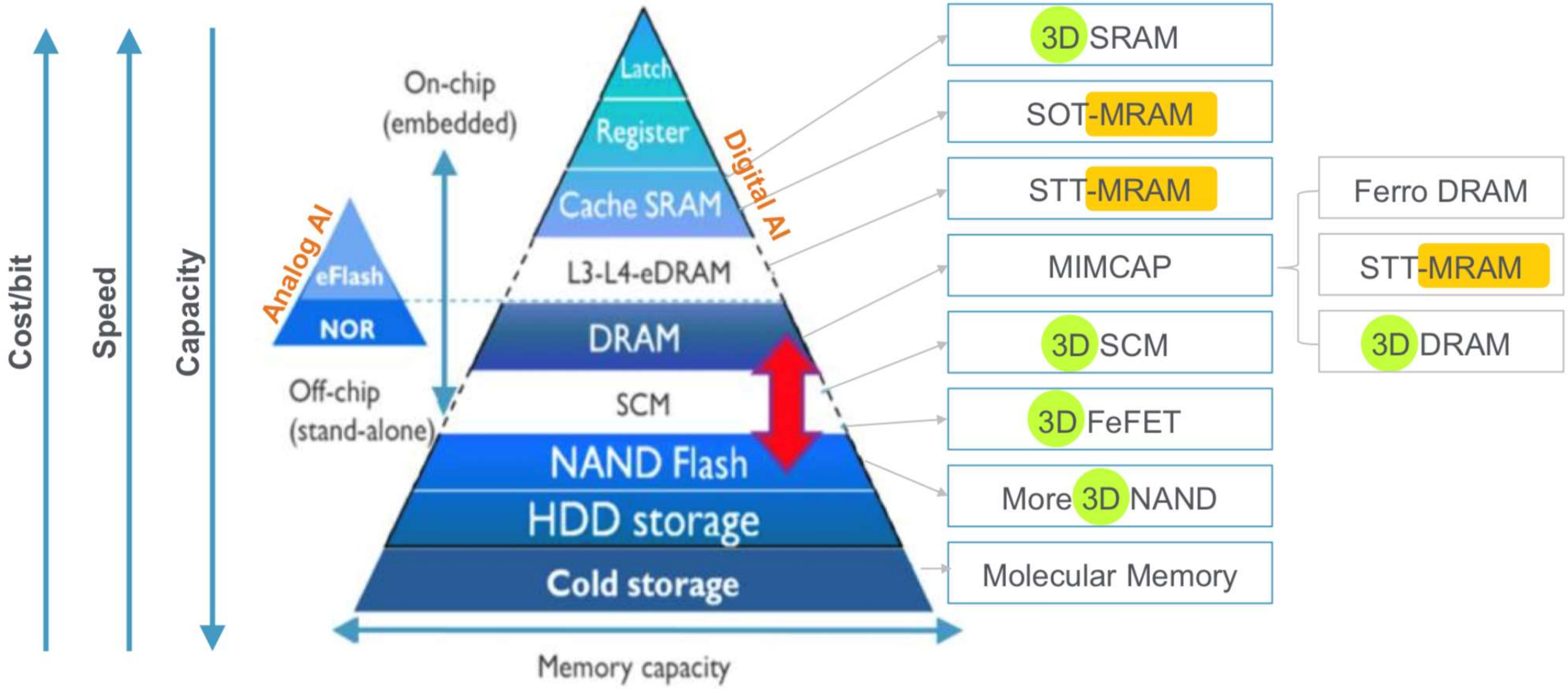
4. Flash ROM

- Similar to EEPROM
- However, can be reprogrammed while still in the computer
- Easier to upgrade programs stored in Flash ROM
- Used to store programs in devices e.g. modems
- Access time is around 45 – 90 nanoseconds

5. ROM cartridges

- Commonly used in games machines
- Prevents software from being easily copied

Types of ROM



https://semiengineering.com/the-next-new-memories/

1-PROM

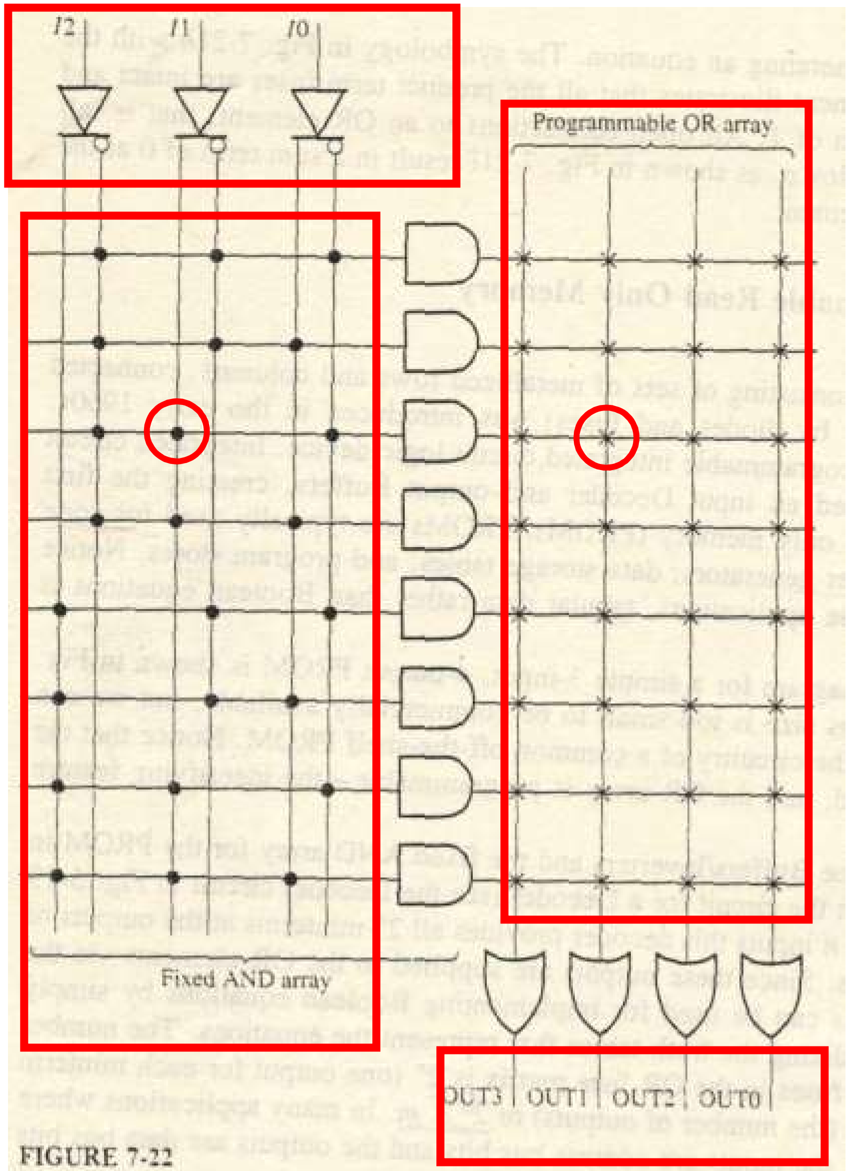
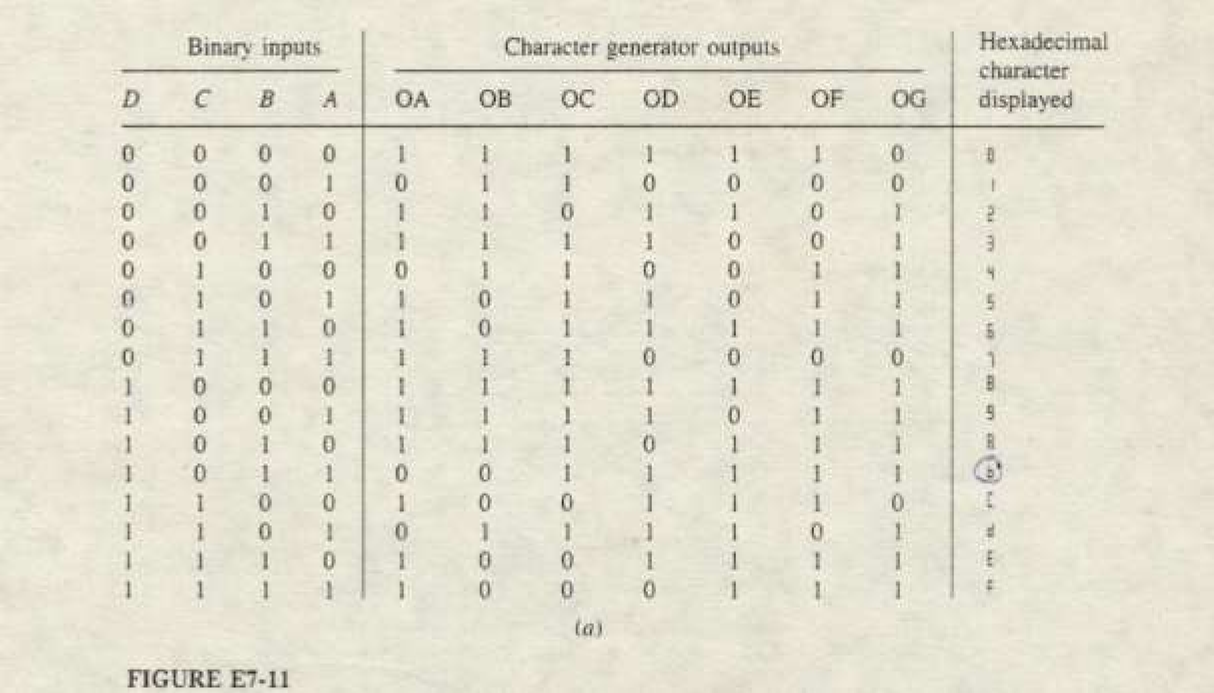


FIGURE 7-22
Simple 3-input, 4-output (8 × 4) programmable read only memory (PROM).

Example 7-11

- **Show a design using ROM to implement the binary to hexadecimal character generator illustrated by the truth table repeated in Fig. E711a.**

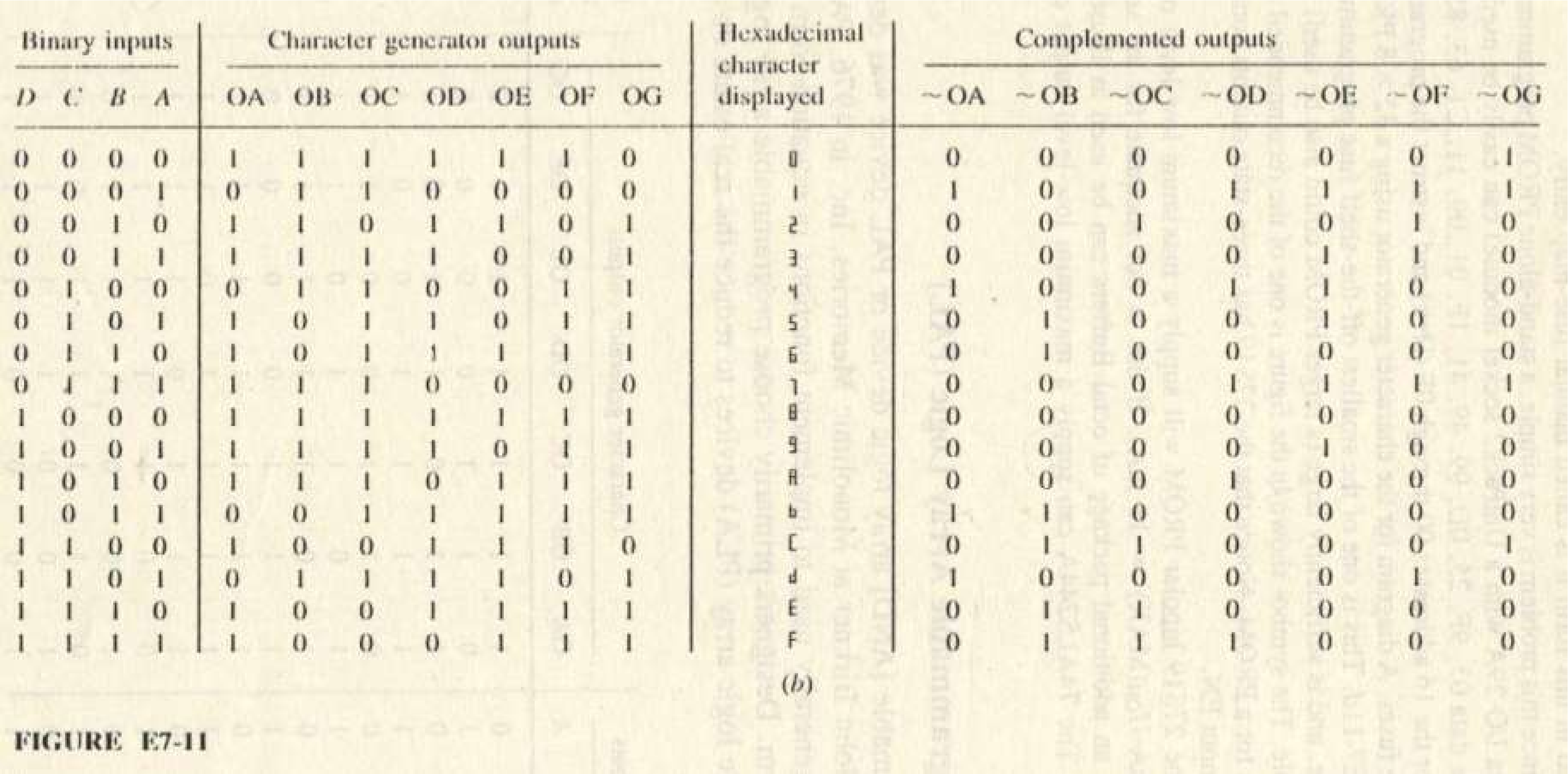


Binary inputs				Character generator outputs							Hexadecimal character displayed
D	*C*	*B*	*A*	OA	OB	OC	OD	OE	OF	OG	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9
1	0	1	0	1	1	1	0	1	1	1	A
1	0	1	1	0	0	1	1	1	1	1	b
1	1	0	0	1	0	0	1	1	1	0	C
1	1	0	1	0	1	1	1	1	0	1	d
1	1	1	0	1	0	0	1	1	1	1	E
1	1	1	1	1	0	0	0	1	1	1	F

(*a*)

FIGURE E7-11

Solution:



Binary inputs				Character generator outputs							Hexadecimal character displayed	Complemented outputs						
D	*C*	*B*	*A*	OA	OB	OC	OD	OE	OF	OG		~OA	~OB	~OC	~OD	~OE	~OF	~OG
0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
0	0	0	1	0	1	1	0	0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	1	1	0	1	1	0	1	2	0	0	1	0	0	1	0
0	0	1	1	1	1	1	1	0	0	1	3	0	0	0	0	1	1	0
0	1	0	0	0	1	1	0	0	1	1	4	1	0	0	1	1	0	0
0	1	0	1	1	0	1	1	0	1	1	5	0	1	0	0	1	0	0
0	1	1	0	1	0	1	1	1	1	1	6	0	1	0	0	0	0	0
0	1	1	1	1	1	1	0	0	0	0	7	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	8	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	0	1	1	9	0	0	0	0	1	0	0
1	0	1	0	1	1	1	0	1	1	1	A	0	0	0	1	0	0	0
1	0	1	1	0	0	1	1	1	1	1	b	1	1	0	0	0	0	0
1	1	0	0	1	0	0	1	1	1	0	C	0	1	1	0	0	0	1
1	1	0	1	0	1	1	1	1	0	1	d	1	0	0	0	0	1	0
1	1	1	0	1	0	0	1	1	1	1	E	0	1	1	0	0	0	0
1	1	1	1	1	0	0	0	1	1	1	F	0	1	1	1	0	0	0

(*b*)

FIGURE E7-11

Solution:

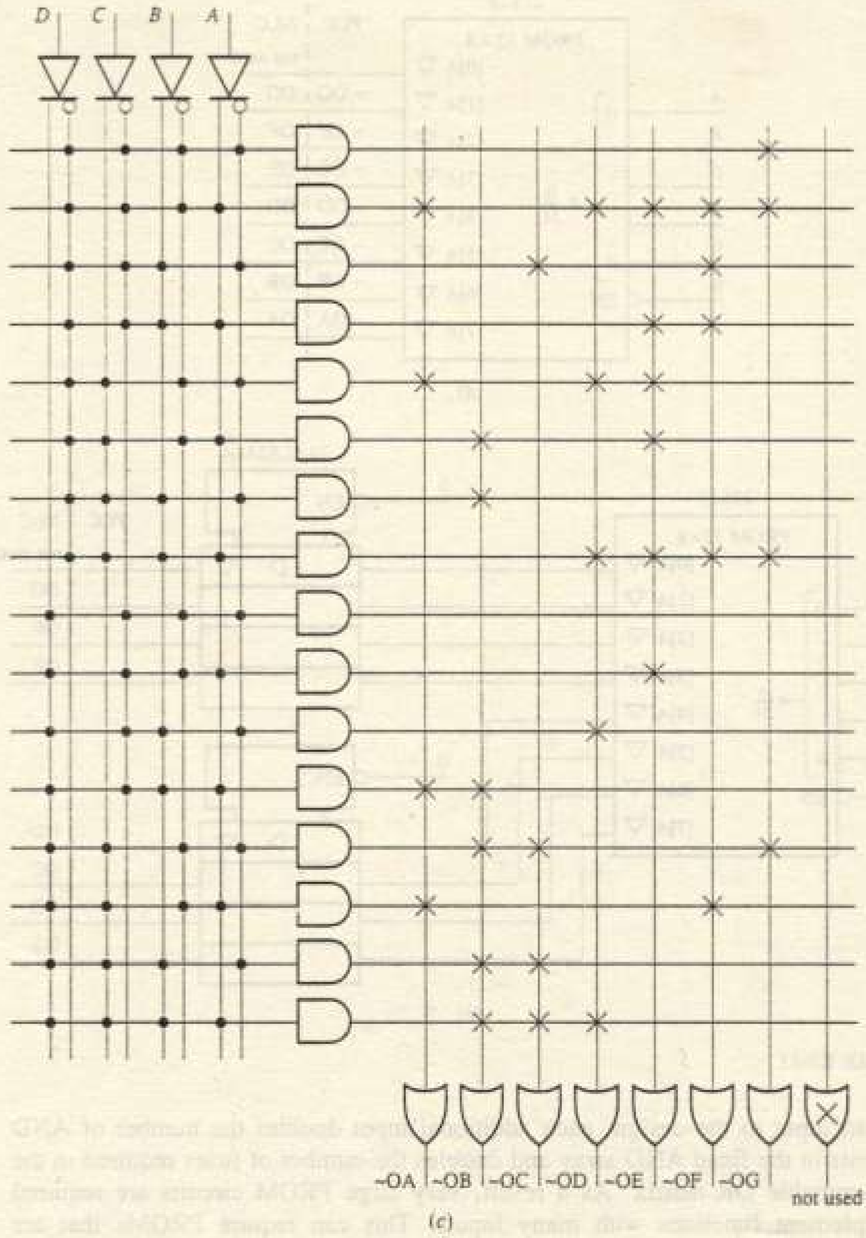
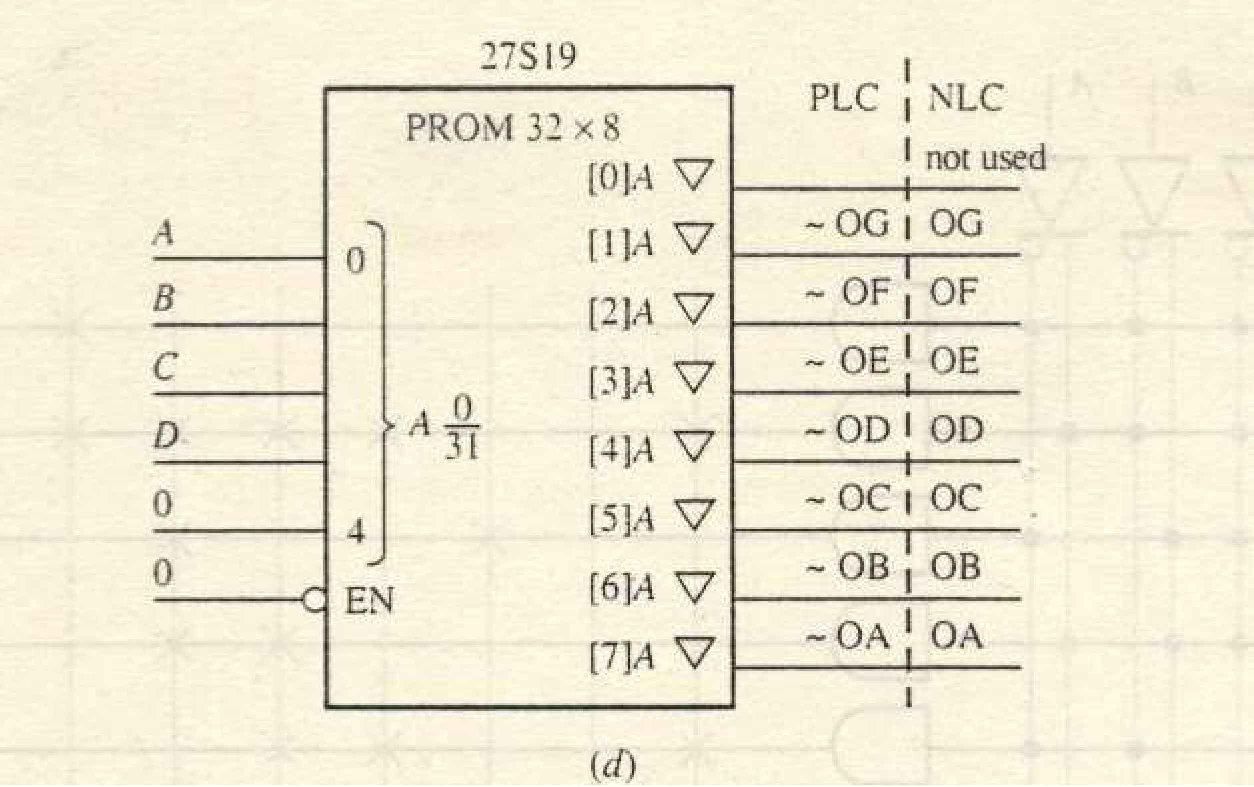


FIGURE E7-11

Solution:


27S19
PROM 32 × 8
PLC
NLC
not used
A
B
C
D
0
0
0
4
EN
A 0/31
[0]A
[1]A
[2]A
[3]A
[4]A
[5]A
[6]A
[7]A
~ OG
OG
~ OF
OF
~ OE
OE
~ OD
OD
~ OC
OC
~ OB
OB
~ OA
OA

(d)

2-Programmable Array Logic (PAL)

2-Programmable Array Logic (PAL)

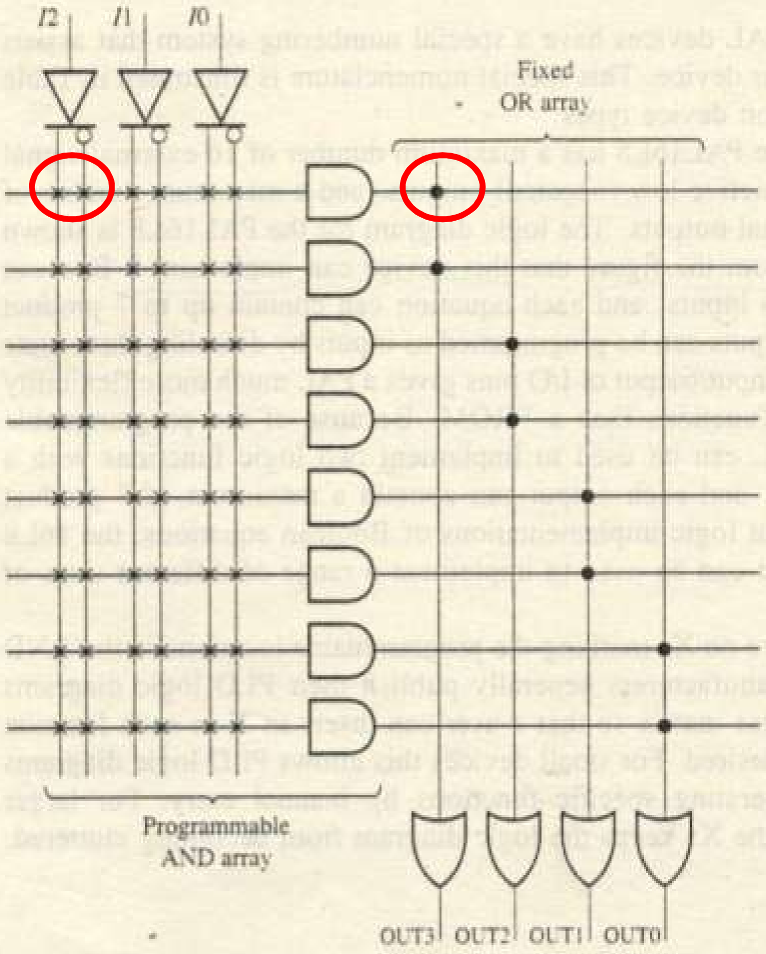


FIGURE 7-23
Simple 3-input, 4-output programmable array logic (PAL).

PAL types

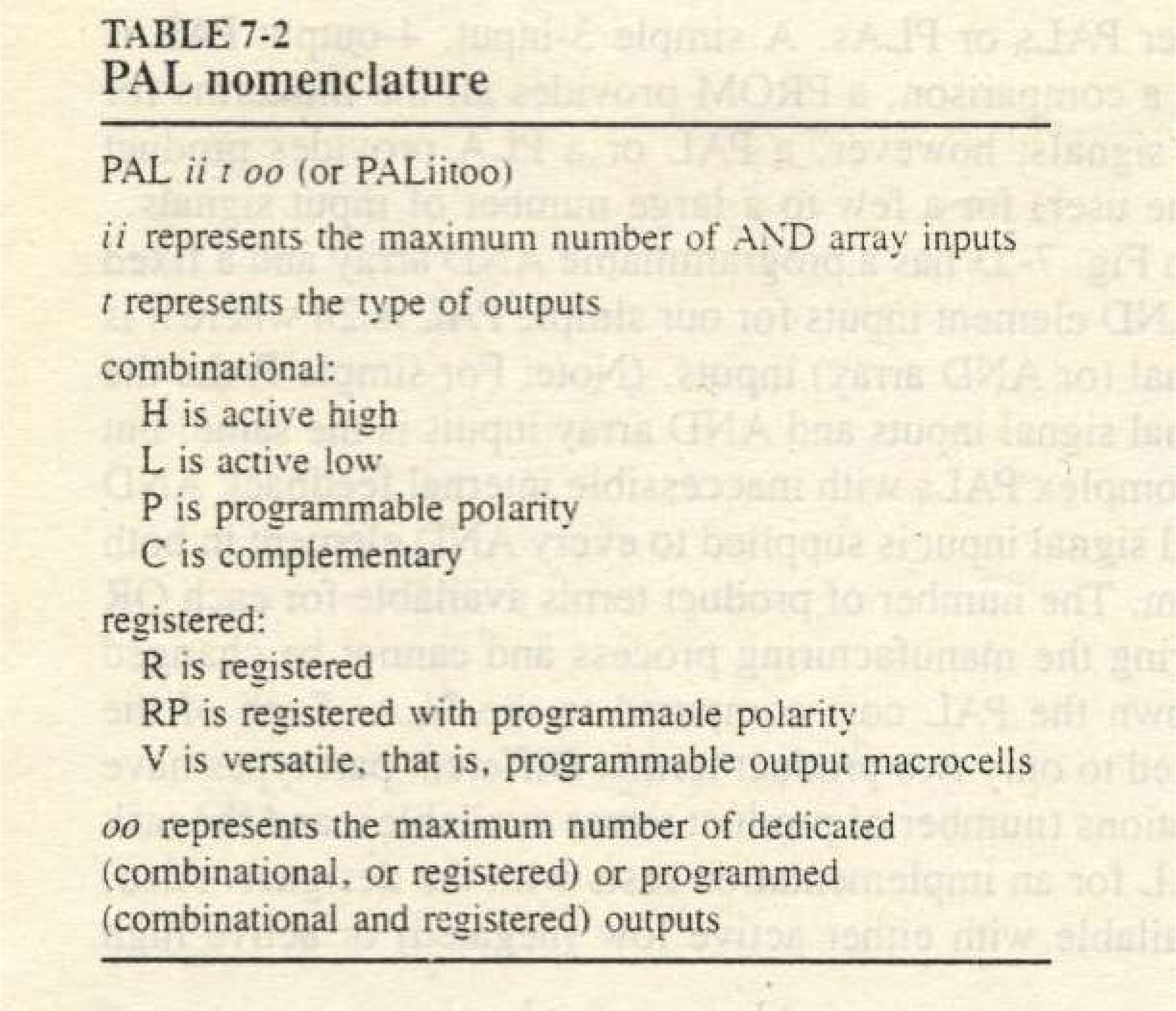


TABLE 7-2
PAL nomenclature

PAL *ii t oo* (or PALiitoo)

ii represents the maximum number of AND array inputs

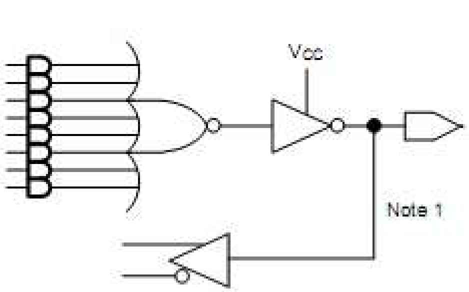
t represents the type of outputs

combinational:
H is active high
L is active low
P is programmable polarity
C is complementary

registered:
R is registered
RP is registered with programmaole polarity
V is versatile, that is, programmable output macrocells

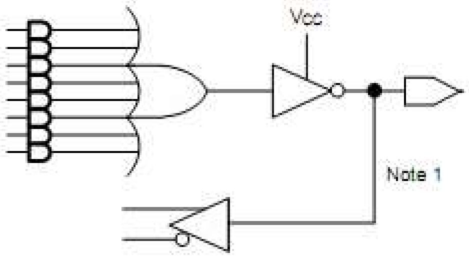
oo represents the maximum number of dedicated (combinational, or registered) or programmed (combinational and registered) outputs

PAL types



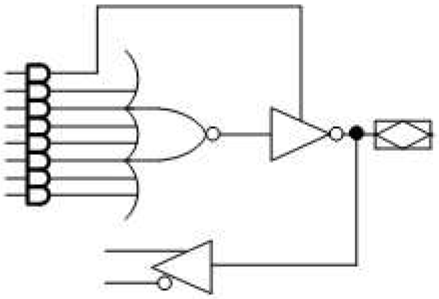
Combinatorial Output Active High

High-Output



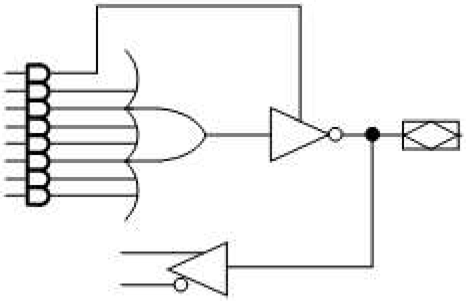
Combinatorial Output Active Low

Low-Output



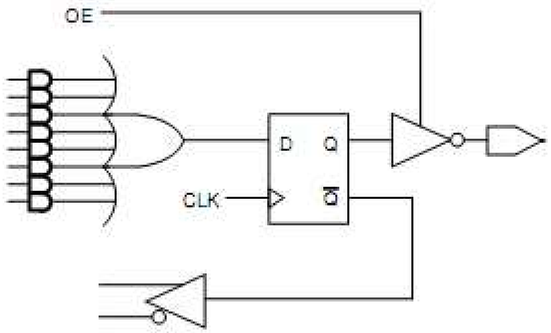
Combinatorial I/O Active High

High-Output/Input



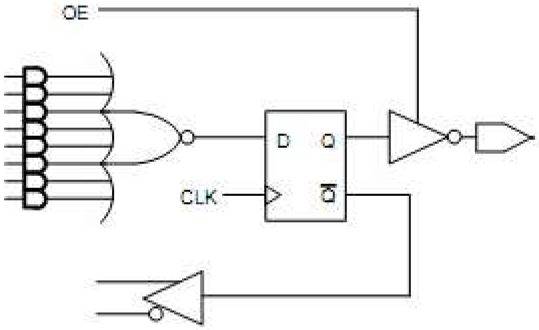
Combinatorial I/O Active Low

Low-Output/Input



Registered Active Low

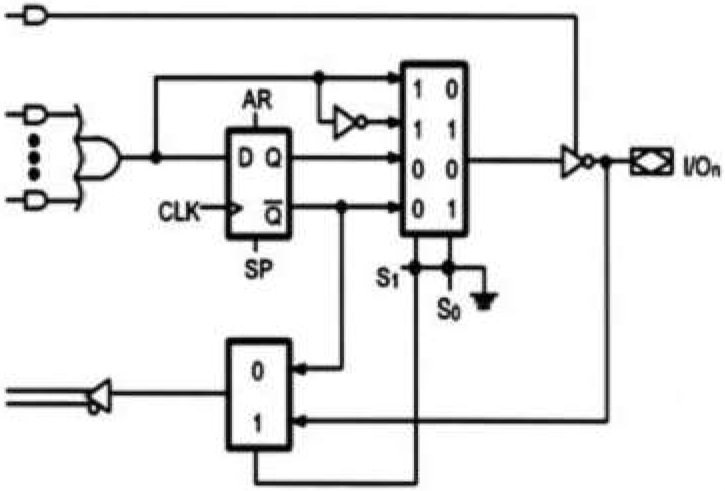
Register-Low-Output



Registered Active High

Register-High-Output

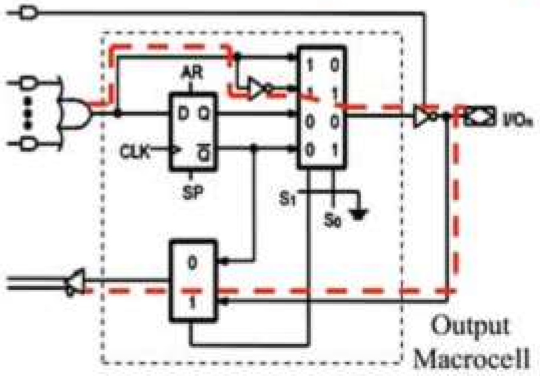
PAL types: versatile



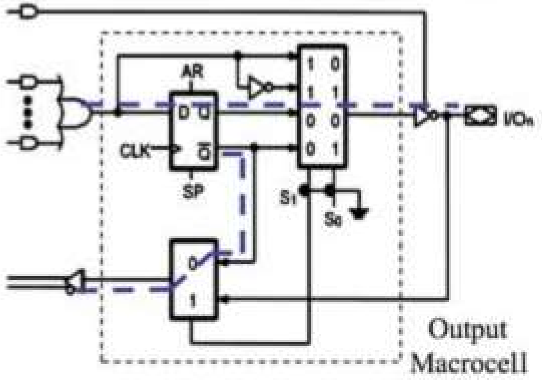
Four configurations:

S_1	S_0	Output Configuration
0	0	Registered / Active Low
0	1	Registered / Active High
1	0	Combinatorial / Active Low
1	1	Combinatorial / Active High

Combinatorial Operation S_1=1

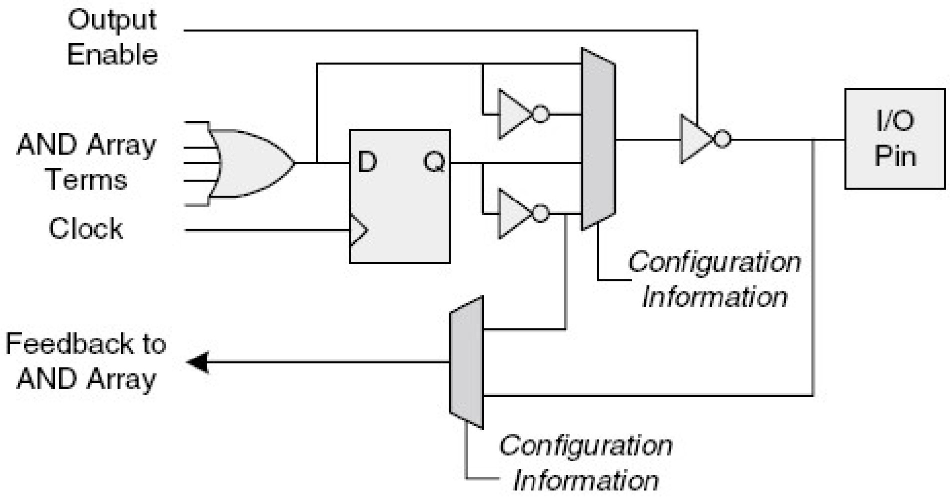


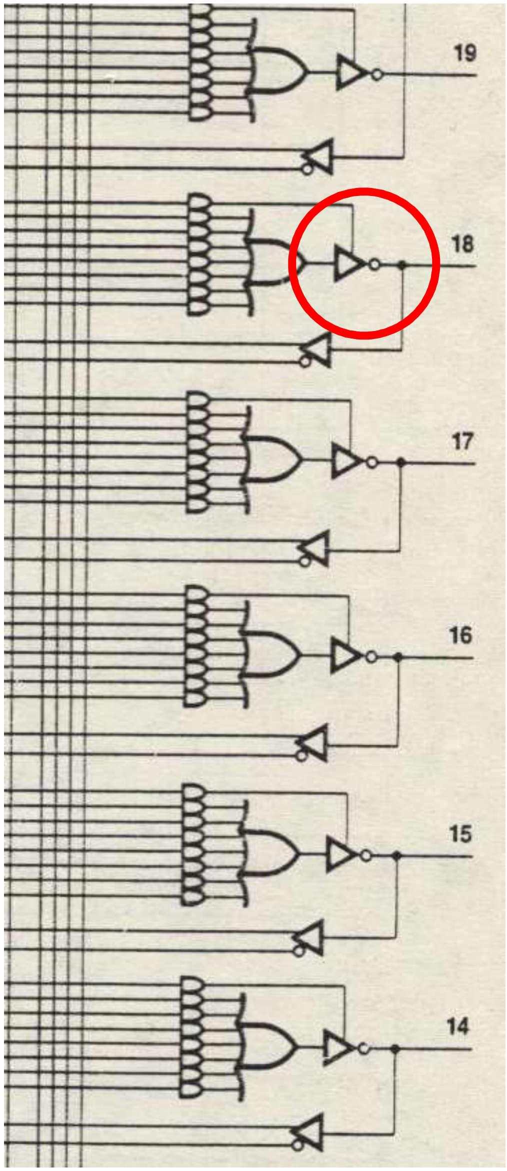
Sequential Operation S_1=0



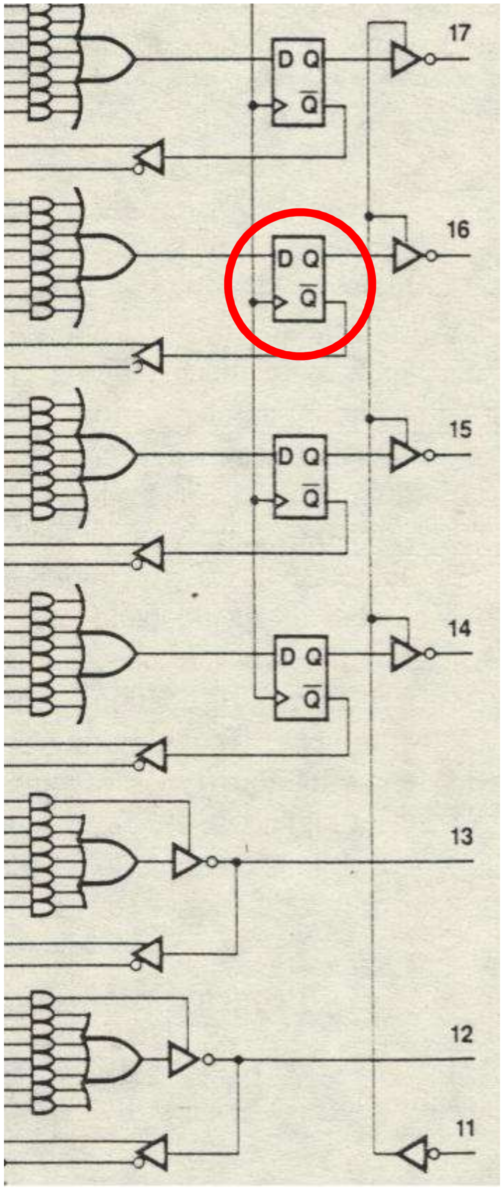
Generic Array Logic (GAL)

- GAL is similar to PAL with output logic microcells, which provide more flexibility.
- GAL can be erased reprogrammed, Instead of using one-time programmable fuse links, GAL use an EEPROM array.



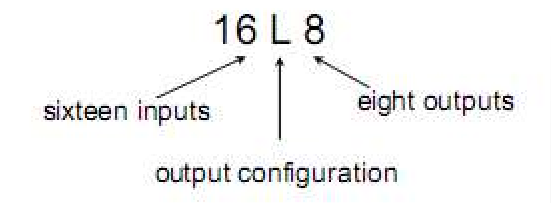


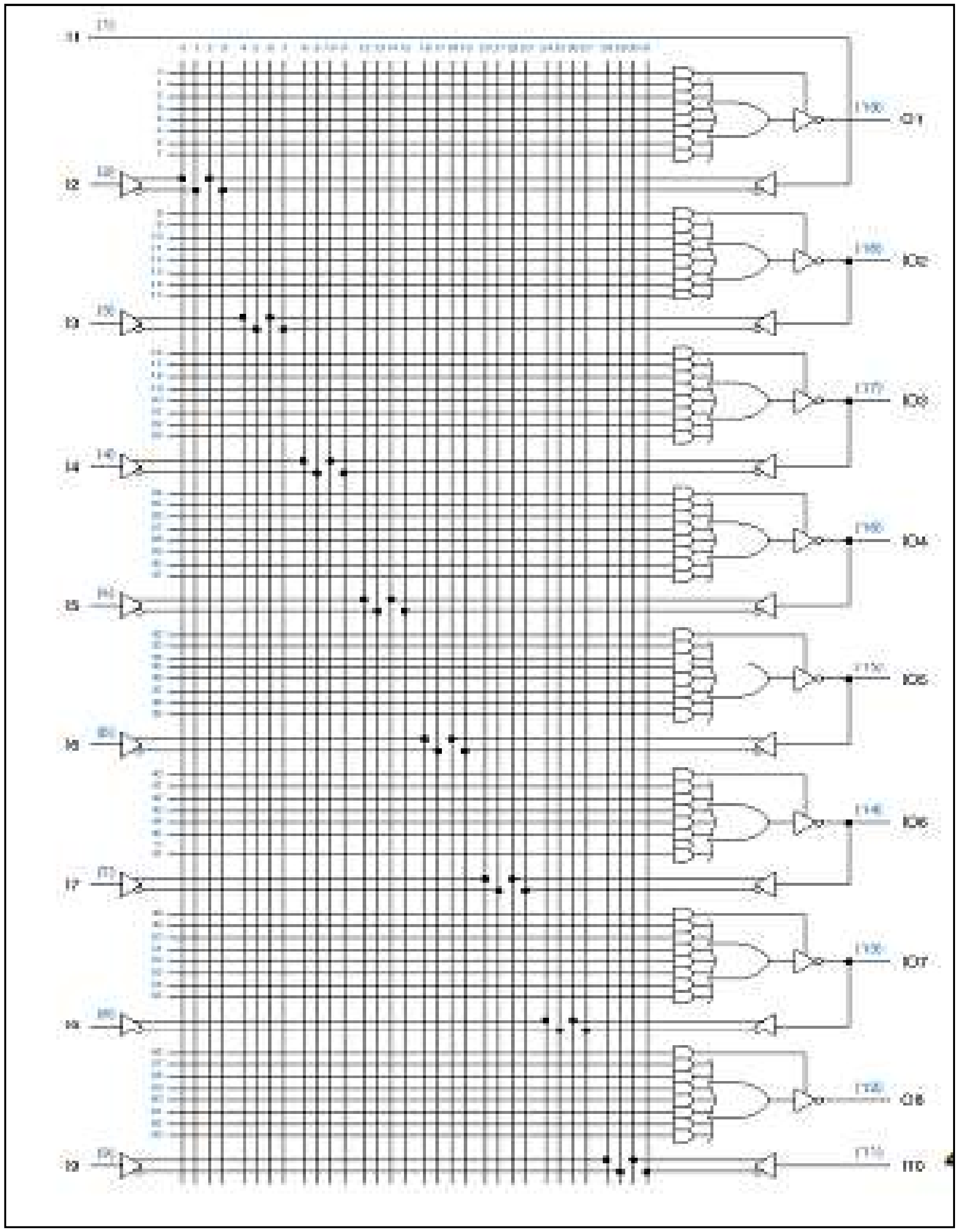
PAL16L4



PAL16R4

PAL16L8


16 L 8
sixteen inputs
output configuration
eight outputs


I1
I2
I3
I4
I5
I6
I7
I8
I9
O1
IO2
IO3
IO4
IO5
IO6
IO7
O8
I10

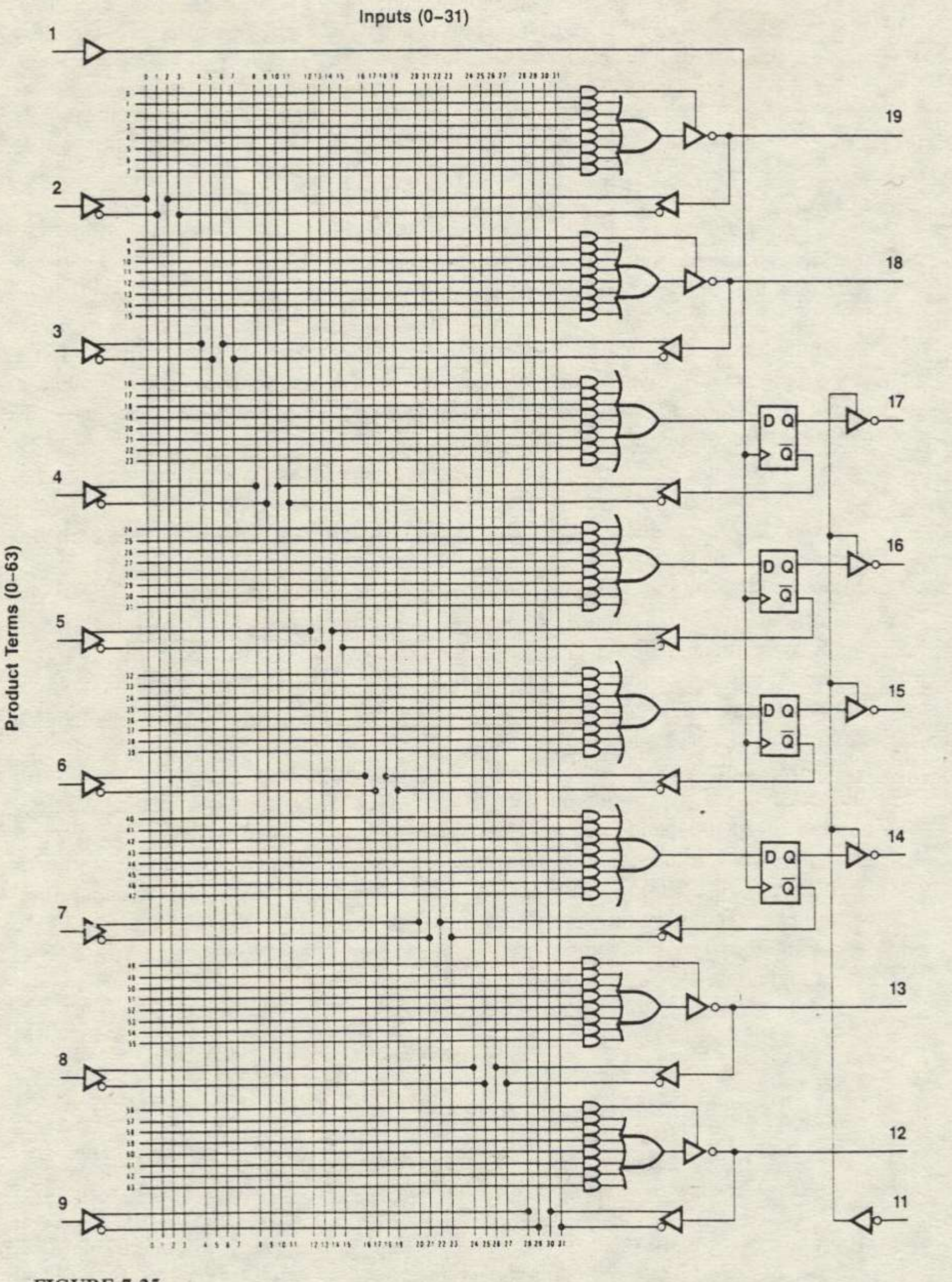


FIGURE 7-25
Logic Diagram PAL16R4. (Courtesy National Semiconductor)

DIP (PLCC) Pinouts

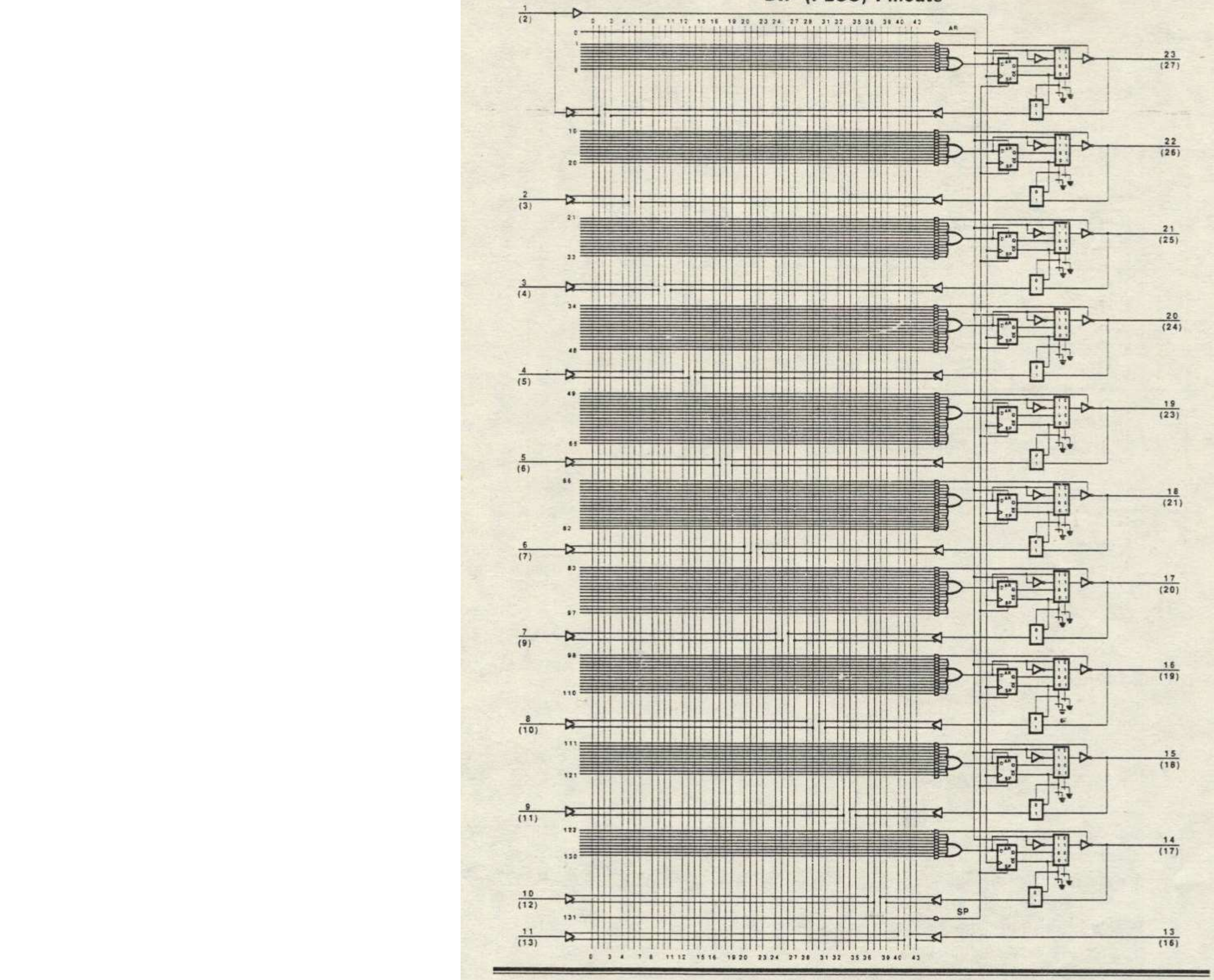
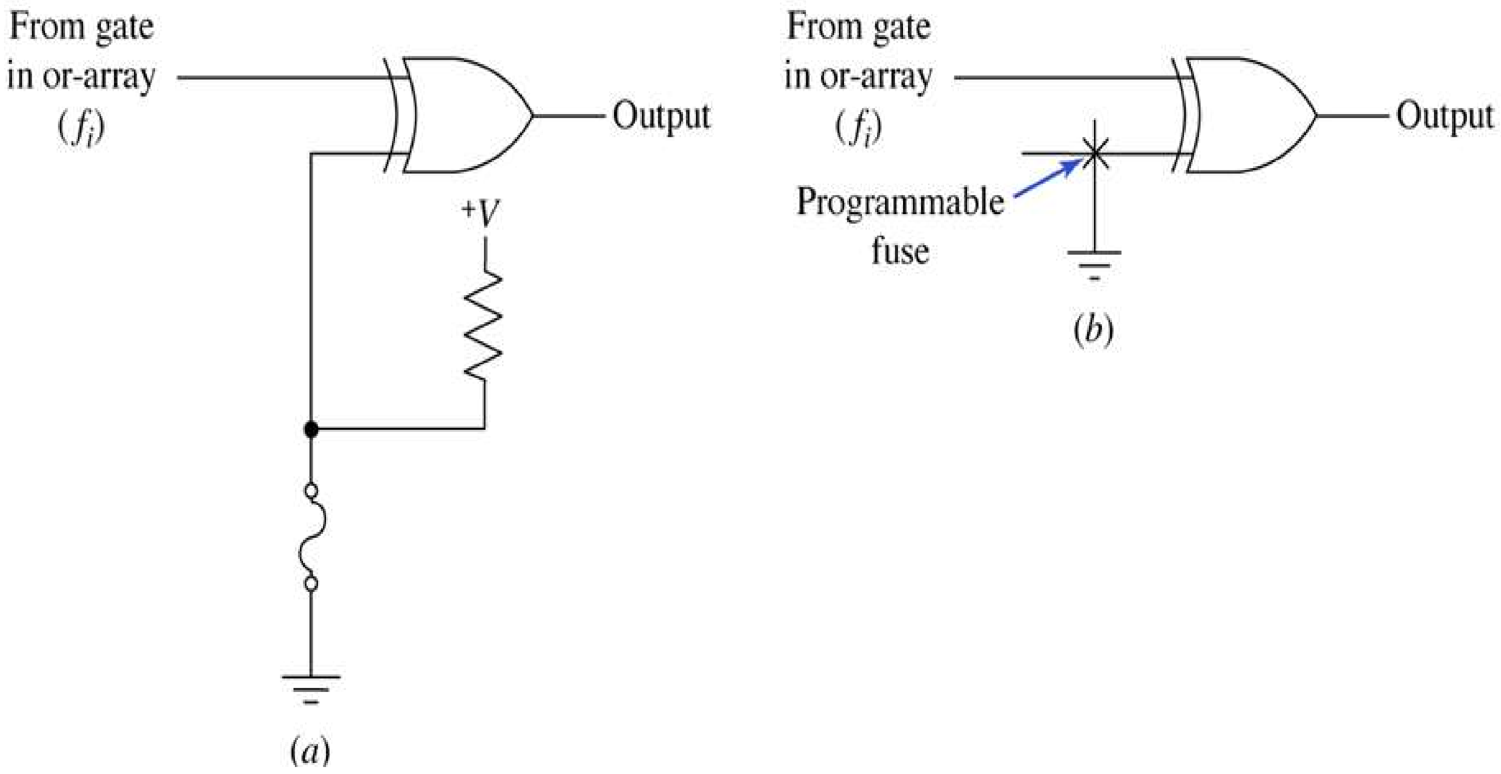
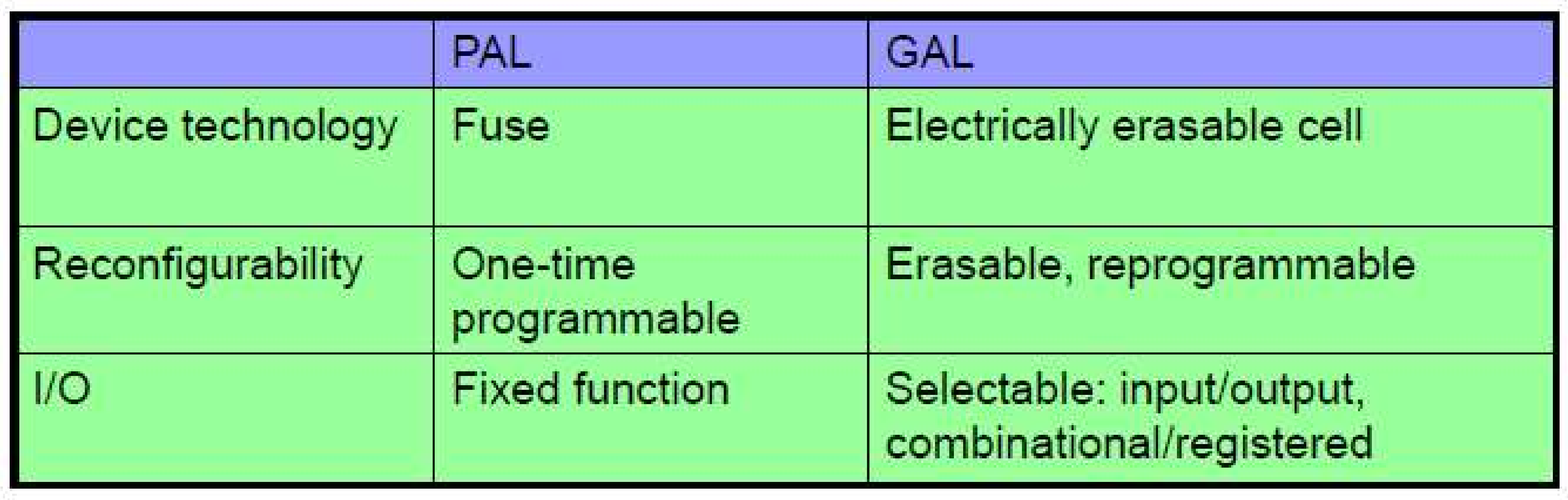


FIGURE 7-26
Logic Diagram PALC22V10. (Copyright ©Advanced Micro Devices. Inc., 1988. Reprinted with permission of copyright owner. All rights reserved.)

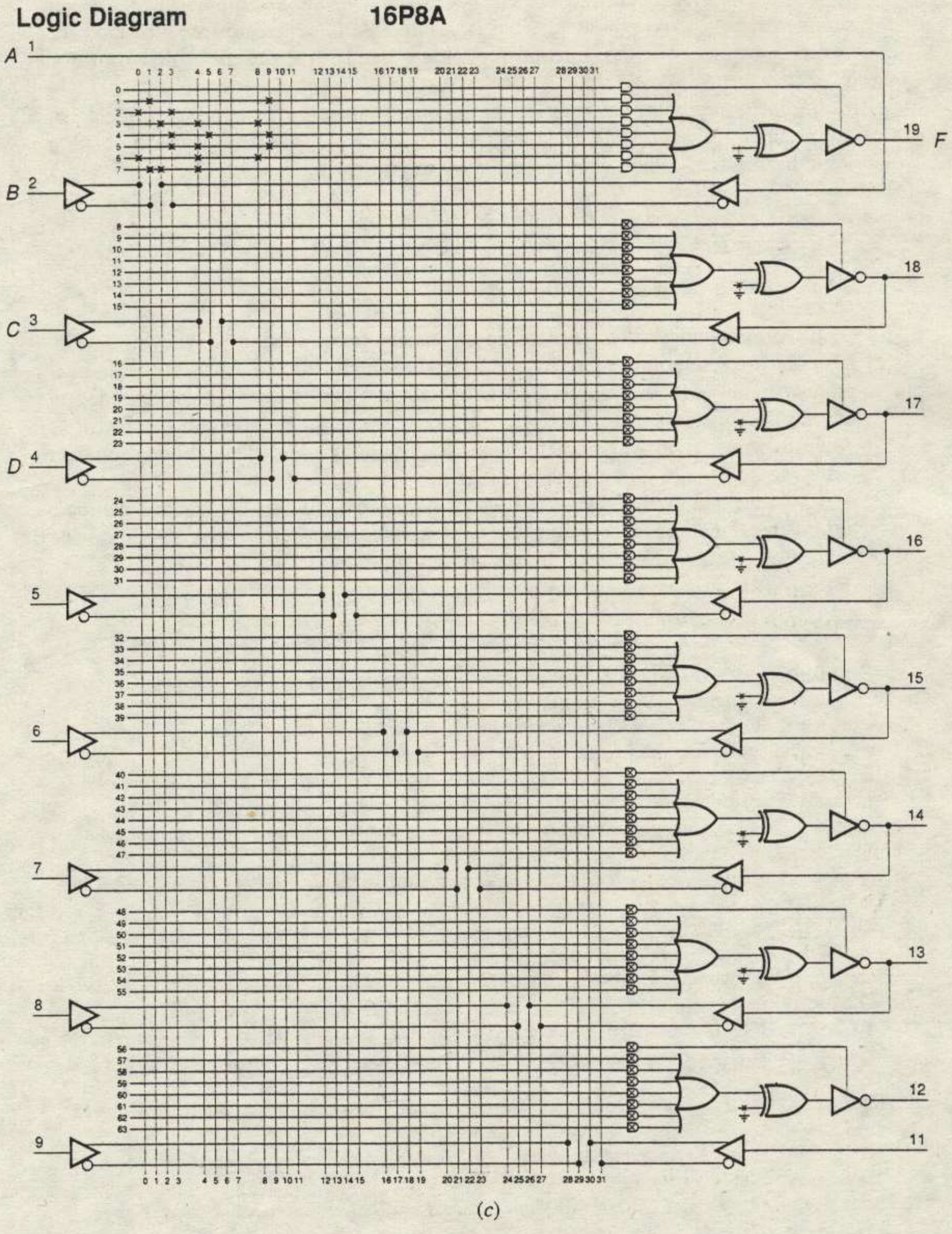
Exclusive-or-gate with a programmable fuse



(*a*) Circuit diagram. (*b*) Symbolic representation.



	PAL	GAL
Device technology	Fuse	Electrically erasable cell
Reconfigurability	One-time programmable	Erasable, reprogrammable
I/O	Fixed function	Selectable: input/output, combinational/registered



(*c*)

FIGURE E7-17 (*continued*)
(Copyright ©Advanced Micro Devices, Inc., 1988. Adapted with permission of copyright owner. All rights reserved.)

PAL Example1:

- Given functions w, x, y, and z. Implement with one PAL4H4.
- Given: Sum of Minterms... After Simplification
- w(A,B,C,D) = Σm(2,12,13)

  w = ABC' + A'B'CD'
- x(A,B,C,D) = Σm(7,8,9,10,11,12,13,14,15)

  x = A + BCD
- y(A,B,C,D) = Σm(0,2,3,4,5,6,7,8,10,11,15)

  y = A'B + CD + B'D'
- z(A,B,C,D) = Σm(1,2,8,12,13)

  z = w + AC'D' + A'B' C'D

W(A,B,C,D) = ∑ (2,12,13) =ABC'+A'B'CD'

CD			C=1	
AB	C'D	C'D	CD	CD'
A'B'	0	0	0	1
A'B	0	0	0	0
AB	1	1	0	0
AB'	0	0	0	0

B=1

A= 1

D=1

X(A,B,C,D) = ∑ (7,8,9,10,11,12,13,14,15)=A + BCD

CD			C=1	
B	C'D	C'D	CD	CD'
A'B'	0	0	0	0
A'B	0	0	1	0
AB	1	1	1	1
AB'	1	1	1	1

B=1

A= 1

D=1

Y(A,B,C,D) = ∑(0,2,3,4,5,6,7,8,10,11,15)

= A'B + CD + B'D'

CD			C=1	
AB	C'D	C'D	CD	CD'
A'B'	1	0	1	1
A'B	1	1	1	1
AB	0	0	1	0
AB'	1	0	1	1

B=1

A= 1

D=1

Z(A,B,C,D) = ∑(1,2,8,12,13)

= ABC' + A'B'CD'+ AC'D' +A'B'C'D'

= W + AC'D' + A'B'C'D

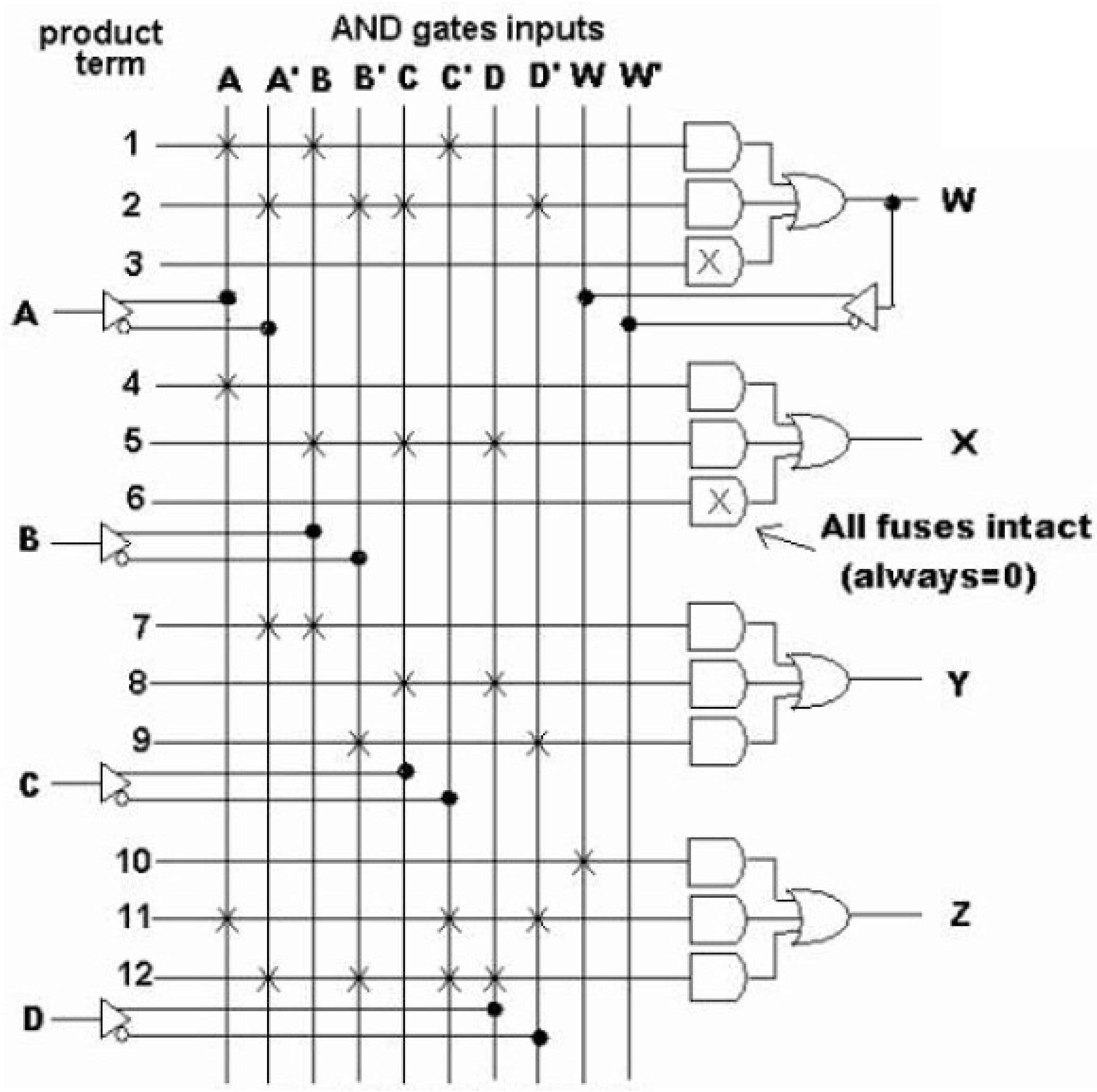
CD			C=1	
AB	C'D	C'D	CD	CD'
A'B'	0	1	0	1
A'B	0	0	0	0
AB	1	1	0	0
AB'	1	0	0	0

B=1

A= 1

D=1

Solution:

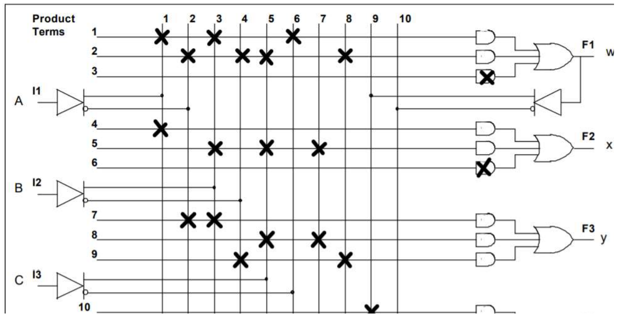


All fuses intact (always=0)

ABC' + A'B'CD'

A + BCD

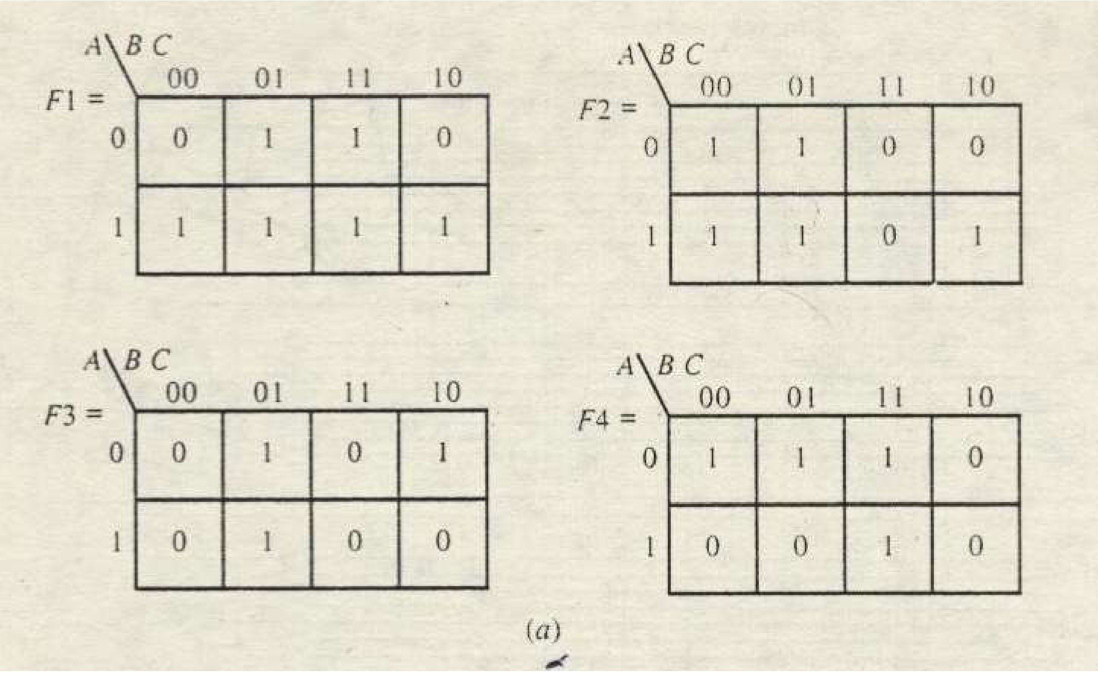
A'B + CD + B'D'



Example 7-12/

1-Show a simple PAL design for the four functions mapped in Fig. E7-12a. Use the simple 3-input, 4-output PAL in Fig. 7-23 if the equations will fit. If the equations will not fit, draw a PAL large enough.

2- Generate the fuse-map information for the equations manually. Assume that all the signals are positive logic signals, and that each signal is available in its true form (uncopemented form).2.Use a PAL16L8 to implement the four functions mapped in Fig. 7-12a. Generate the fuse-map information for the equations manually.



F1 =

A\BC	00	01	11	10
0	0	1	1	0
1	1	1	1	1

F2 =

A\BC	00	01	11	10
0	1	1	0	0
1	1	1	0	1

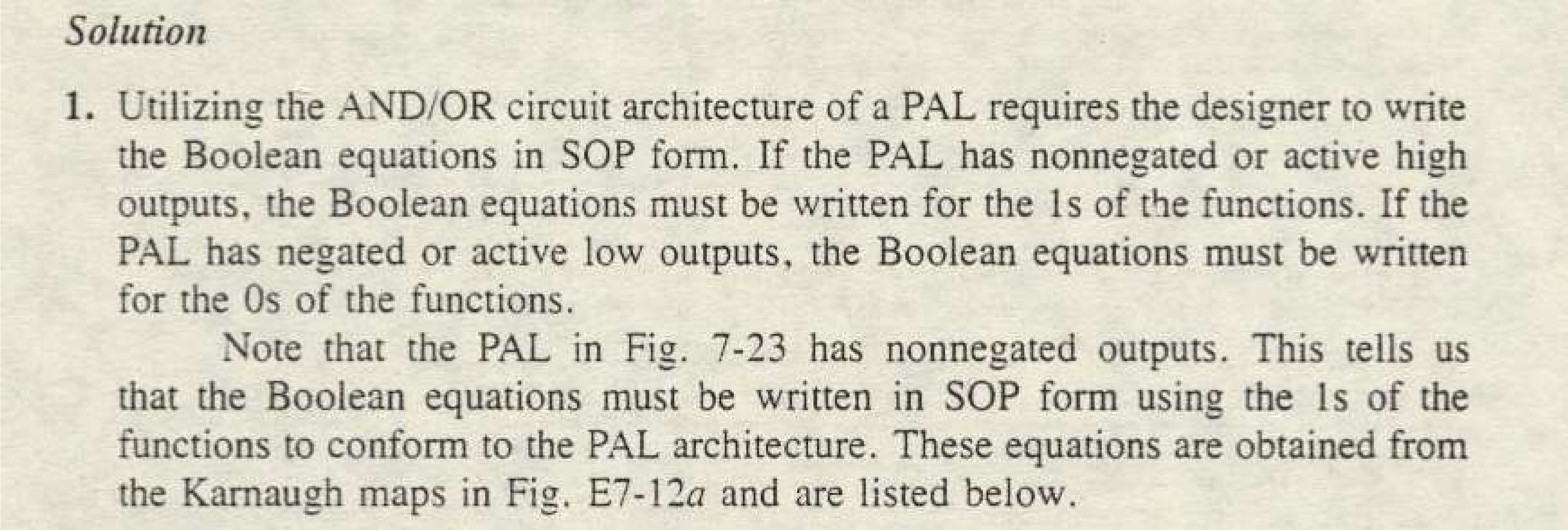
F3 =

A\BC	00	01	11	10
0	0	1	0	1
1	0	1	0	0

F4 =

A\BC	00	01	11	10
0	1	1	1	0
1	0	0	1	0

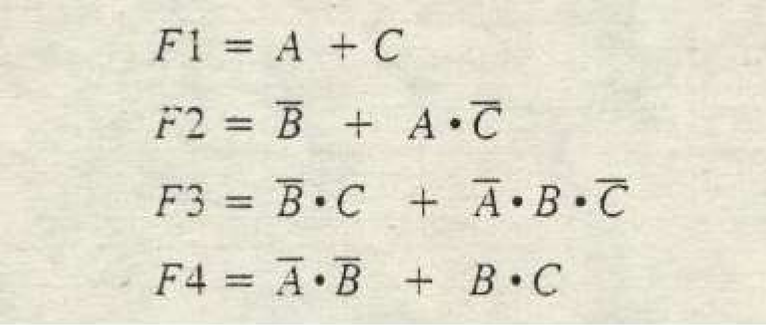
(a)



Solution

1. Utilizing the AND/OR circuit architecture of a PAL requires the designer to write the Boolean equations in SOP form. If the PAL has nonnegated or active high outputs, the Boolean equations must be written for the 1s of the functions. If the PAL has negated or active low outputs, the Boolean equations must be written for the 0s of the functions.

   Note that the PAL in Fig. 7-23 has nonnegated outputs. This tells us that the Boolean equations must be written in SOP form using the 1s of the functions to conform to the PAL architecture. These equations are obtained from the Karnaugh maps in Fig. E7-12*a* and are listed below.

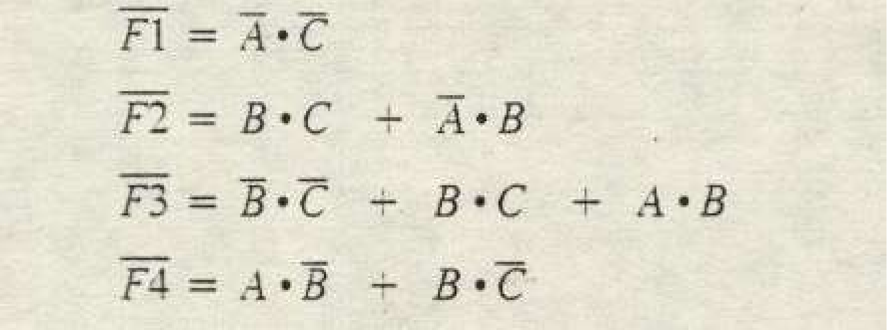


$$F1 = A + C$$

$$F2 = \overline{B} + A \cdot \overline{C}$$

$$F3 = \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C}$$

$$F4 = \overline{A} \cdot \overline{B} + B \cdot C$$



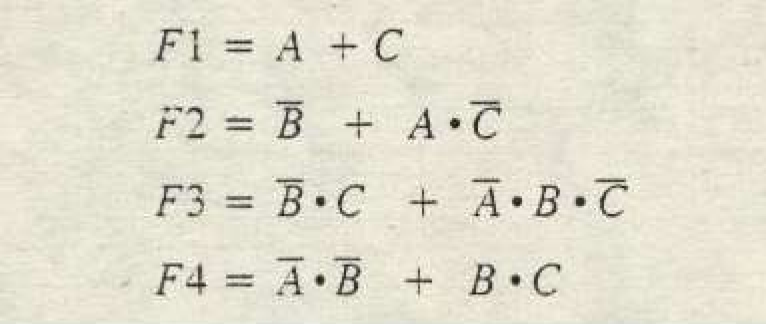
$$\overline{F1} = \overline{A} \cdot \overline{C}$$

$$\overline{F2} = B \cdot C + \overline{A} \cdot B$$

$$\overline{F3} = \overline{B} \cdot \overline{C} + B \cdot C + A \cdot B$$

$$\overline{F4} = A \cdot \overline{B} + B \cdot \overline{C}$$

Solution: a-

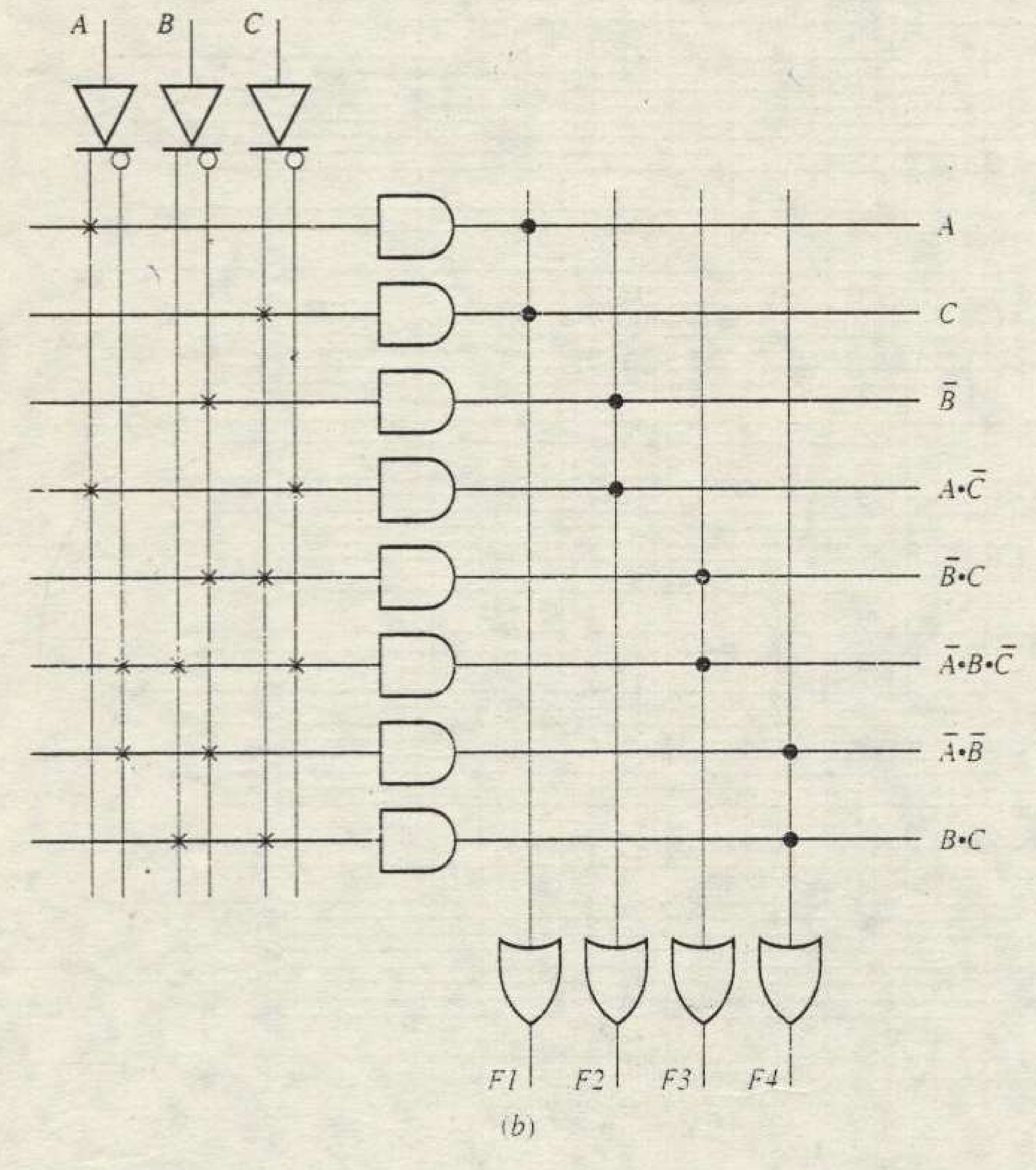


$F1 = A + C$

$F2 = \overline{B} + A \cdot \overline{C}$

$F3 = \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C}$

$F4 = \overline{A} \cdot \overline{B} + B \cdot C$



(b)

FIGURE E7-12

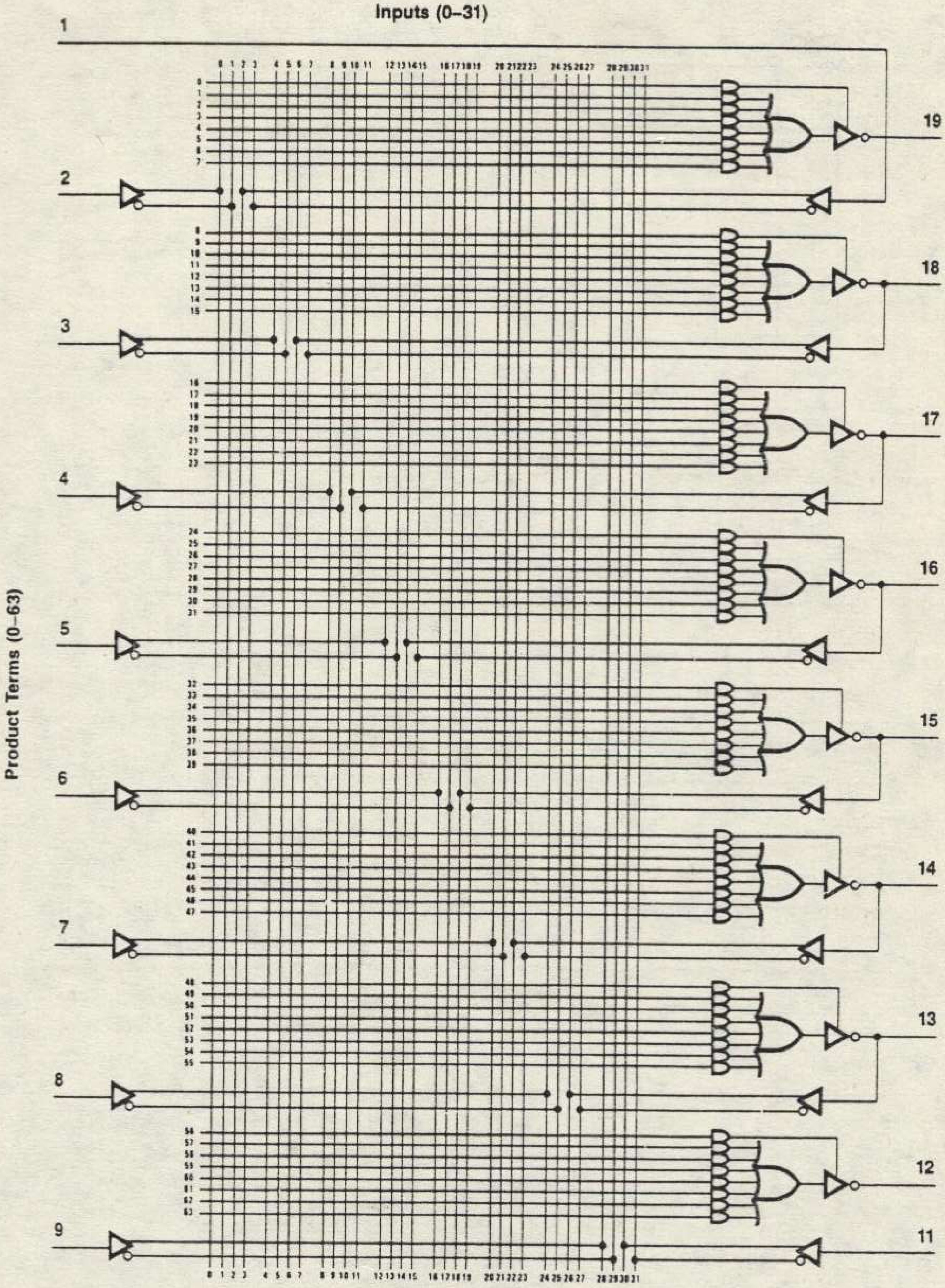


FIGURE 7-24

Logic Diagram PAL16L8. (Courtesy National Semiconductor)

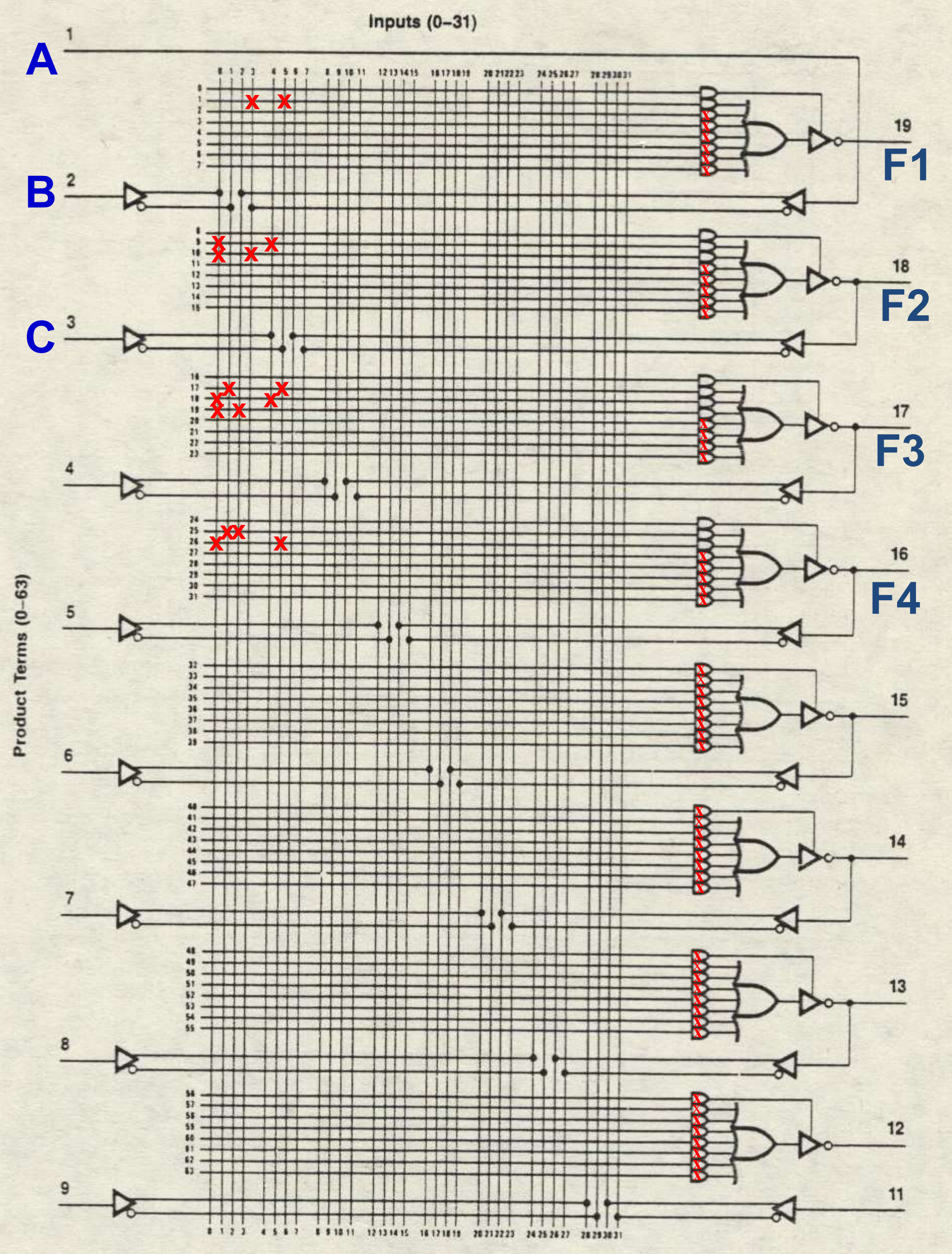
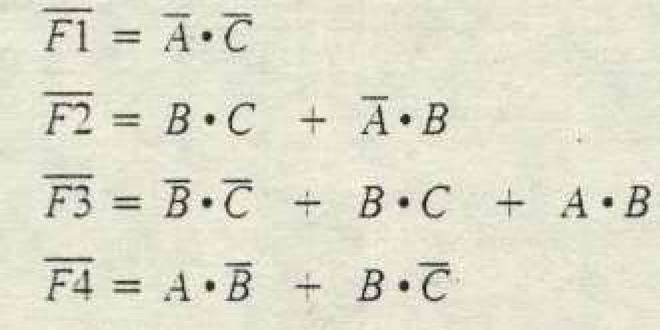


FIGURE 7-24
Logic Diagram PAL16L8. (Courtesy National Semiconductor)



$$\overline{F1} = \overline{A} \cdot \overline{C}$$

$$\overline{F2} = B \cdot C + \overline{A} \cdot B$$

$$\overline{F3} = \overline{B} \cdot \overline{C} + B \cdot C + A \cdot B$$

$$\overline{F4} = A \cdot \overline{B} + B \cdot \overline{C}$$

Example 7-12 (Con.)

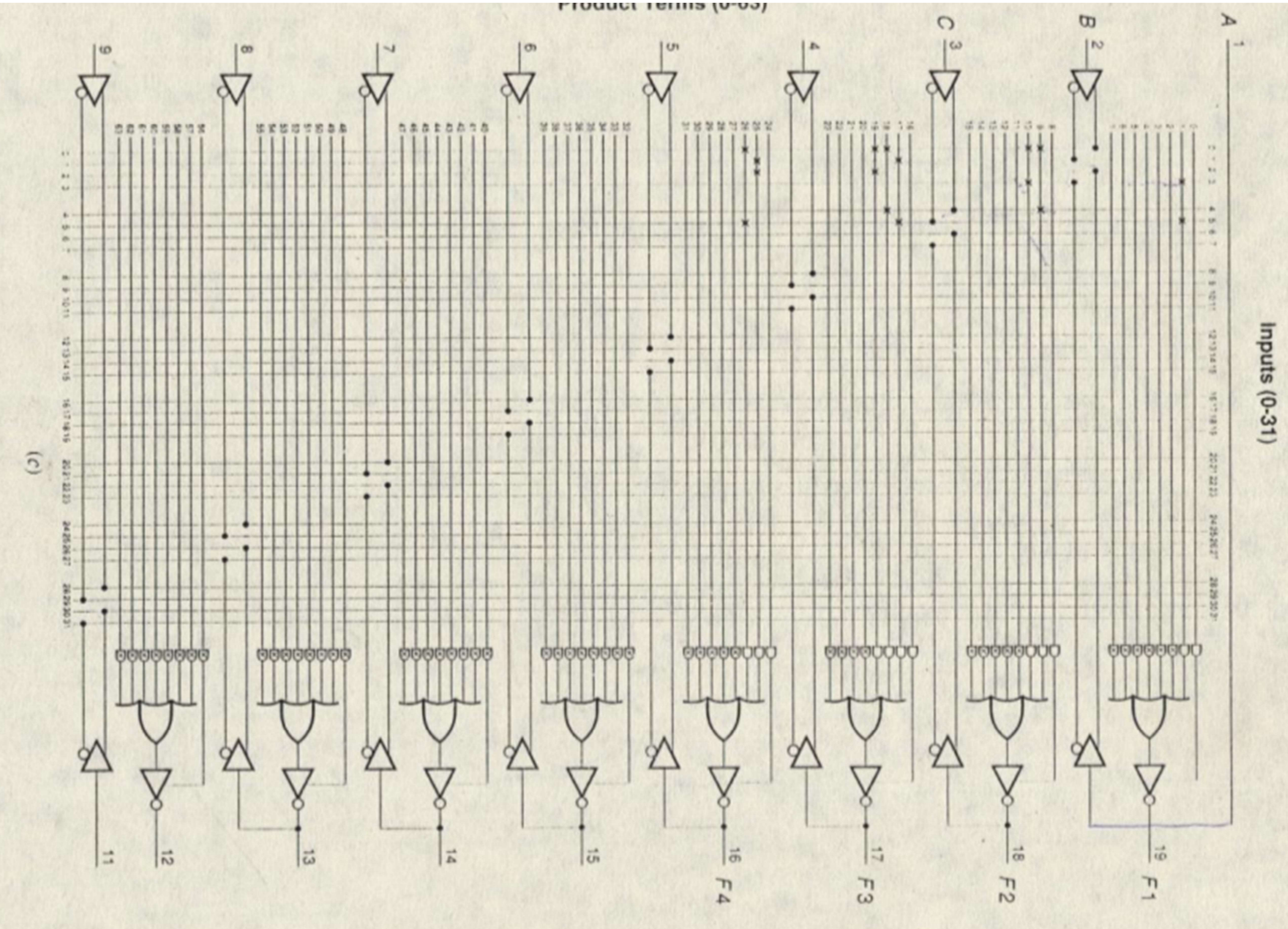


FIGURE E7-12
Logic Diagram PAL16L8. (Logic diagram adapted courtesy of National Semiconductor)

Example 7-12 (Con.)

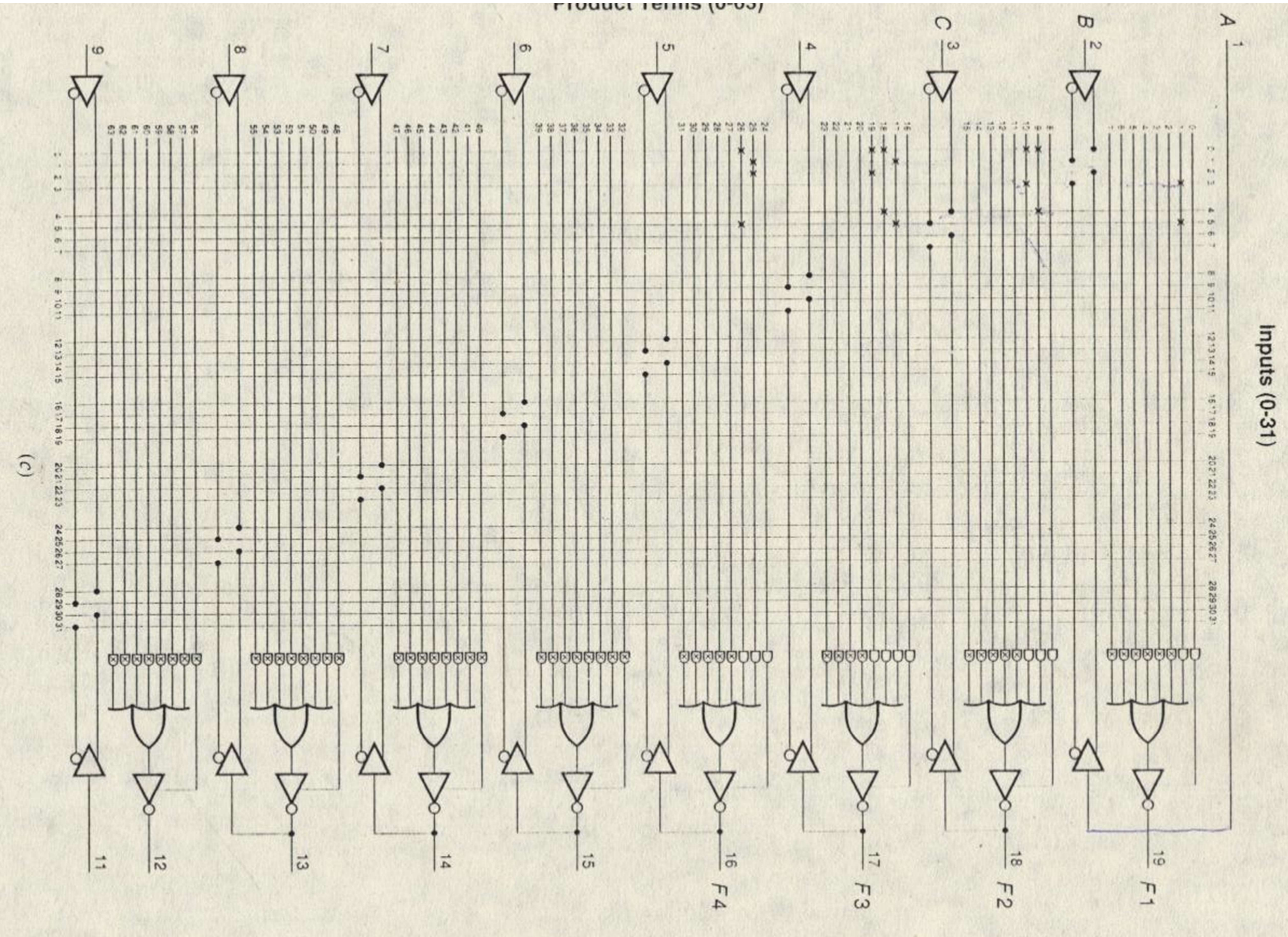


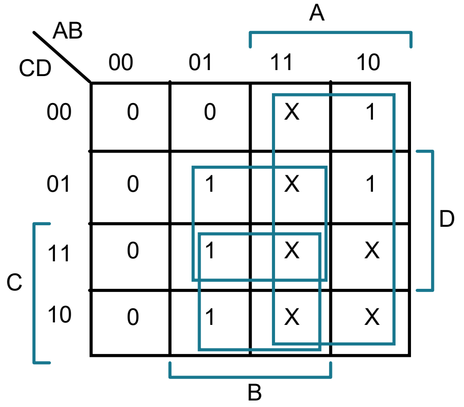
FIGURE E7-12
Logic Diagram PAL16L8. (Logic diagram adapted courtesy of National Semiconductor)

Example2: check
BCD to Gray Code Converter

Truth Table

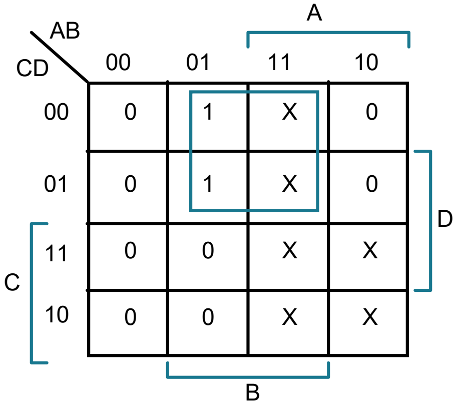
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

K-maps



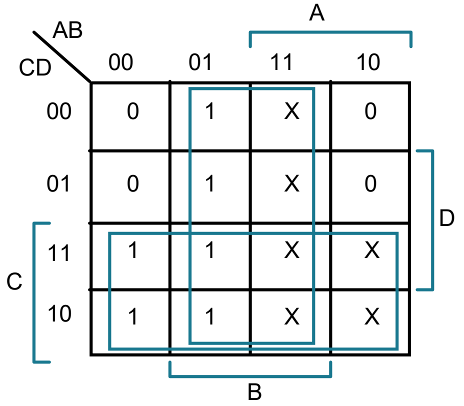
CD \ AB	00	01	11	10
00	0	0	X	1
01	0	1	X	1
11	0	1	X	X
10	0	1	X	X

K-map for W



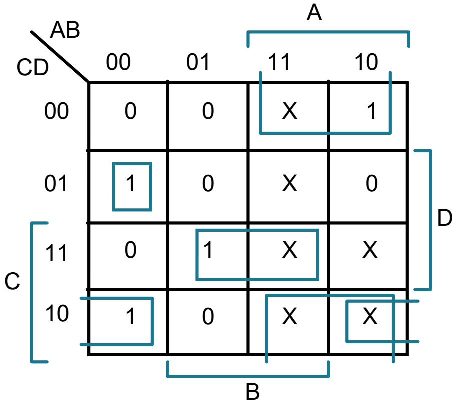
CD \ AB	00	01	11	10
00	0	1	X	0
01	0	1	X	0
11	0	0	X	X
10	0	0	X	X

K-map for X



CD \ AB	00	01	11	10
00	0	1	X	0
01	0	1	X	0
11	1	1	X	X
10	1	1	X	X

K-map for Y



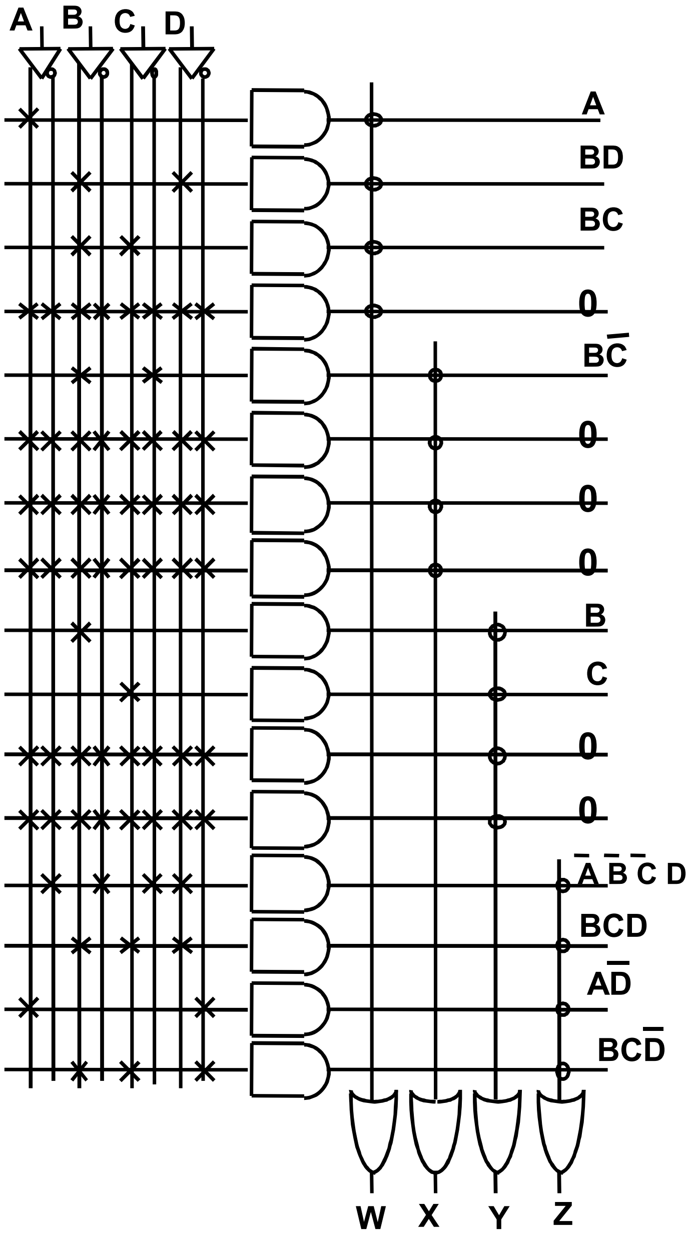
CD \ AB	00	01	11	10
00	0	0	X	1
01	1	0	X	0
11	0	1	X	X
10	1	0	X	X

K-map for Z

Minimized Functions:

W = A + B D + B C
X = B C'
Y = B + C
Z = A'B'C'D + B C D + A D' + B' C D'

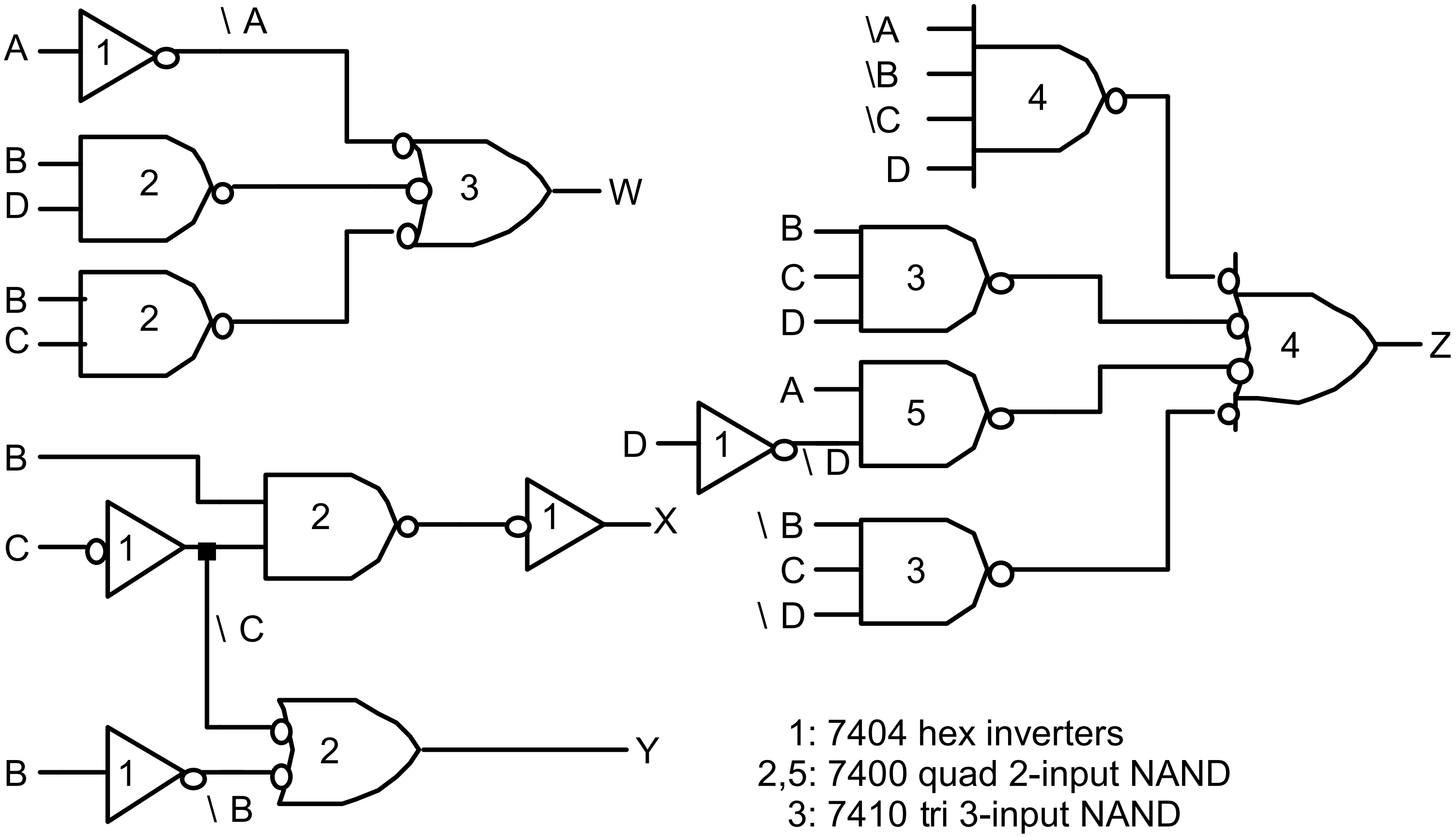
Solutions:-



W = A + B D + B C
X = B C'
Y = B + C
Z = A'B'C'D + B C D + A D' + B' C D'

4 product terms per each OR gate

Code Converter Discrete Gate Implementation



1: 7404 hex inverters
2,5: 7400 quad 2-input NAND
3: 7410 tri 3-input NAND
4: 7420 dual 4-input NAND

4 SSI Packages vs. 1 PLA/PAL Package!

HW1:

Design a magnitude comparator for the following system.

A-Show a simple PAL design for the four functions mapped. Use the simple 4-input, 4-output PAL .

B-Use a PAL16L8 to implement the four functions (Generate the fuse-map).

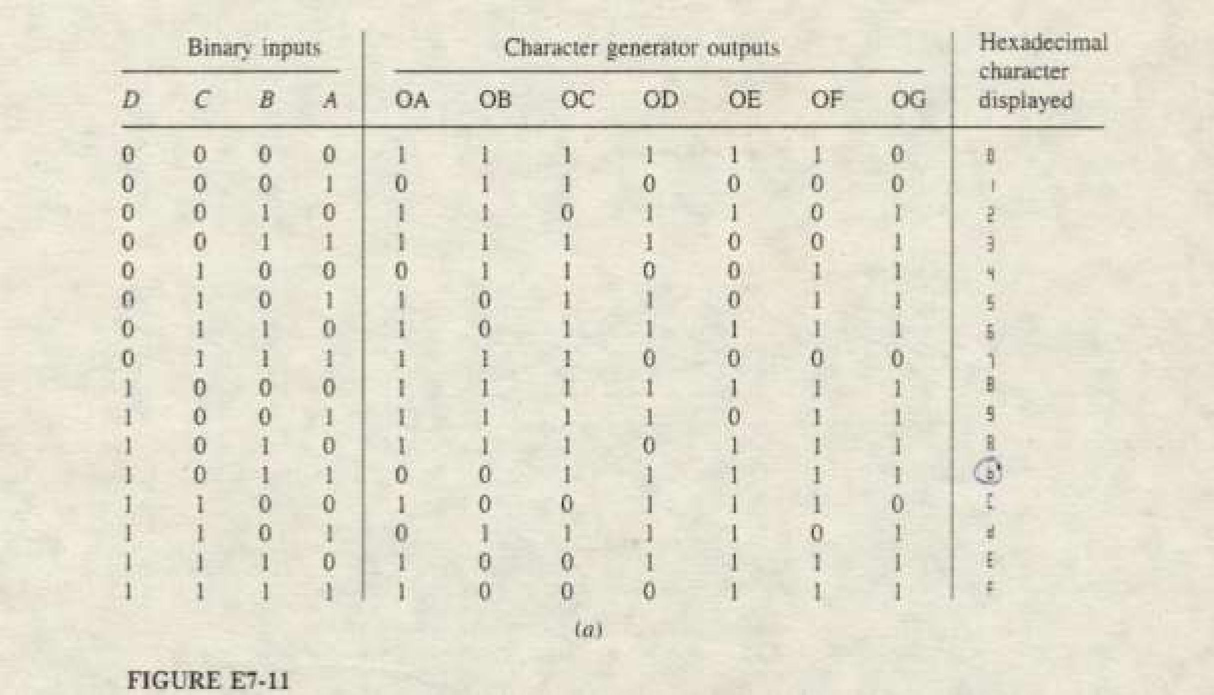
A	B	C	D	EQ	NE	LT	GT
0	0	0	0	1	0	0	0
0	0	0	1	0	1	1	0
0	0	1	0	0	1	1	0
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0	0	1	0	1
1	1	1	1	1	0	0	0

HW2:

- Show a design using PAL16P8
  to implement the binary to hexadecimal character

generator illustrated by the truth table.

- A- Use common cathode 7-segments.
- B- Use common anode 7-segments.

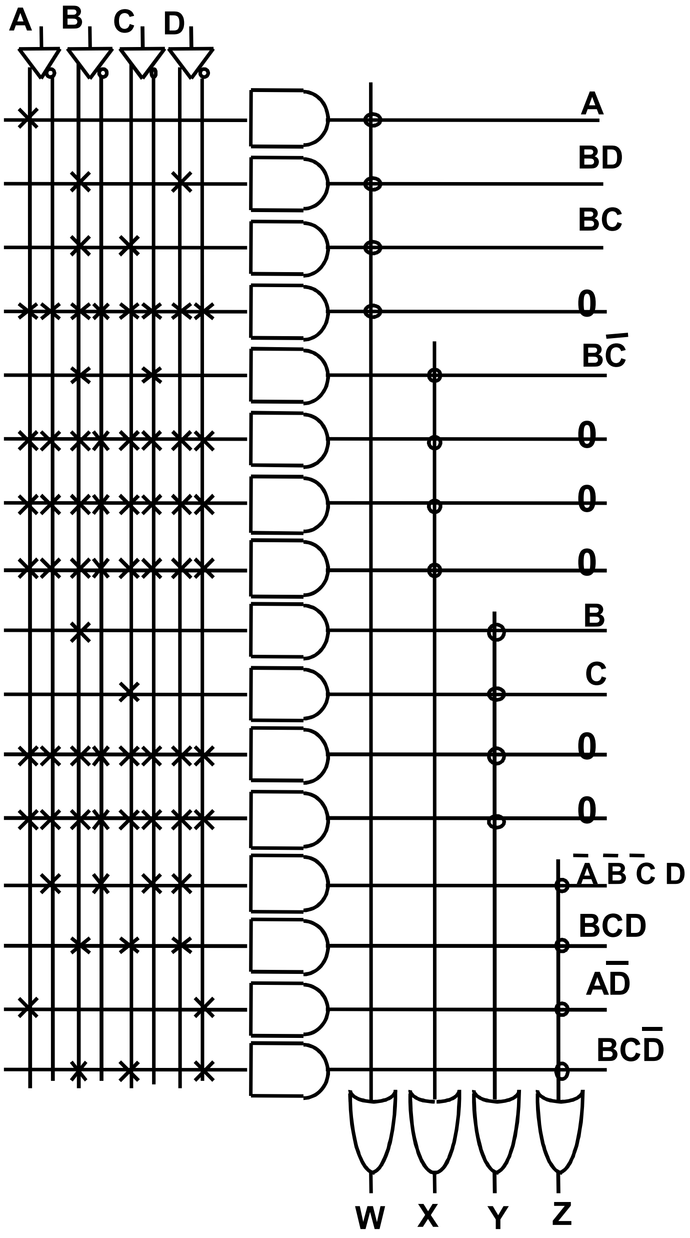


Binary inputs				Character generator outputs							Hexadecimal character displayed
D	*C*	*B*	*A*	OA	OB	OC	OD	OE	OF	OG	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9
1	0	1	0	1	1	1	0	1	1	1	A
1	0	1	1	0	0	1	1	1	1	1	b
1	1	0	0	1	0	0	1	1	1	0	C
1	1	0	1	0	1	1	1	1	0	1	d
1	1	1	0	1	0	0	1	1	1	1	E
1	1	1	1	1	0	0	0	1	1	1	F

(*a*)

FIGURE E7-11

Solutions:-



W = A + B D + B C
X = B C'
Y = B + C
Z = A'B'C'D + B C D + A D' + B' C D'

4 product terms per each OR gate

3-Programmable Logic Array (PLA)

3-Programmable Logic Array (PLA)

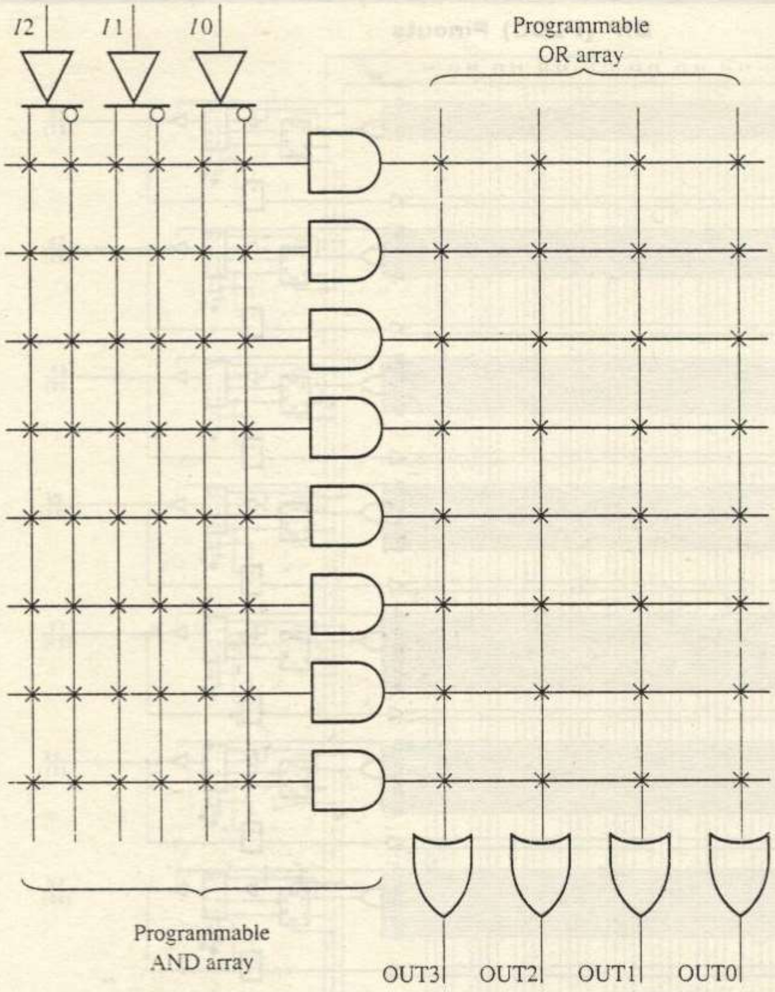


FIGURE 7-27
Simple 3-input, 4-output programmable logic array (PLA).

(PLA)

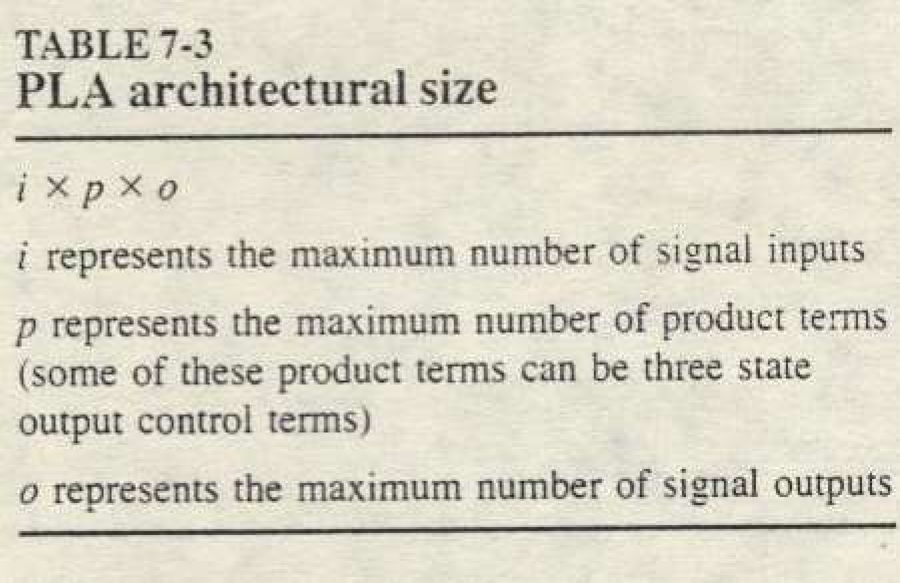


TABLE 7-3
PLA architectural size

i × *p* × *o*

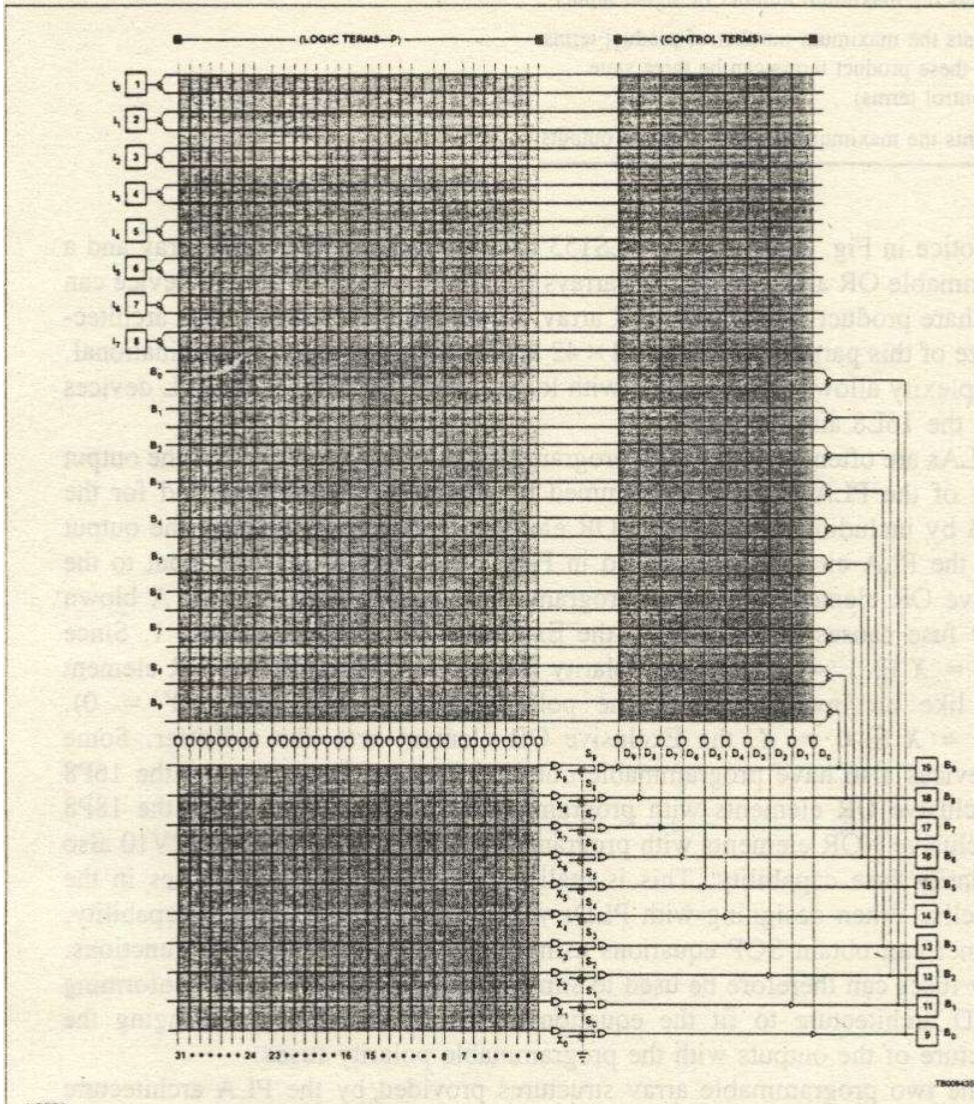
i represents the maximum number of signal inputs

p represents the maximum number of product terms (some of these product terms can be three state output control terms)

o represents the maximum number of signal outputs

Field-Programmable Logic Array (18 × 42 × 10) Pl

FPLA LOGIC DIAGRAM



NOTES:
1. All programmed "AND" gate locations are pulled to logic "1".
2. All programmed "OR" gate locations are pulled to logic "0".
3. ⊙ Programmable connection.

PLAs

Design Example

Multiple functions of A, B, C

F1 = A B C

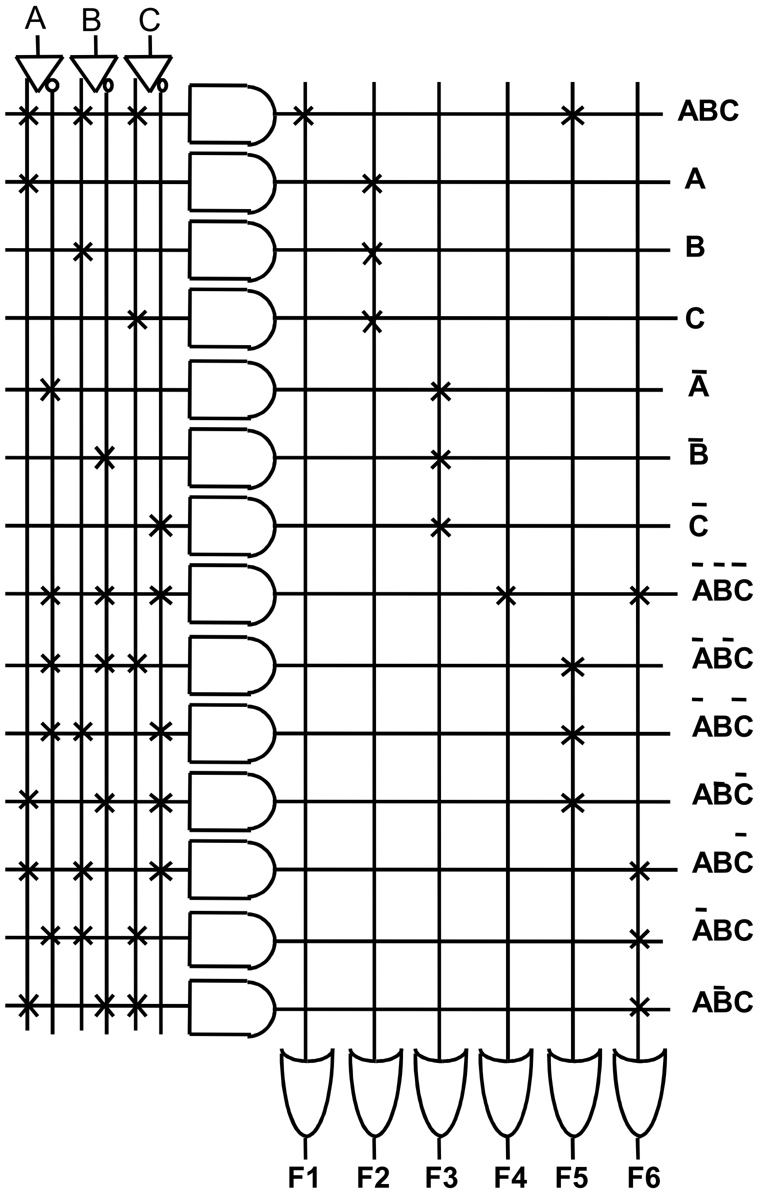
F2 = A + B + C

F3 = $\overline{\text{A B C}}$

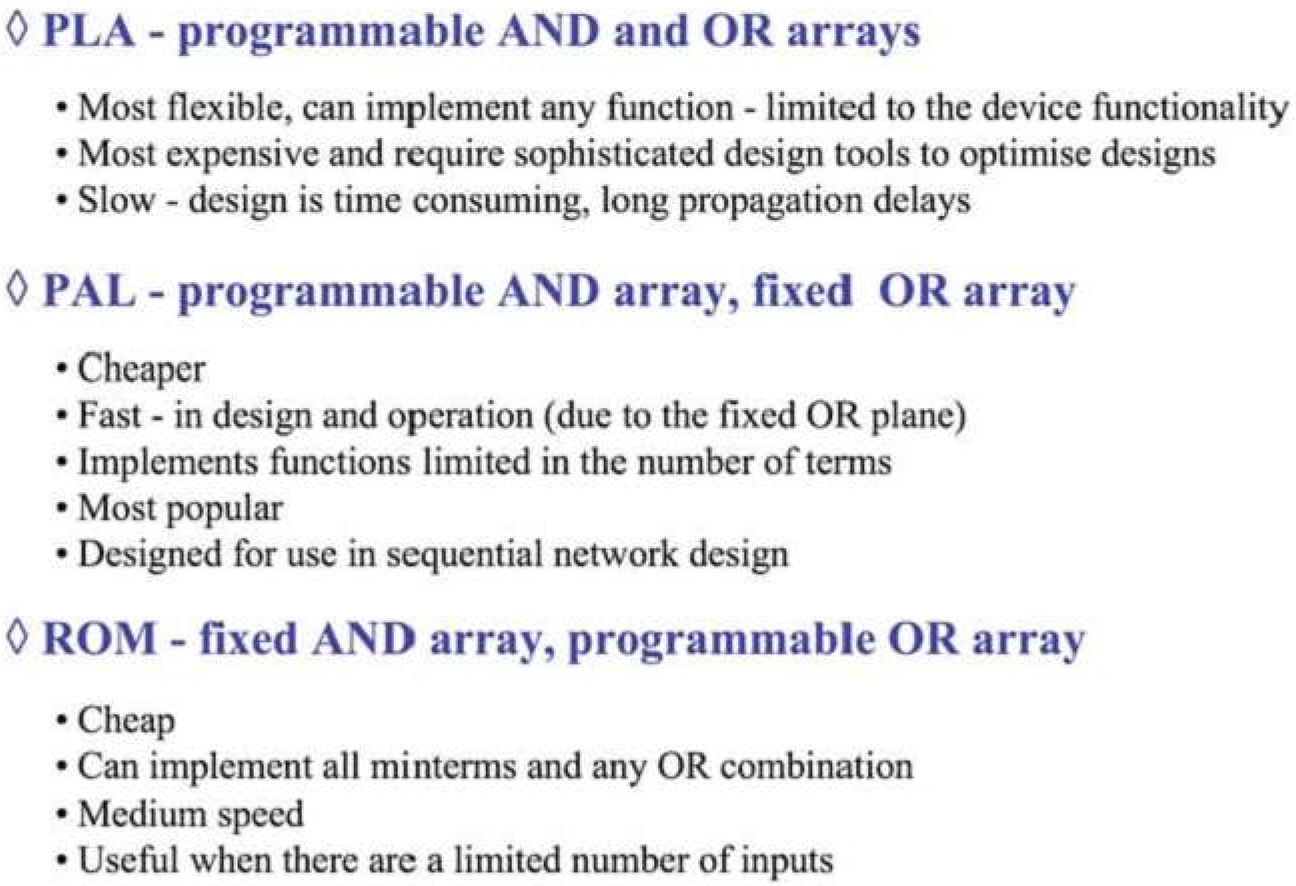
F4 = $\overline{\text{A + B + C}}$

F5 = A xor B xor C

F6 = A xnor B xnor C



PLA, PAL, ROM Overview



◊ **PLA - programmable AND and OR arrays**

- Most flexible, can implement any function - limited to the device functionality
- Most expensive and require sophisticated design tools to optimise designs
- Slow - design is time consuming, long propagation delays

◊ **PAL - programmable AND array, fixed OR array**

- Cheaper
- Fast - in design and operation (due to the fixed OR plane)
- Implements functions limited in the number of terms
- Most popular
- Designed for use in sequential network design

◊ **ROM - fixed AND array, programmable OR array**

- Cheap
- Can implement all minterms and any OR combination
- Medium speed
- Useful when there are a limited number of inputs

PALCE16V8

SG1, selects configuration options for all microcells in the device , two local configuration cells, SL0n and SL1n, select configurations for I/On only. In this case, the cells shown are SL03 and SL13for configurations of I/O3

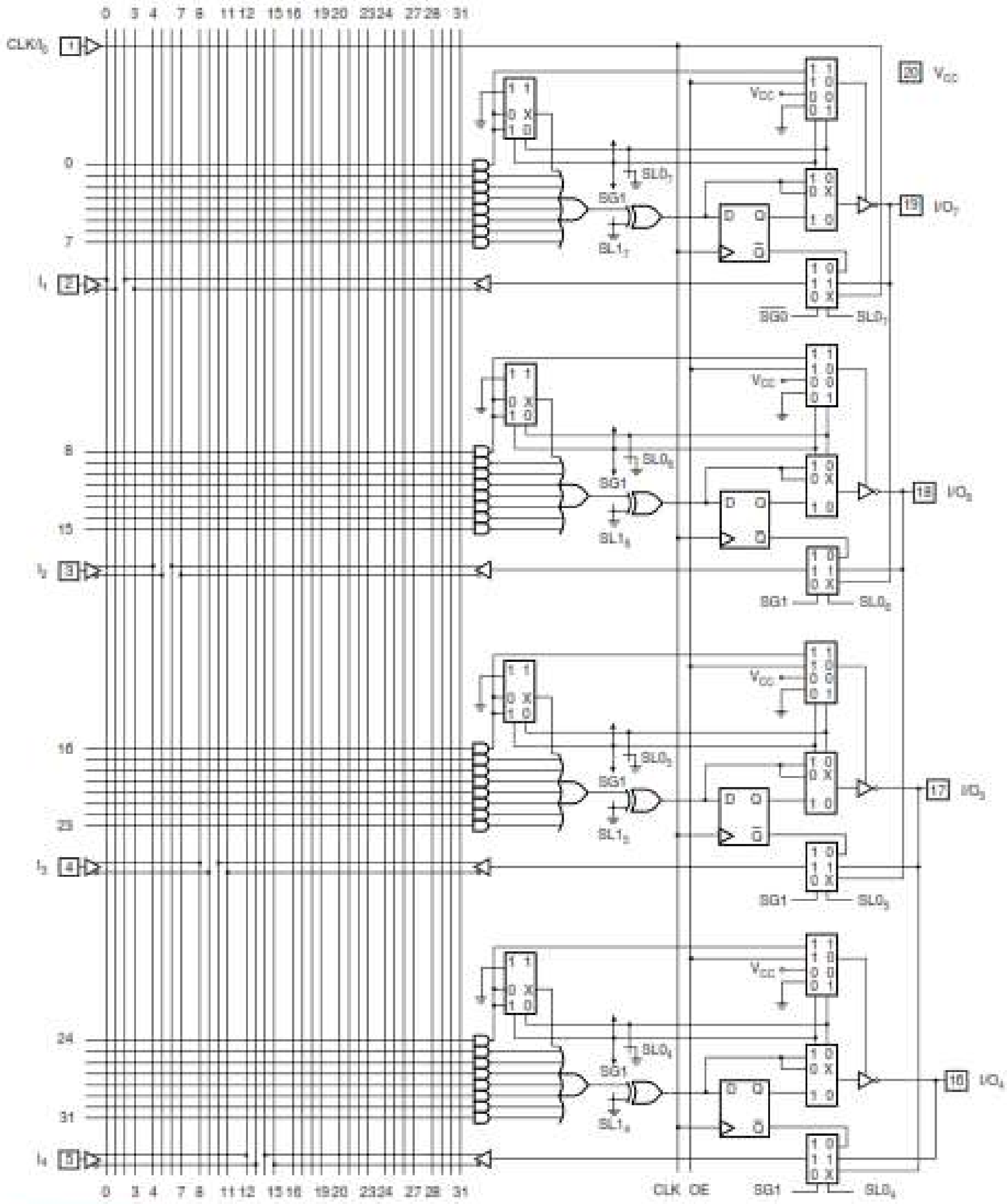


FIGURE 8.17 (a)

PALCE16V8 Logic Diagram (Courtesy of Lattice Semiconductor Corporation)