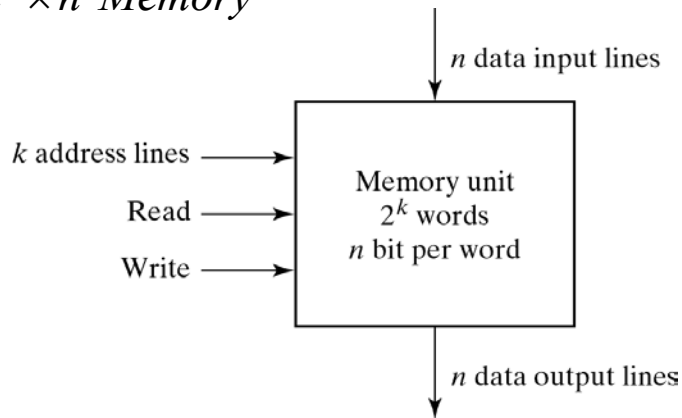


# MEMORY UNIT DEFINITION

$2^k \times n$  Memory



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 7-2 Block Diagram of a Memory Unit

## Basic Memory Definitions

- RAM – Random Access Memory – today it means semiconductor memory that is volatile
- ROM – Read Only Memory – not used much today, except in other forms such as PROM, EPROM, EEPROM - nonvolatile
  - PROM – Programmable Read Only Memory
  - EPROM – Erasable Programmable Read Only Memory
  - EEPROM Electrically Erasable Programmable Read Only Memory

# MEMORY UNIT INTERPRETATION

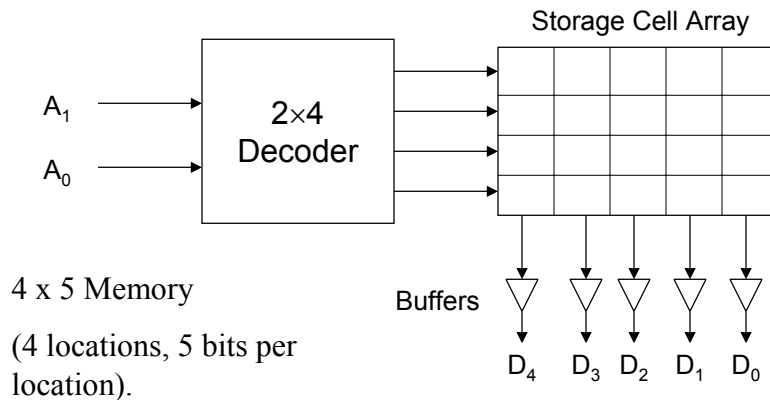
		Memory address		
		Binary	decimal	memory content
$k = 10$		000000000	0	1011010101011101
		000000001	1	1010101110001001
$n = 16$		000000010	2	0000110101000110
		⋮	⋮	⋮
		111111101	1021	1001110100010100
		111111110	1022	0000110100011110
		111111111	1023	1101111000100101

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 7-3 Content of a 1024 × 16 Memory

# MEMORY UNIT ARCHITECTURE

- $n \times m$  Device
  - $(\log_2 n)$  inputs called “address lines”
  - $m$  outputs called “data lines”



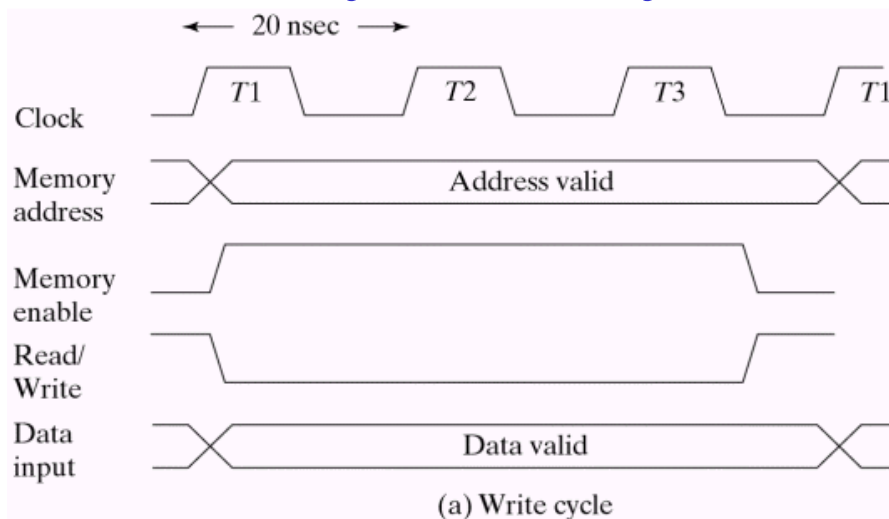
## Memory Parameters

- Access Time
  - Time required to access a word and read it
- Cycle Time
  - Time required to complete a write operation

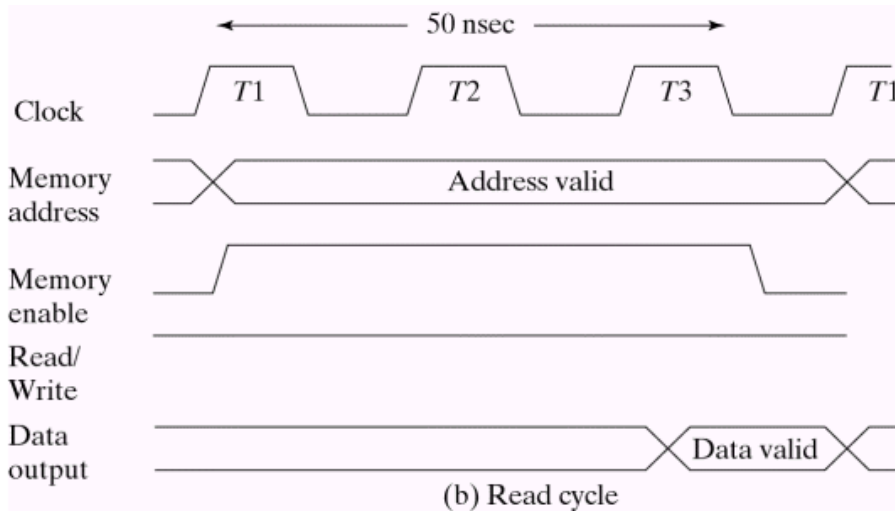
### EXAMPLE

- CPU at 50 MHz (20ns period)
- 50ns Cycle Time
- Need at least three periods (50ns)
- Periods called T1, T2, T3

## Memory Write Cycle



# Memory Read Cycle



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

## RAM Definitions

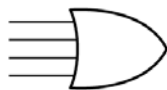
- RAM – Semiconductor Memory (in chips/not disks)
  - Two General Types:
    - SRAM – Static RAM – Requires 4 Transistors/bit
      - Volatile – no refresh
      - Used for Cache
    - DRAM – Dynamic RAM – Requires 1 Transistor and one capacitor
      - Requires Refresh
      - Volatile – needs refresh

## Verilog Model of a Memory Unit

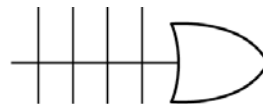
```
//HDL Example 7-1
//-----
//Read and write operations of memory.
//Memory size is 64 words of 4 bits each.
module memory (Enable,ReadWrite,Address,DataIn,DataOut);
    input  Enable,ReadWrite;
    input  [3:0] DataIn;
    input  [5:0] Address;
    output [3:0] DataOut;
    reg [3:0] DataOut;
    reg [3:0] Mem [0:63];          //64 x 4 memory
    always @ (Enable or ReadWrite)
        if (Enable)
            if (ReadWrite)
                DataOut = Mem[Address]; //Read
            else
                Mem[Address] = DataIn; //Write
        else DataOut = 4'bz;          //High impedance state
endmodule
```

## Multi-Input Gates and Fuses

- Many Logic Devices Have Multi-input Gates
- Fuses versus Anti-Fuses
- Field Programmable Gate Arrays (FPGAs)
- Programmable Logic Devices (PLDs)
  - Future ???



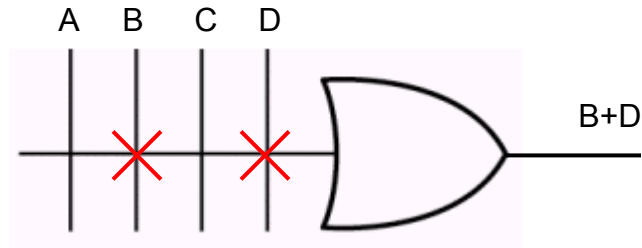
(a) Conventional symbol



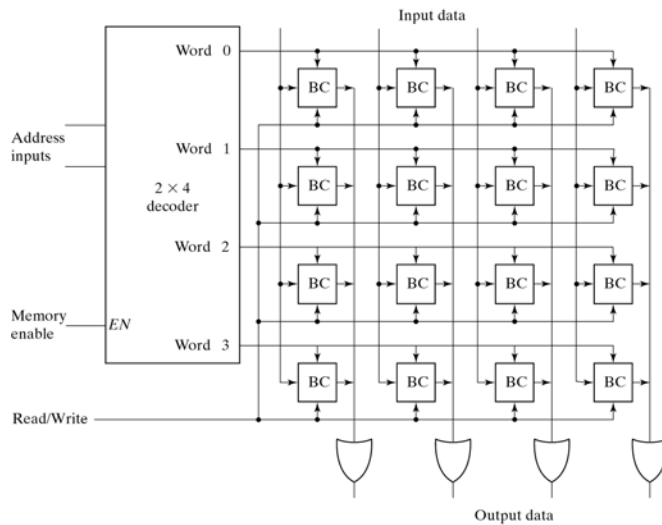
(b) Array logic symbol

Fig. 7-1 Conventional and Array Logic Diagrams for OR Gate

# Multi-Input Gate Notation



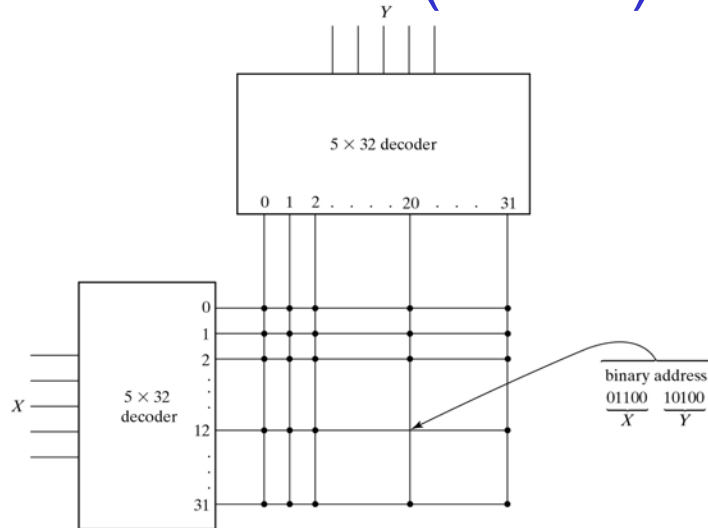
# 4x4 RAM Structure



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 7-6 Diagram of a 4 x 4 RAM

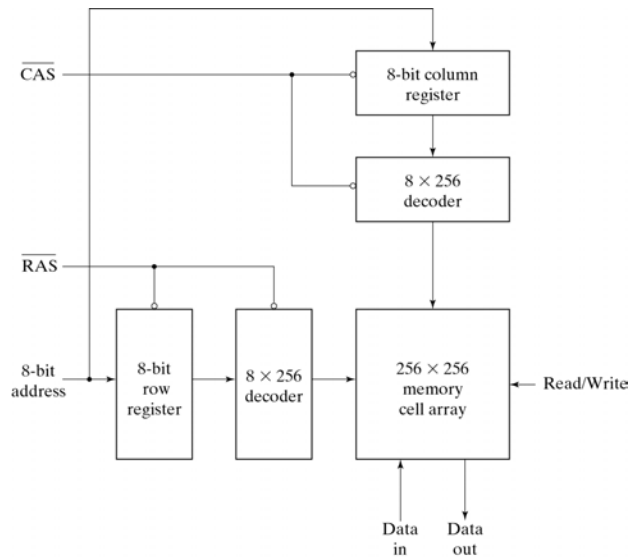
# 2D RAM Structure (DRAM)



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 7-7 Two-Dimensional Decoding Structure for a 1K-Word Memory

# DRAM Structure



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 7-8 Address Multiplexing for a 64K DRAM

# ROM Memory Diagram

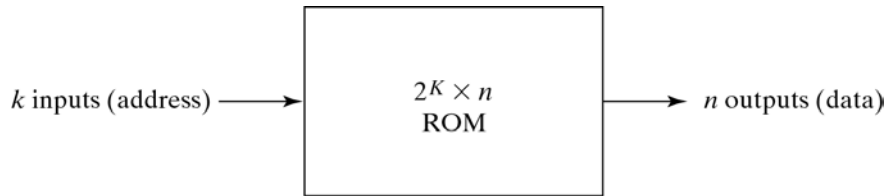


Fig. 7-9 ROM Block Diagram

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

# ROM Memory Model

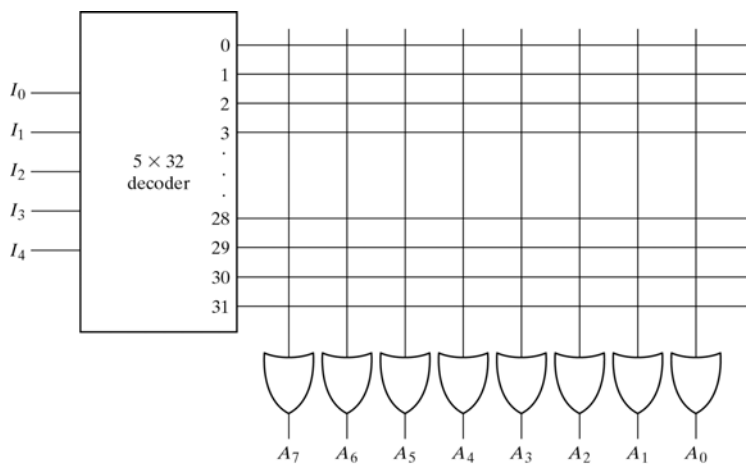


Fig. 7-10 Internal Logic of a  $32 \times 8$  ROM

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

# ROM Truth Table

Inputs					Outputs							
I4	I3	I2	I1	I0	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

# ROM Function Implementation

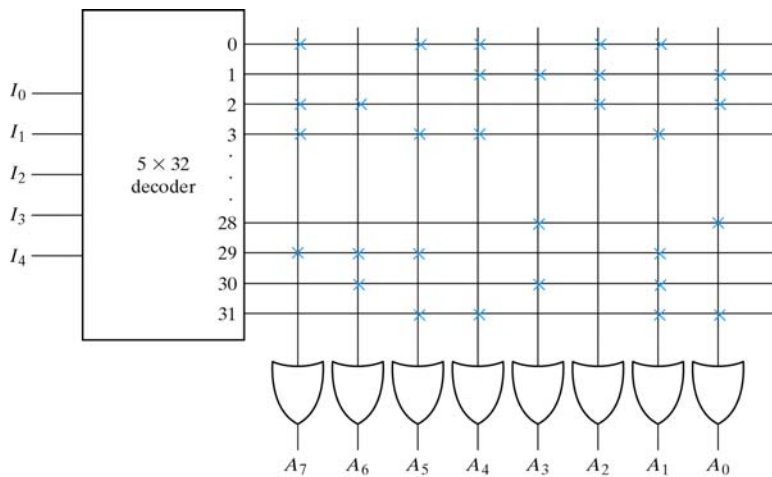


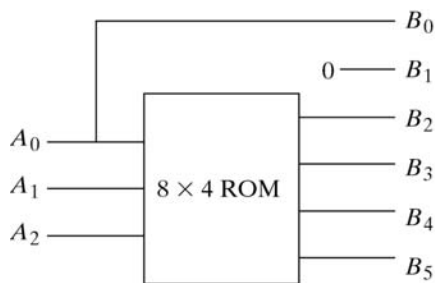
Fig. 7-11 Programming the ROM According to Table 7-3

# ROM Function Implementation

<u>Inputs</u>			<u>Outputs</u>						
A2	A1	A0	B5	B4	B3	B2	B1	B0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	40

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

# ROM Function Implementation



(a) Block diagram

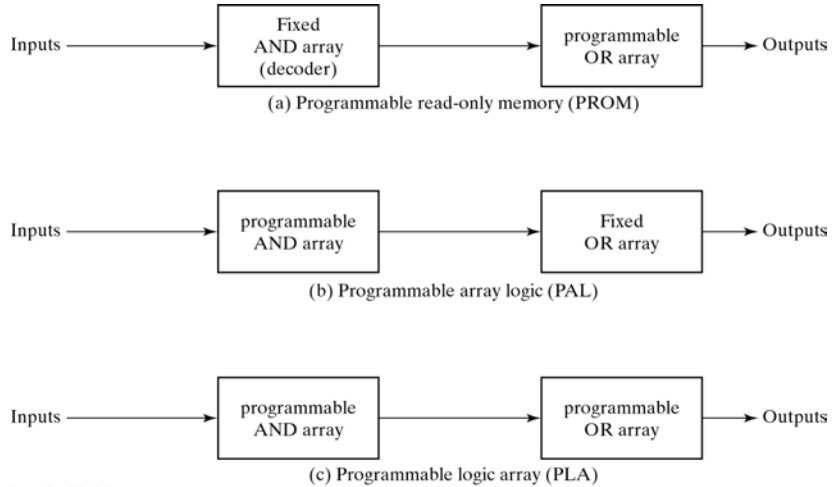
A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	B <sub>5</sub>	B <sub>4</sub>	B <sub>3</sub>	B <sub>2</sub>
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

(b) ROM truth table

Fig. 7-12 ROM Implementation of Example 7-1

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

# Programmable Logic Devices (PLDs)



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 7-13 Basic Configuration of Three PLDs

## PLA Implementation

$$F_1 = A\bar{B} + AC + \bar{A}B\bar{C}$$

$$F_2 = \overline{AC + BC}$$

	Product Term	Inputs			Outputs	
		A	B	C	(T) $F_1$	(C) $F_2$
$A\bar{B}$	1	1	0	-	1	-
$AC$	2	1	-	1	1	1
$BC$	3	-	1	1	-	1
$\bar{A}B\bar{C}$	4	0	1	0	1	-

# Design Data

		BC		B	
		00	01	11	10
A	0	1	1	0	1
	1	1	0	0	0
		C			

$$F_1 = A'B' + A'C' + B'C'$$

$$F_1 = (AB + AC + BC)'$$

		BC		B	
		00	01	11	10
A	0	1	0	0	0
	1	0	1	1	1
		C			

$$F_2 = AB + AC + A'B'C'$$

$$F_2 = (A'C + A'B + AB'C)'$$

PLA programming table

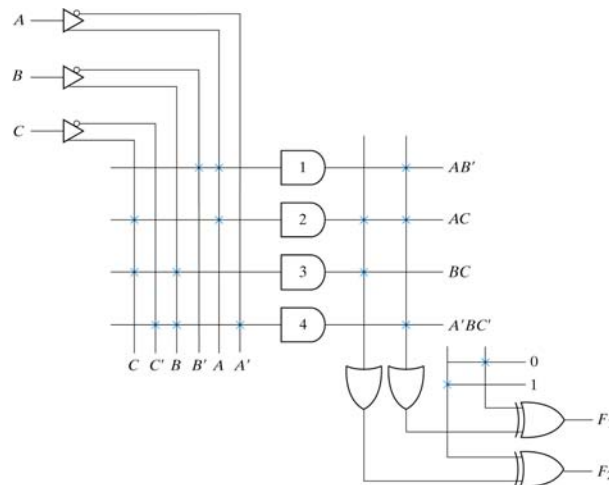
Product term	Inputs A B C	Outputs (C) (T)	
		F <sub>1</sub>	F <sub>2</sub>
AB	1 1 -	1	1
AC	1 - 1	1	1
BC	- 1 1	1	-
A'B'C'	0 0 0	-	1

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 7-15 Solution to Example 7-2

## PLA Structure with 'Programmable Inverters'

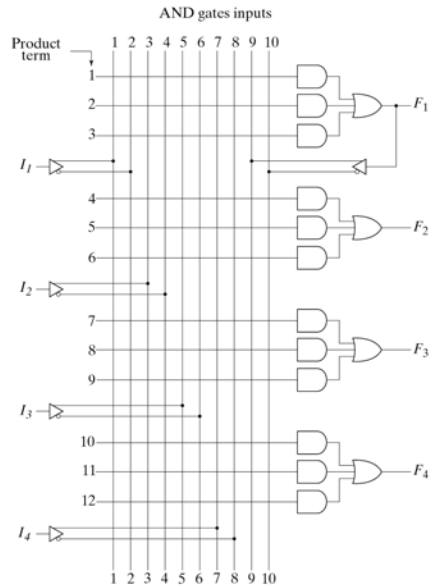
Give the Boolean expression for F<sub>1</sub>.



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs

## PAL Structure



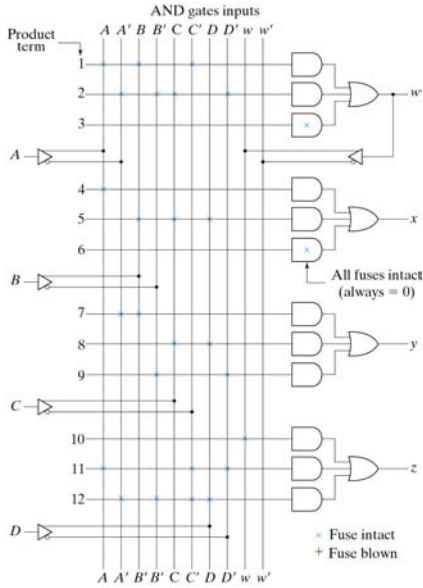
© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 7-16 PAL with Four Inputs, Four Outputs, and Three-Wide AND-OR Structure

## PAL Programming Table

Product Term	AND Inputs					Outputs
	A	B	C	D	W	
1	1	1	0	-	-	$w=ABC'$
2	0	0	1	0	-	$+A'B'CD'$
3	-	-	-	-	-	
4	1	-	-	-	-	$x=A$
5	-	1	1	1	-	$+BCD$
6	-	-	-	-	-	
7	0	1	-	-	-	$y=A'B$
8	-	-	1	1	-	$+CD$
9	-	0	-	0	-	$+B'D'$
10	-	-	-	-	1	$z=w$
11	1	-	0	0	-	$+AC'D'$
12	0	0	0	1	-	$+A'B'C'D$

## PAL Implementation



© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

Fig. 7-17 Fuse Map for PAL as Specified in Table 7-6

## Sequential Programmable Logic Device (SPLD)

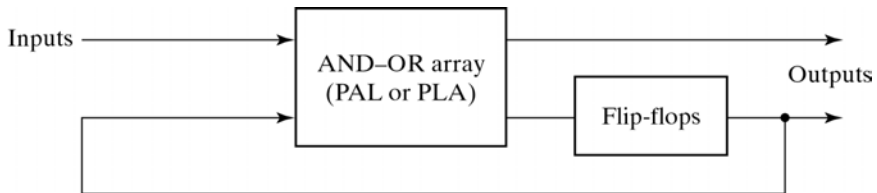


Fig. 7-18 Sequential Programmable Logic Device

© 2002 Prentice Hall, Inc.  
M. Morris Mano  
DIGITAL DESIGN, 3e.

# Typical SPLD Macrocell

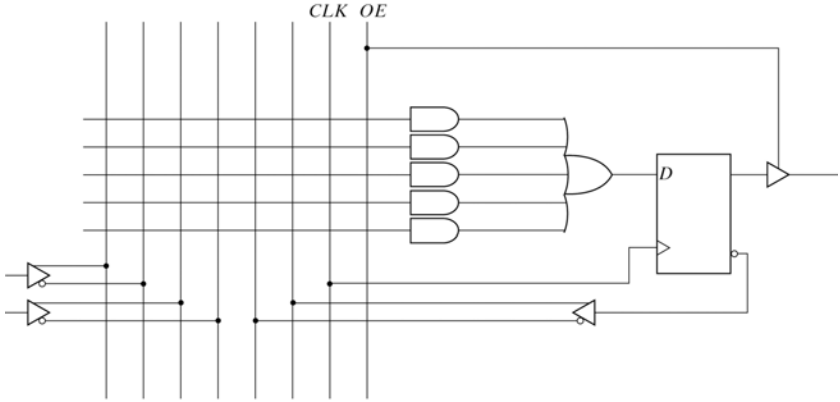


Fig. 7-19 Basic Macrocell Logic

© 2002 Prentice Hall, Inc.  
 M. Morris Mano  
**DIGITAL DESIGN, 3e.**

# Complex Programmable Logic Device (CPLD)

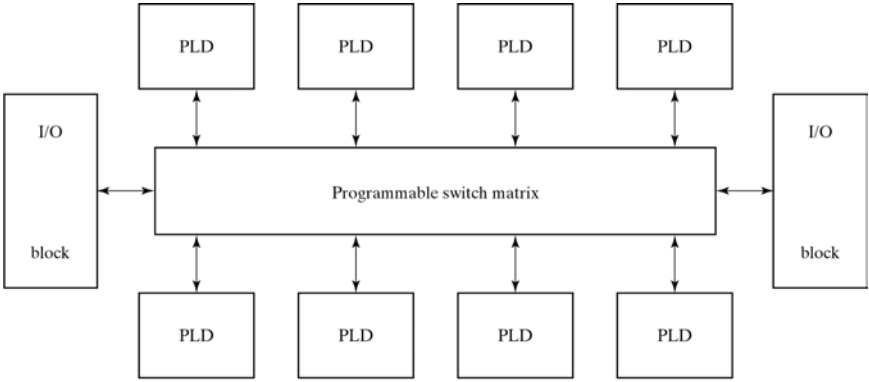


Fig. 7-20 General CPLD Configuration

© 2002 Prentice Hall, Inc.  
 M. Morris Mano  
**DIGITAL DESIGN, 3e.**