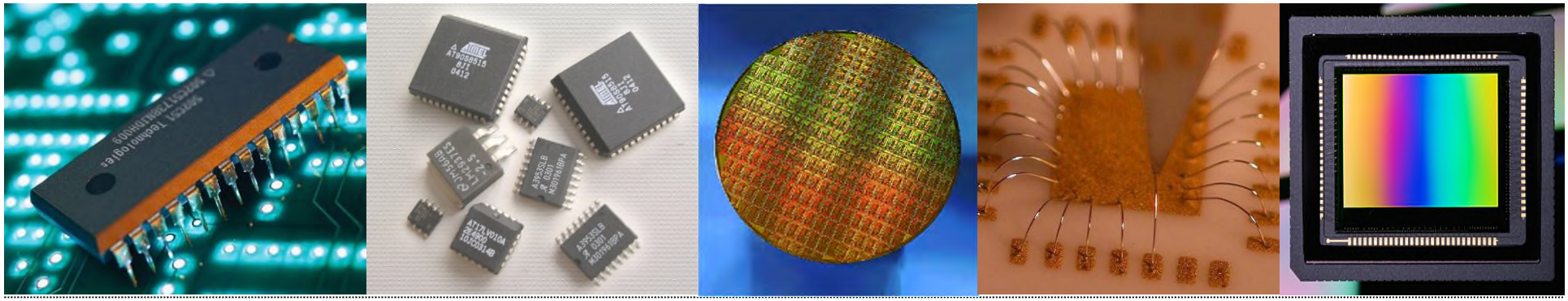


7. Memory and Programmable Logic

7.1 INTRODUCTION

- **Memory**

  RAM(random access memory): read and write operation

  ROM(read only memory) : only read operation

- **Programmable Logic Device**

  PLD(programmable logic device)

  PLA(programmable logic array)

  PAL(programmable array logic)

  FPGA(field programmable gate array)

(a) Conventional symbol (b) Array logic symbol

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

7.2 RANDOM-ACCESS MEMORY

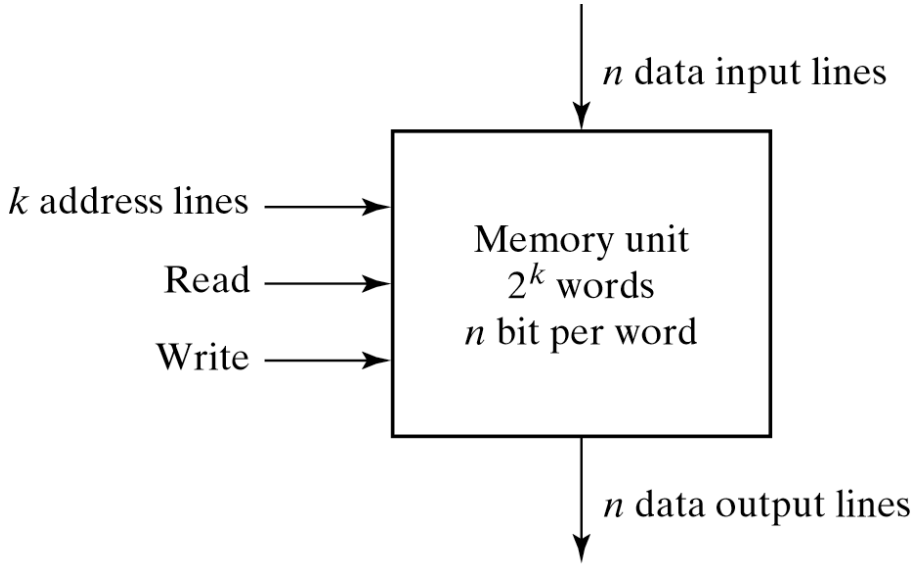


Fig. 7-2 Block Diagram of a Memory Unit

- The address lines select one particular word.
- A decoder inside the memory accepts this address and opens the paths needed to select the word specified.

7.2 RANDOM-ACCESS MEMORY

- 1024x16 Memory

Memory address		Memory content
Binary	Decimal	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
⋮	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

The 1K * 16 memory has 10 bits in the address and 16 bits in each word.

7.2 RANDOM-ACCESS MEMORY - Write and Read Operations

- Write operation

1. Transfer the binary address of the desired word to the address lines.
2. Transfer the data bits that must be stored in memory to the data input lines.
3. Activate the write input

- Read operation

1. Transfer the binary address of the desired word to the address lines.
2. Activate the read input.

Table 7.1
Control Inputs to Memory Chip

Memory Enable	Read/Write	Memory Operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

7.2 RANDOM-ACCESS MEMORY - Timing Waveforms

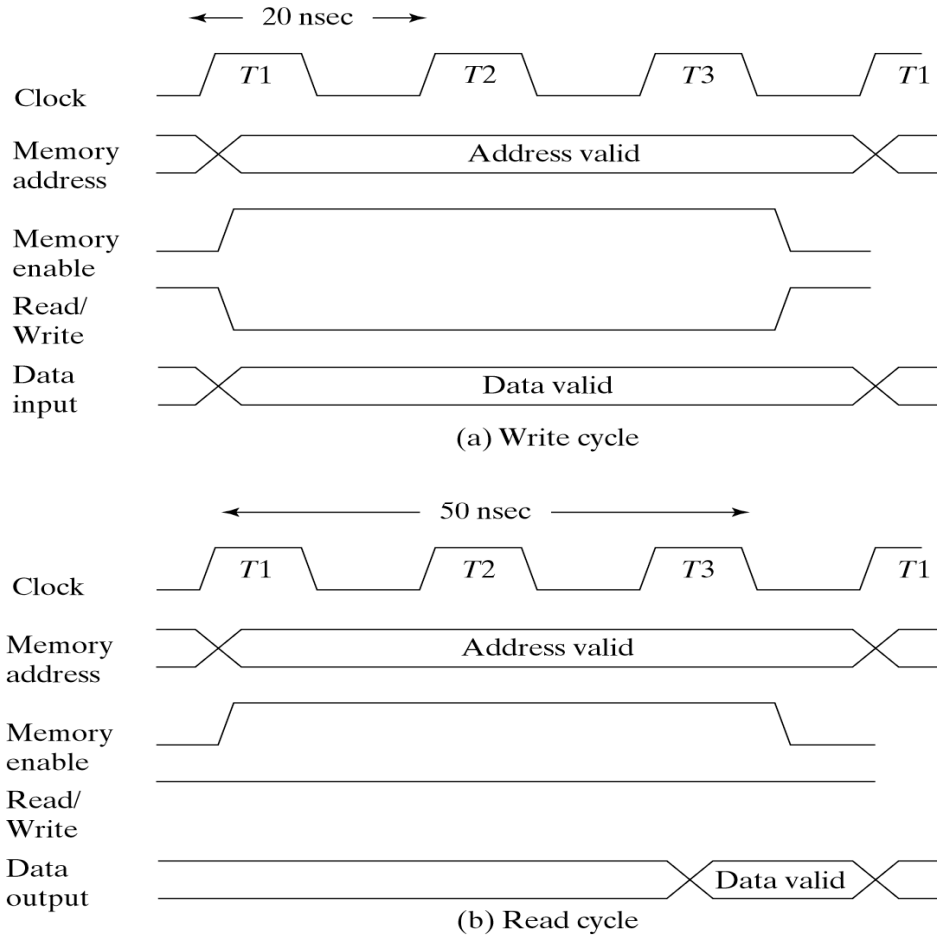
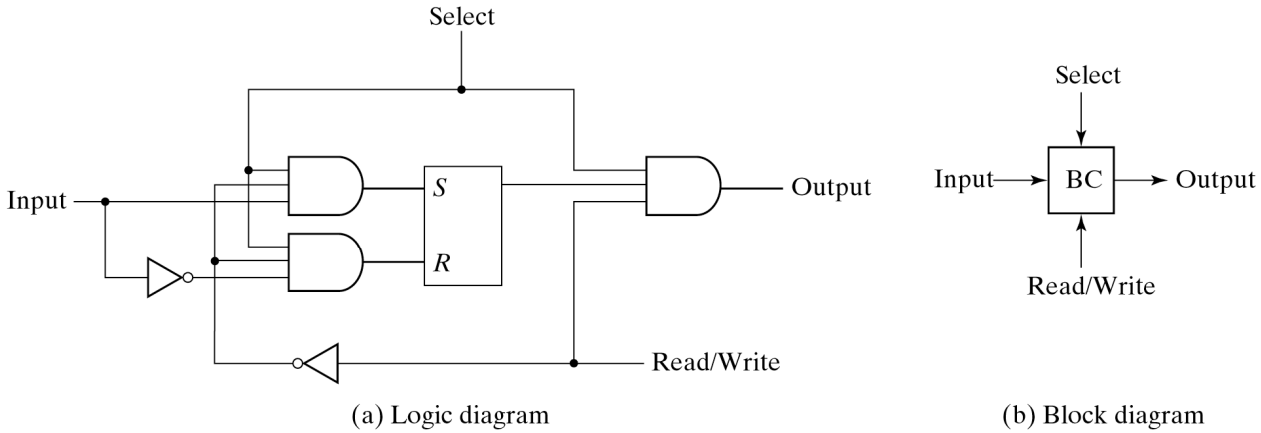


Fig. 7-4 Memory Cycle Timing Waveforms

7.2 RANDOM-ACCESS MEMORY - Types of Memories

- **Random access Memory** -each word occupy one particular location.the access time is always the same
- **Sequential access Memory**-information is read out only when the required word has been reached. the access time is variable.
- Volatile –Memory units lose the stored information when power is turned off.
- **Nonvolatile**-A nonvolatile memory retains its stored information after removal of power. ex) magnetic disk ,ROM(Read Only Memory)
- **Static RAM**-The stored information remains valid as long as power is applied to the unit. Static **RAM** is easier to use and has shorter read and write cycles.
- **Dynamic RAM**-The capacitors must be periodically recharged by refreshing the dynamic memory. Dynamic **RAM** offers reduced power consumption and larger storage capacity

7.3 MEMORY DECODING - Internal Construction



(a) Logic diagram (b) Block diagram

Fig. 7-5 Memory Cell

- The equivalent logic of a binary cell that stores one bit of information
- The binary cell stores one bit in its internal flip-flop
- It has three inputs and one output. The read/write input determines the cell operation when it is selected.

7.3 MEMORY DECODING - Internal Construction

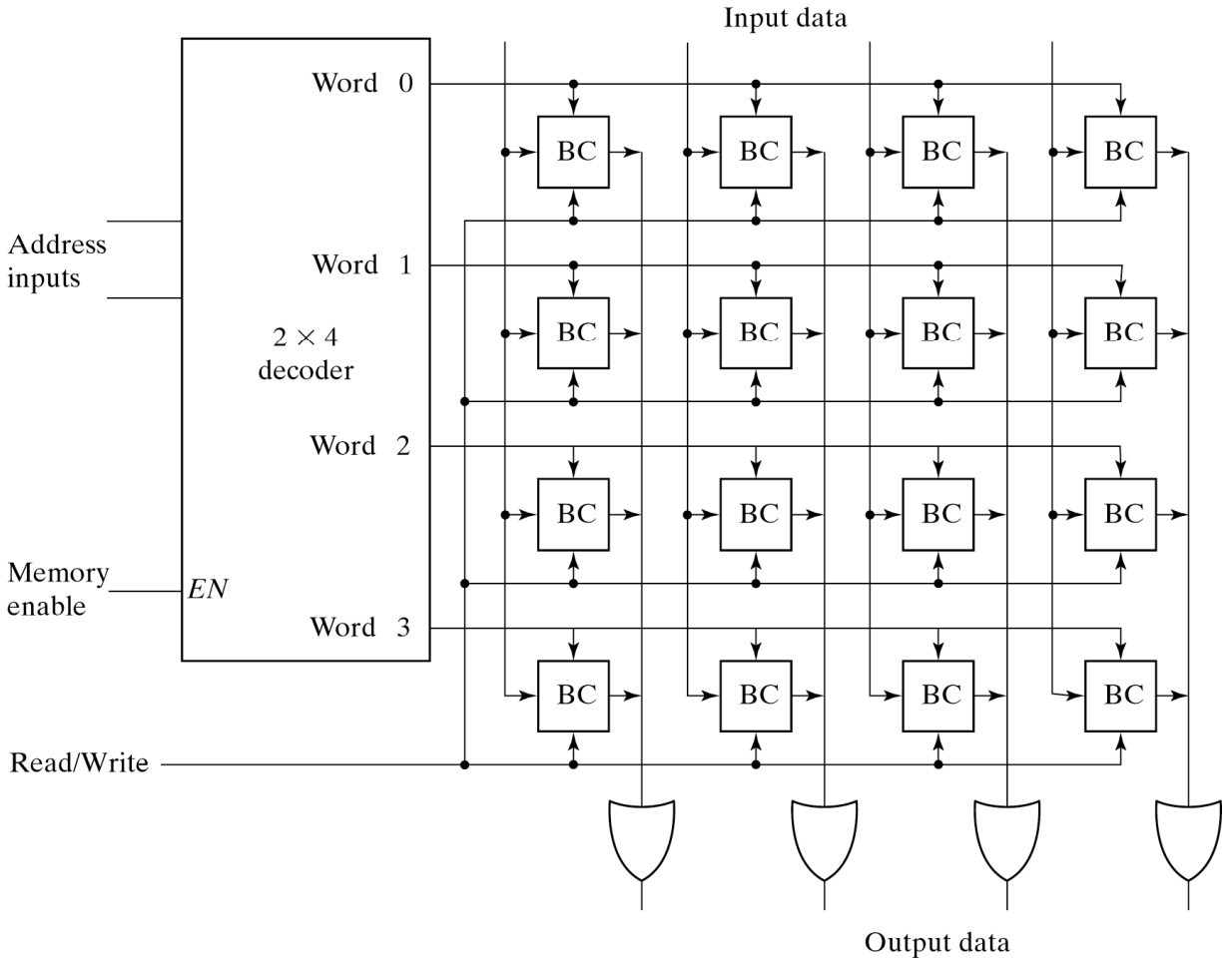


Fig. 7-6 Diagram of a 4 × 4 RAM

7.3 MEMORY DECODING - Internal Construction

- It has 16 binary cells
- Memory enable=0; all outputs of the decoder =0 , none of the memory words are selected.
- Memory enable=1; one of the four words is selected. The read/write input determines the operation.
- During the read operation, the four bits of the selected word go through **OR** gates to the output terminals.
- During the write operation, the data available in the input lines are transferred into the four binary cells of the selected word.

7.3 MEMORY DECODING - Coincident Decoding

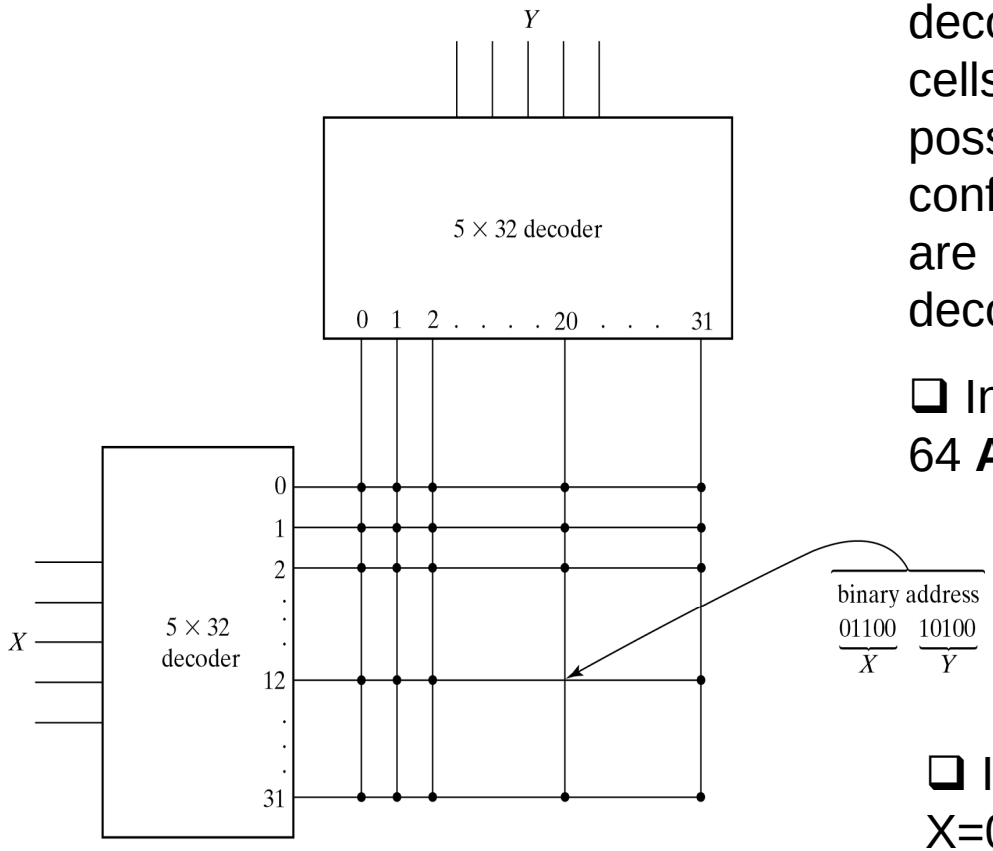


Fig. 7-7 Two-Dimensional Decoding Structure for a 1K-Word Memory

- The basic idea in two-dimensional decoding is to arrange the memory cells in an array that is close as possible as square. In this configuration, two k/2-input decoders are used instead of one k-input decoder.
- In the two-decoder case,we need 64 **AND** gates with five inputs in each.
- If the word address is 404, X=01100 (12) and Y=10100(20).

7.3 MEMORY DECODING - Address Multiplexing

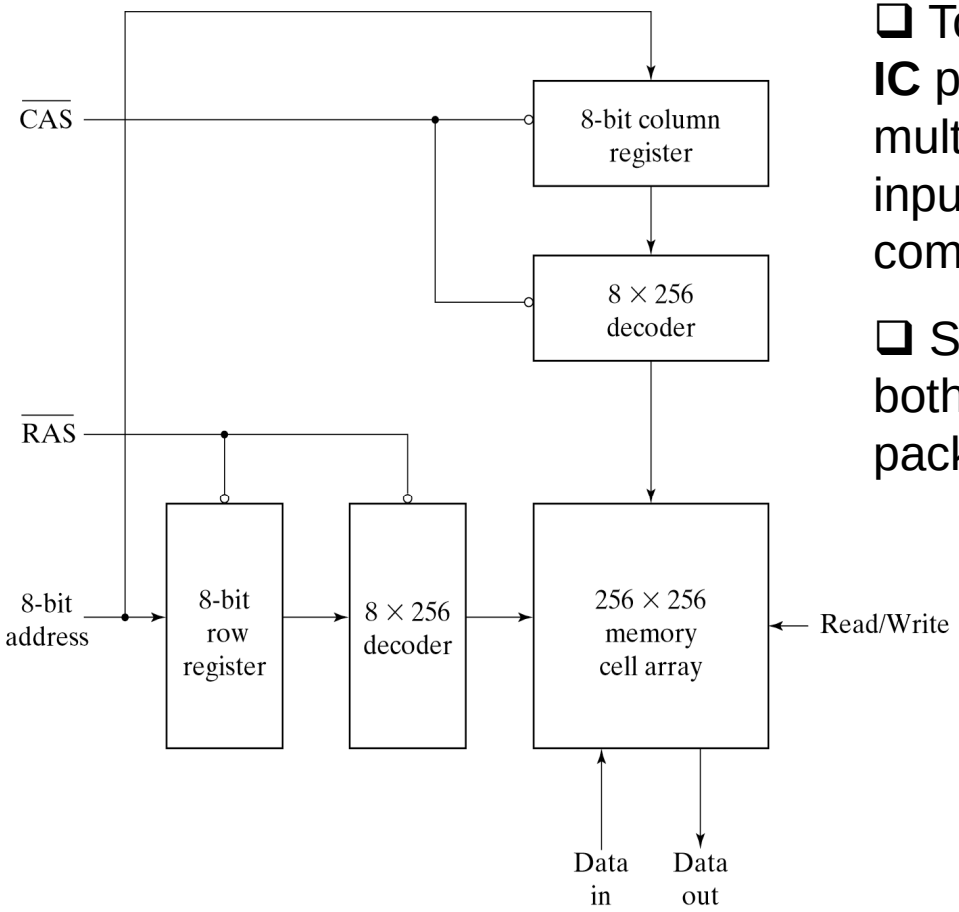


Fig. 7-8 Address Multiplexing for a 64K DRAM

- To reduce the number of pins in the **IC** package, designers utilize address multiplexing whereby one set of address input pins accommodates the address components.
- Since the same set of pins is used for both parts of the address, the size of the package is decreased significantly.

7.4 ERROR DETECTION AND CORRECTION

8-bit data word =>11000100

Bit position 1 2 3 4 5 6 7 8 9 10 11 12

P_1 P_2 1 P_4 1 0 0 P_8 0 1 0 0

P_1=XOR of bits(3,5,7,9,11)=0 (**XOR** performs the odd function.)

P_2=XOR of bits(3,6,7,10,11)=0 **7-4는 시험범위 제외**

P_4=XOR of bits(5,6,7,12)=1

P_8=XOR of bits(9,10,11,12)=1

In memory, **Bit position** 1 2 3 4 5 6 7 8 9 10 11 12

0 0 1 1 1 0 0 1 0 1 0 0

When the 12 bits are read from memory ,The four check bits

C_1=XOR of bits (1,3,5,7,9,11) **C_2=XOR** of bits (2,3,6,7,10,11)

C_3=XOR of bits (4,5,6,7,12) **C_4=XOR** of bits (8,9,10,11,12)

7.4 ERROR DETECTION AND CORRECTION - Hamming code

Since the bits were stored with even parity $C=C_8C_4C_2C_1=0000$ (No error)

Bit position	1	2	3	4	5	6	7	8	9	10	11	12	
	0	0	1	1	1	0	0	1	0	1	0	0	**No error**
	1	0	1	1	1	0	0	1	0	1	0	0	**Error in bit 1**
	0	0	1	1	0	0	0	1	0	1	0	0	**Error in bit 5**

Check bits	C_8	C_4	C_2	C_1
With error in bit 1 :	0	0	0	0
With error in bit 5:	0	1	0	1

The error can be corrected by complementing the corresponding bit.

The **Hamming code** can detect and correct only a single error. Multiple errors are not detected.

To correct a single error and detect double errors ,we **include the additional parity bit.**

If C=0 and P=0 No error occurred

If C=1 and p=1 A single error occurred , which can be corrected

If C=1 and P=0 A double error occurred, which is detected but cannot be corrected

If C=0 and P=1 An error occurred in the P_{13} bit

7.5 READ-ONLY MEMORY

ROM=Decoder + **OR** gates

-permanent binary information is stored.

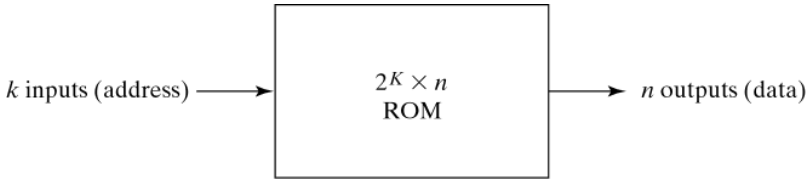


Fig. 7-9 ROM Block Diagram

k input lines and n output lines

The number of output lines ,n= the number of bits per word

7.5 READ-ONLY MEMORY

ROM = **AND** gates connected as a decoder + a number of **OR** gates

5 X 32 decoder has 32 **AND** gates and 5 inverters.

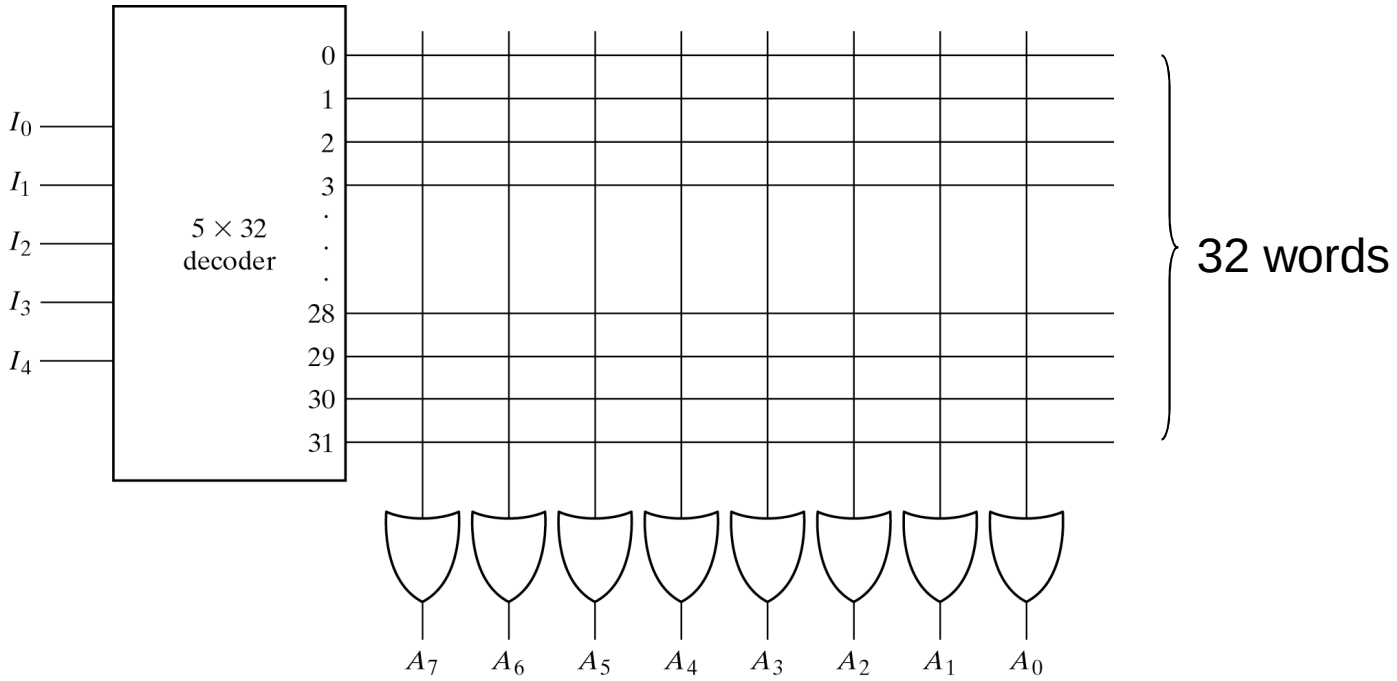


Fig. 7-10 Internal Logic of a 32 × 8 ROM

7.5 READ-ONLY MEMORY

- ROM Truth Table(Partial)

Table 7-3
ROM Truth Table (Partial)

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

7.5 READ-ONLY MEMORY - Combinational Circuit Implementation

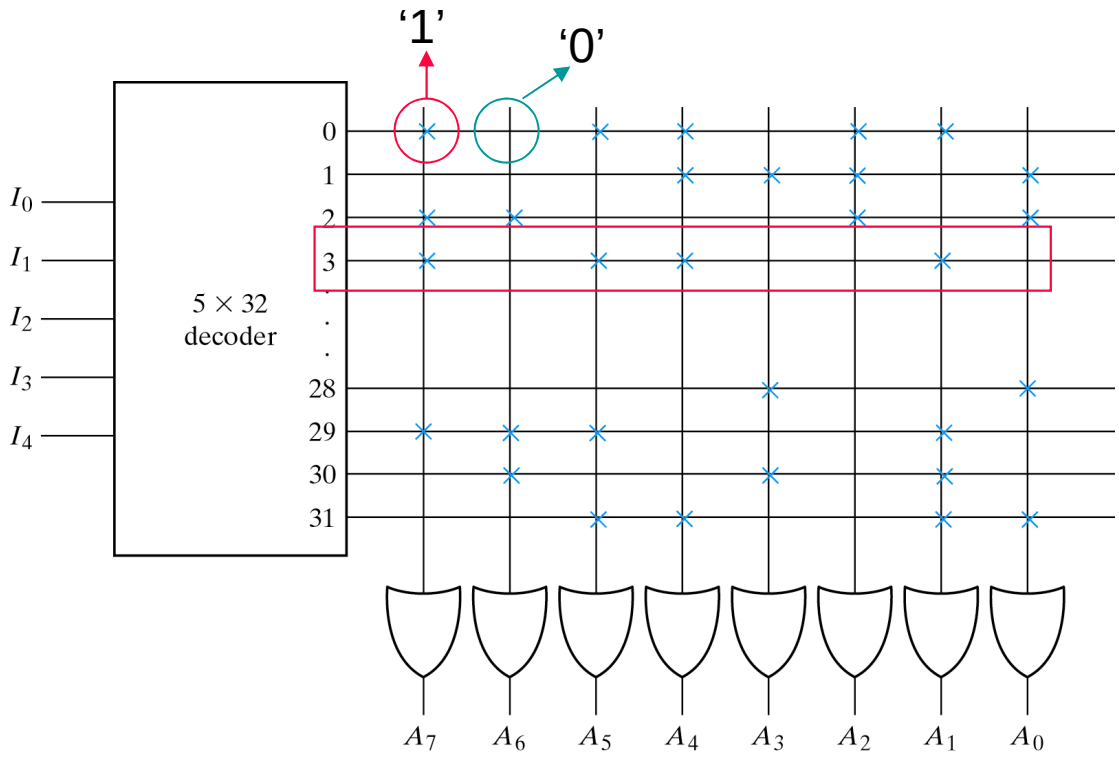


Fig. 7-11 Programming the ROM According to Table 7-3

$A_7(I_4,I_3,I_2,I_0)$=Sum of minterms(0,2,3,…,29)

Input->00011(3)

Others-> all '0'

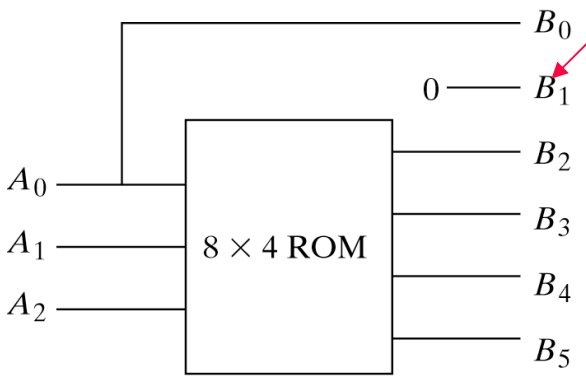
Output->10110010

7.5 READ-ONLY MEMORY - Combinational Circuit Implementation

- EXAMPLE 7-1

Table 7-4
Truth Table for Circuit of Example 7-1

Inputs			Outputs						Decimal
A_1	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	
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	49



(a) Block diagram

A_2	A_1	A_0	B_5	B_4	B_3	B_2
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

7.5 READ-ONLY MEMORY - Types of ROMs

For large quantities**, mask programming** is economical .

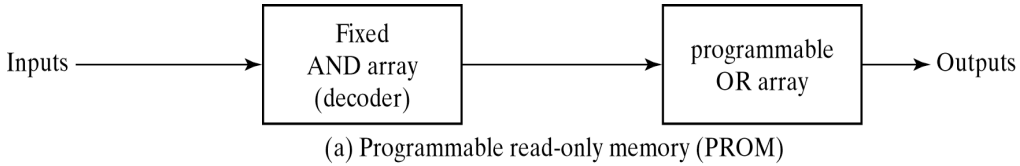
For small quantities , programmable read-only memory(**PROM**) is more economical.(all '**1**'s)

PROM- irreversible and permanent

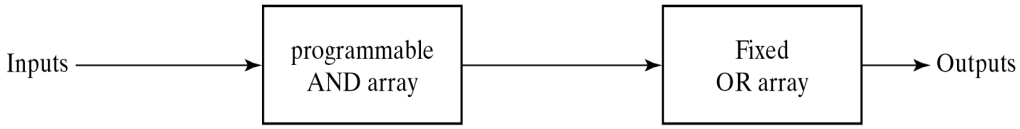
EPROM(Erasable PROM)- can be restructured to the initial value(**UV** light->discharge)

EEPROM(Electrically erasable PROM)- can be erased with electrical signals instead ultra violet light.

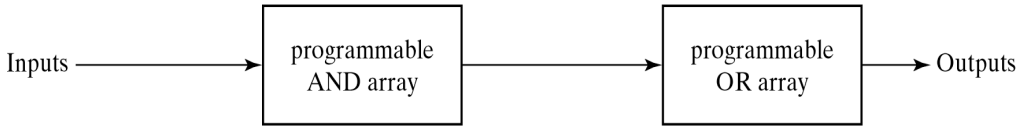
7.5 READ-ONLY MEMORY - Combinational PLDs



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL)



(c) Programmable logic array (PLA)

Fig. 7-13 Basic Configuration of Three PLDs

- A combinational **PLD** is an **IC** with programmable gates divided into an **AND** array and an **OR** array to provide an **AND-OR** sum of product implementation.
- Three major types **PLD**s

(a)fixed **AND** array +programmable **OR** array

(b)programmable **AND** array + fixed **OR** array

(c)programmable **AND** array + programmable **OR** array

7.6 PROGRAMMABLE LOGIC ARRAY

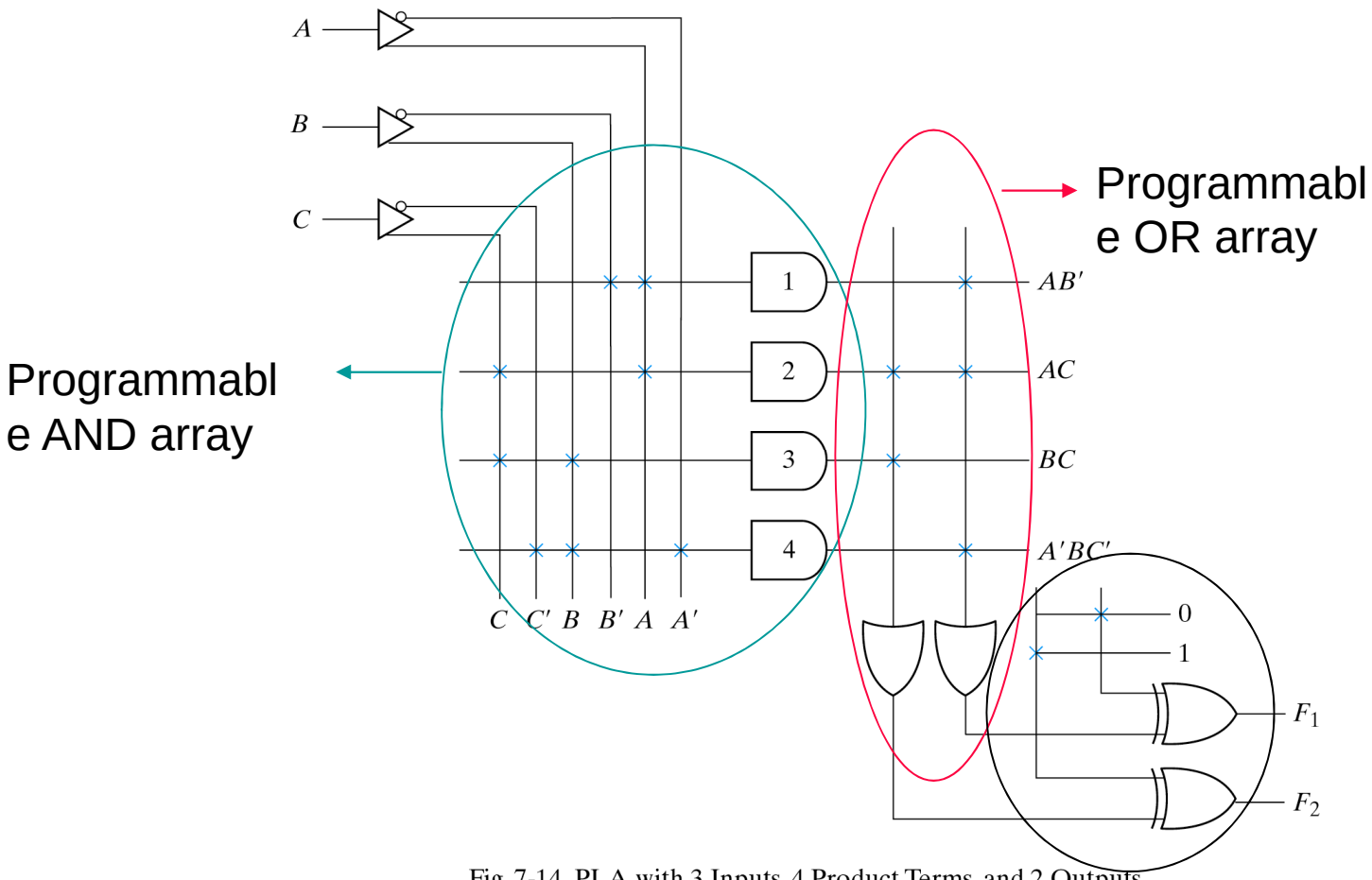


Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs

7.6 PROGRAMMABLE LOGIC ARRAY

PLA Programming Table

- The **PLA** does not provide full decoding of the variables.(decoder <->**AND** array)
- The **PLA** consists of **n** inputs ,**m** outputs , **k** product terms(**AND** gate) ,and **m** sum terms(**OR** gate)
- The number of programmed fuses is $2n*k+k*m+m$

  , whereas that of a **ROM** is $2^n * m$
- **F_1=AB' + AC + A'BC'**

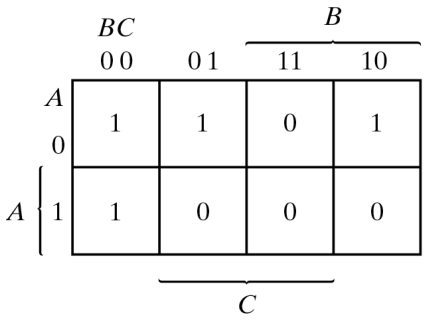
  F_2= (AC + BC)'

Table 7-5
PLA Programming Table

		Inputs			Outputs (T)	(C)
	Product Term	**A**	**B**	**C**	F_1	F_2
AB′	1	1	0	–	1	–
AC	2	1	–	1	1	1
BC	3	–	1	1	–	1
A′BC′	4	0	1	0	1	–

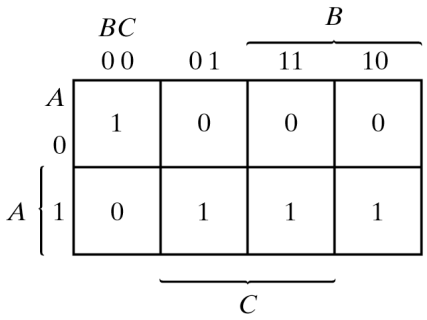
7.6 PROGRAMMABLE LOGIC ARRAY

EXAMPLE 7-2



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

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



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

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

$$F_1(A,B,C) = \sum(0,1,2,4)$$

$$F_2(A,B,C) = \sum(0,5,6,7)$$

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

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

PLA programming table

	Product term	Inputs A	B	C	Outputs (C) F_1	(T) F_2
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	–
$A'B'C'$	4	0	0	0	–	1

Fig. 7-15 Solution to Example 7-2

7.7 PROGRAMMABLE ARRAY LOGIC

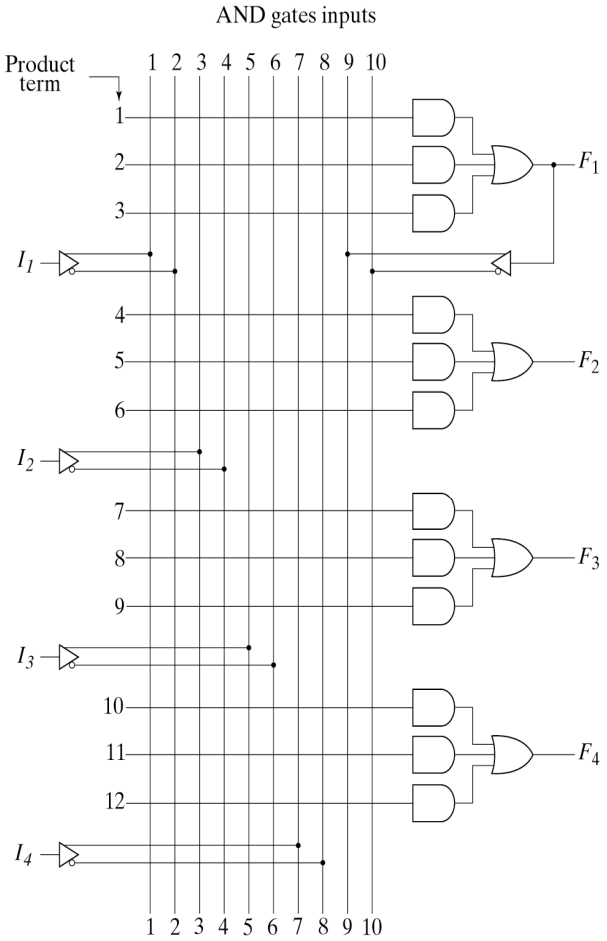


Fig. 7-16 PAL with Four Inputs, Four Outputs, and Three-Wide AND-OR Structure

PAL-With a fixed **OR** array and a programmable **AND** array.

-Not as flexible as the **PLA**(Only the **AND** gate are programmable.)

7.7 PROGRAMMABLE ARRAY LOGIC

- Example(PAL)

$$w(A,B,C,D) = \sum(2,12,13)$$
$$x(A,B,C,D) = \sum(7,8,9,10,11,12,12,14,15)$$
$$y(A,B,C,D) = \sum(0,2,3,4,5,6,7,8,10,11,15)$$
$$z(A,B,C,D) = \sum(1,2,8,12,13,)$$

$$w = ABC' + A'B'CD'$$
$$x = A + BCD$$
$$y = A'B + CD + B'D'$$
$$z = ABC' + A'B'CD' + AC'D' + A'B'C'D$$
$$= w + AC'D' + A'B'C'D$$

Table 7-6
PAL Programming Table

Product Term	AND Inputs A	B	C	D	W	Outputs
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	–	$+ A'B'C'D$

7.7 PROGRAMMABLE ARRAY LOGIC

- Fuse Map for PAL

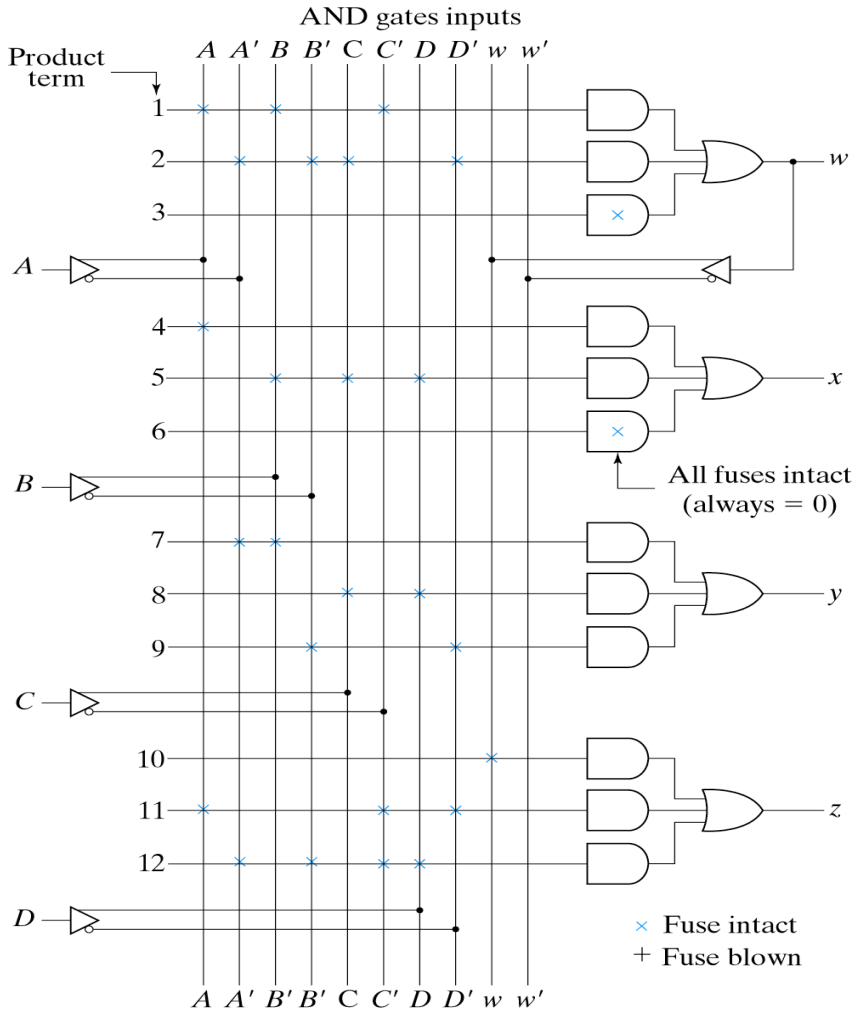


Fig. 7-17 Fuse Map for PAL as Specified in Table 7-6

7.8 SEQUENTIAL PROGRAMMABLE DEVICES

1. Sequential(or simple) Programmable Logic Device (**SPLD**)
2. Complex Programmable Logic Device(**CPLD**)
3. Field Programmable Gate Array(**FPGA**)

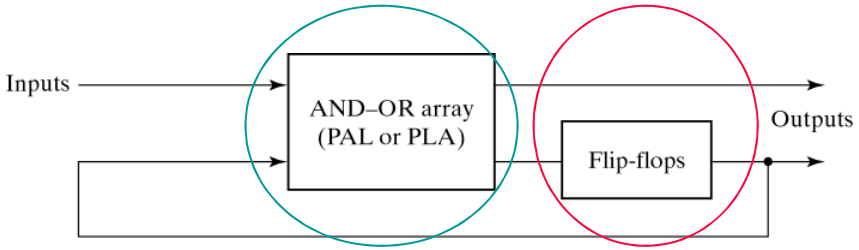


Fig. 7-18 Sequential Programmable Logic Device

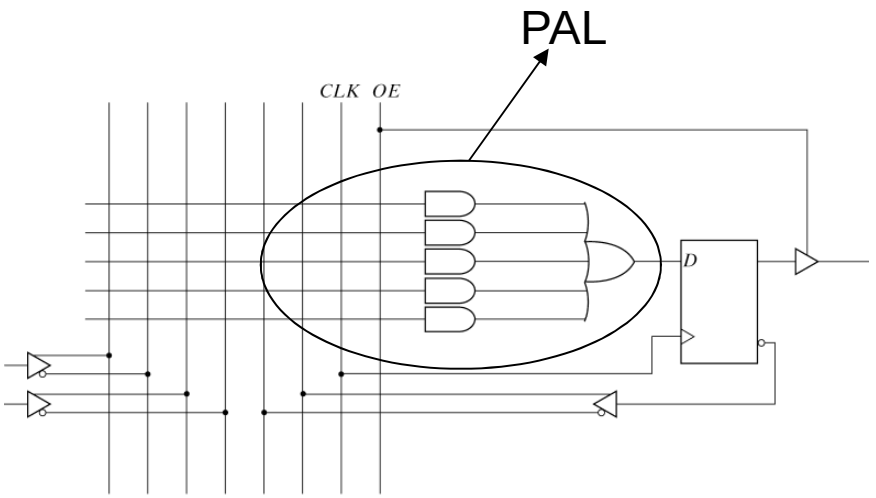


Fig. 7-19 Basic Macrocell Logic

7.8 SEQUENTIAL PROGRAMMABLE DEVICES

CPLD &FPGA

CPLD-a collection of individual PLDs

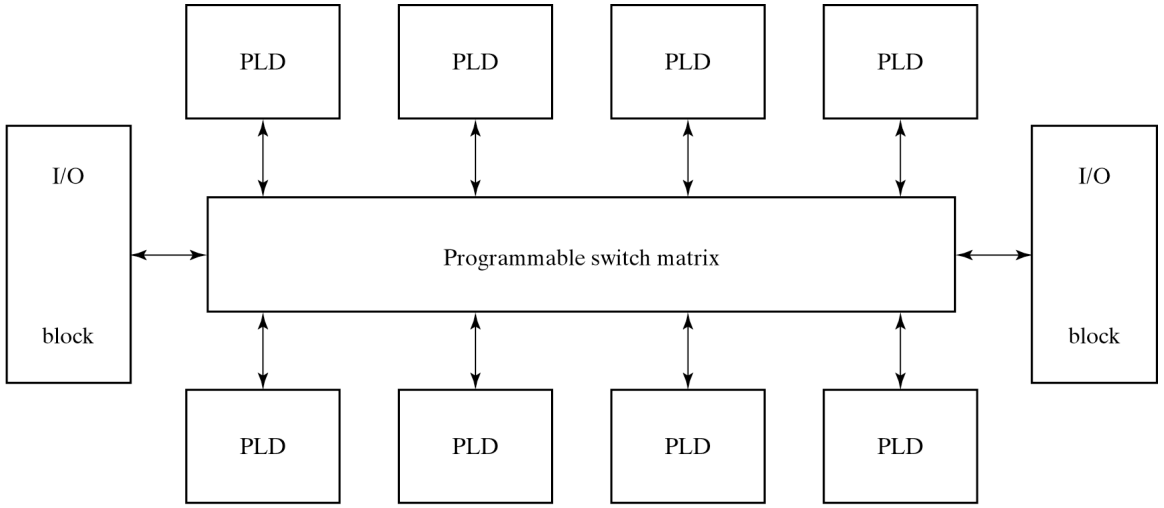


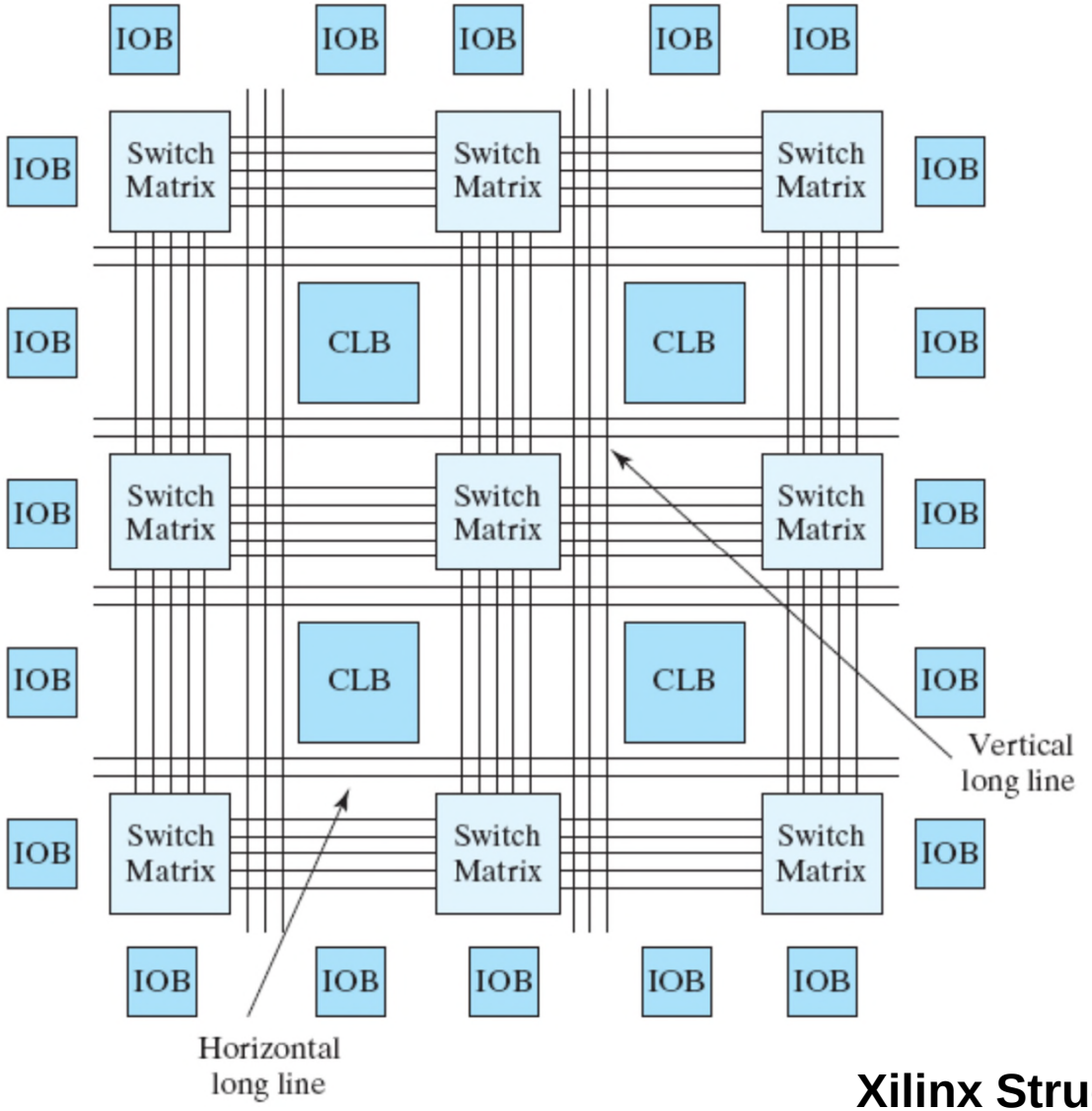
Fig. 7-20 General CPLD Configuration

FPGA(Field Programmable Gate Array)

-a **VLSI** circuit that can be programmed in the user's location.

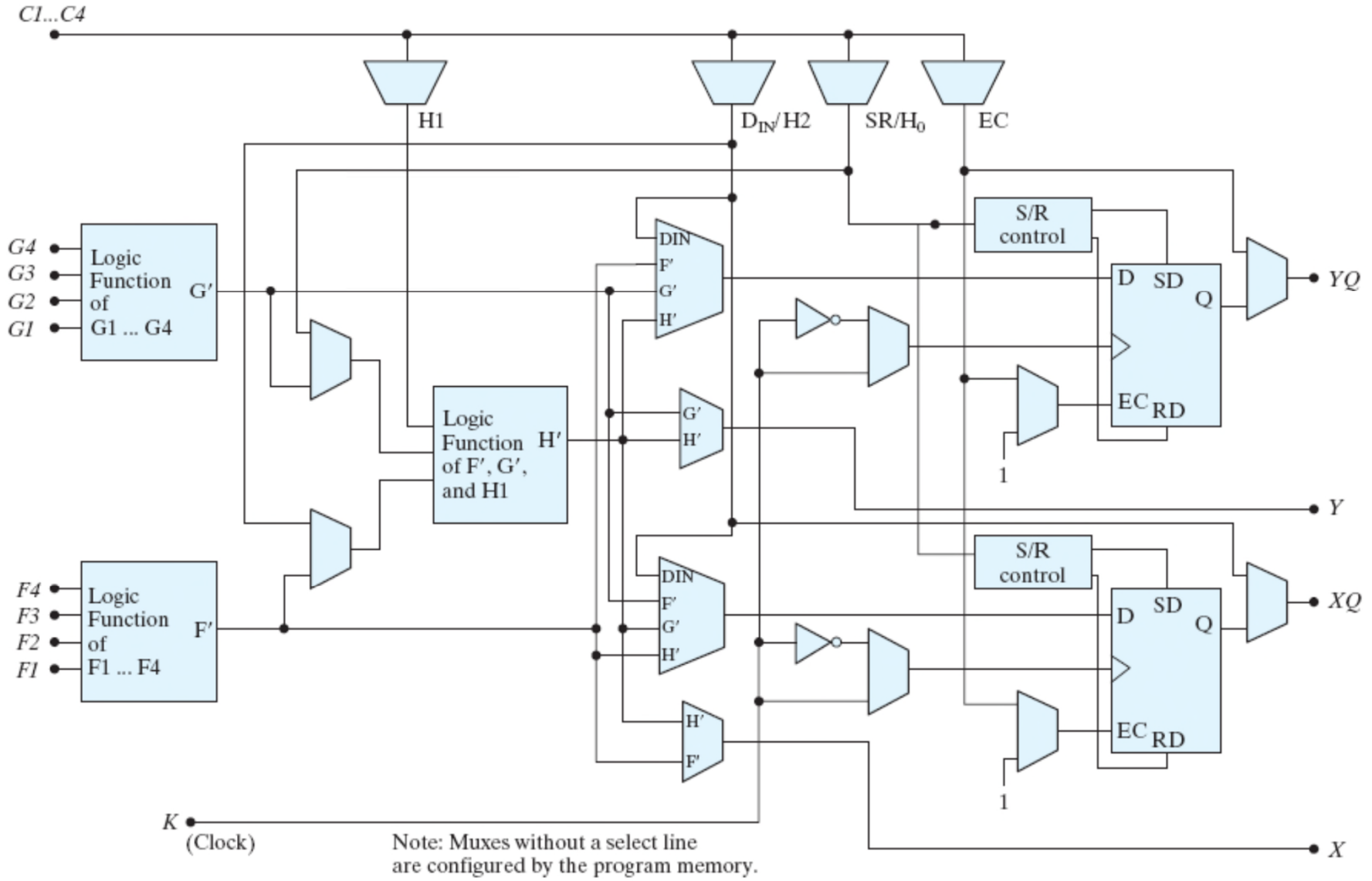
-look-up tables,multiplexes,gates, and flip-flops

7.8 SEQUENTIAL PROGRAMMABLE DEVICES

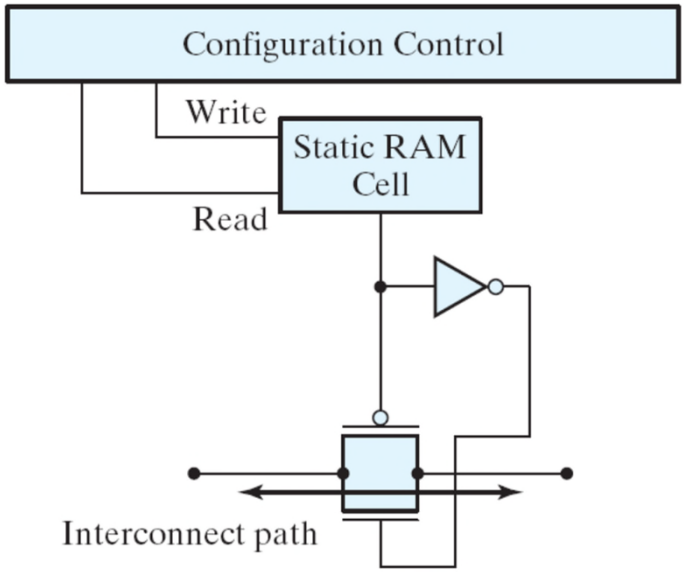


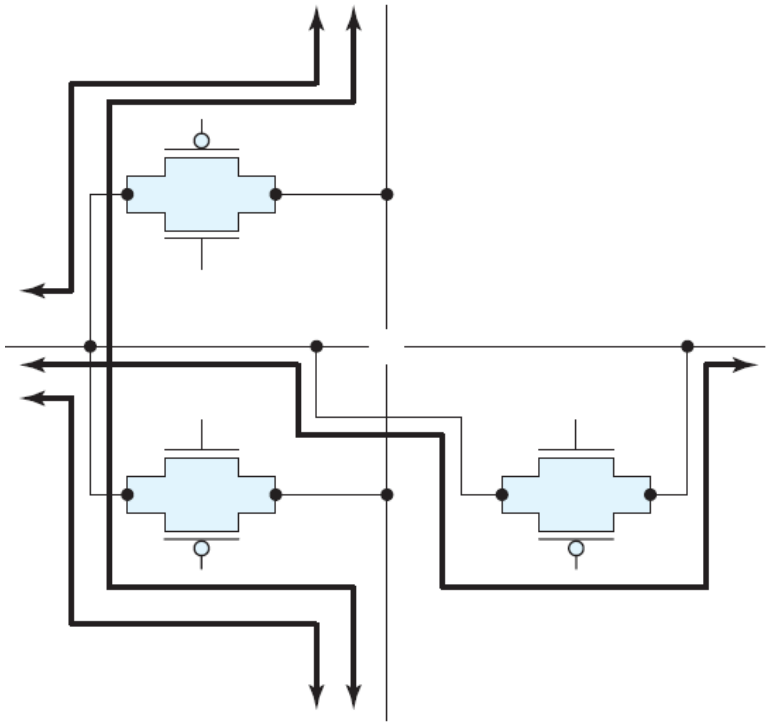
Xilinx Structure

7.8 SEQUENTIAL PROGRAMMABLE DEVICES

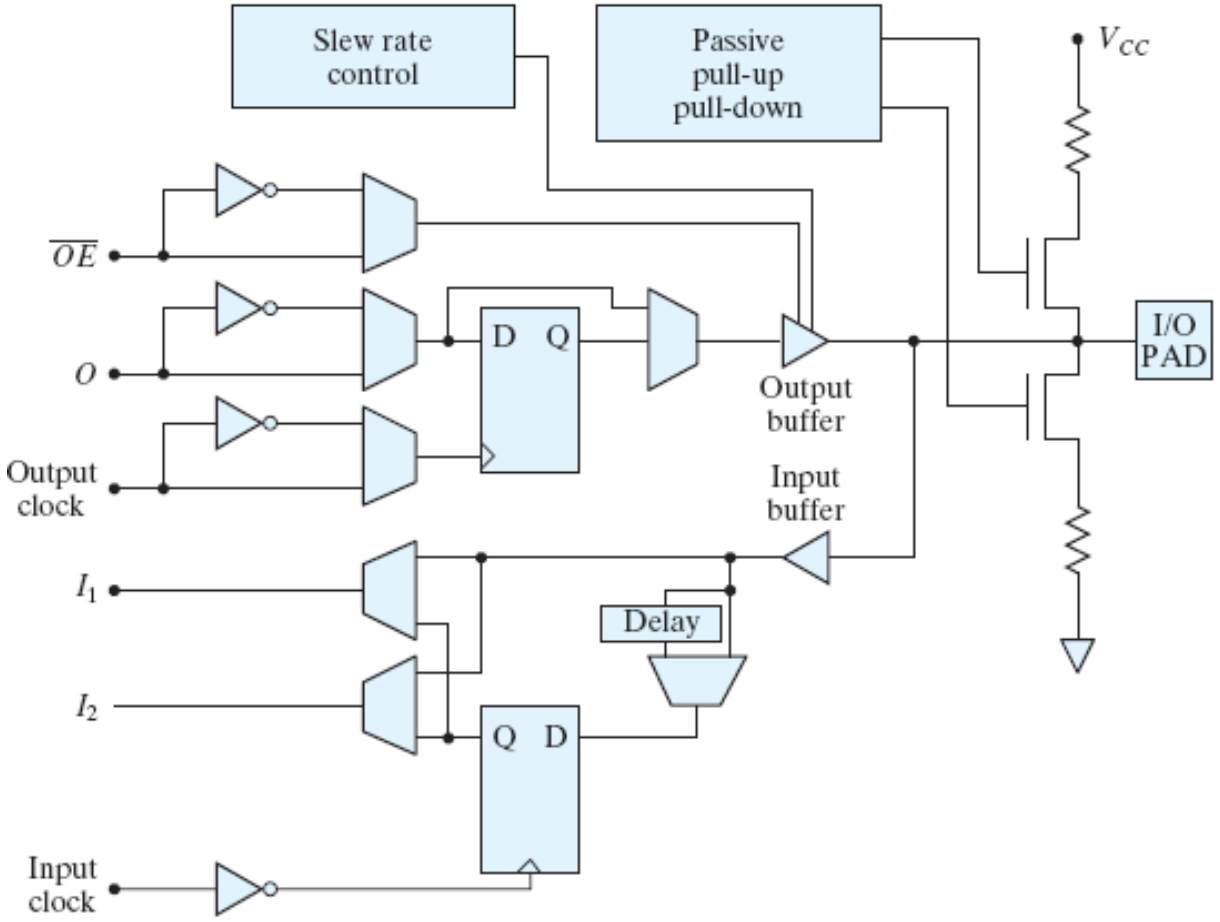


7.8 SEQUENTIAL PROGRAMMABLE DEVICES

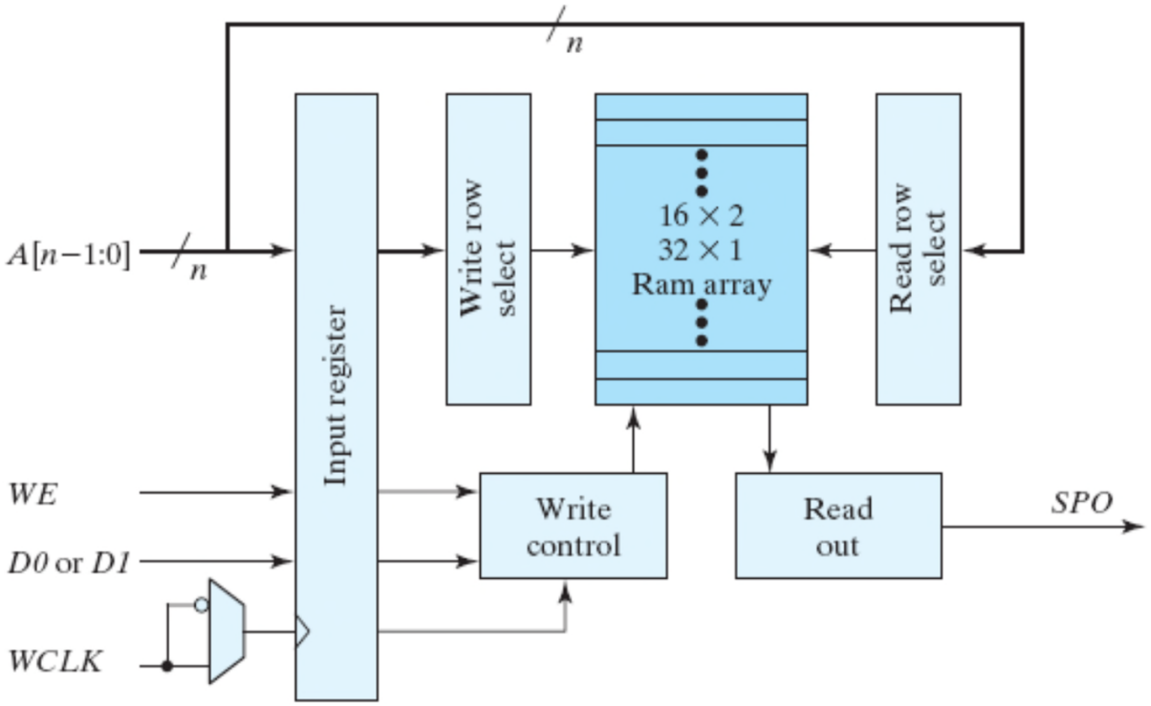




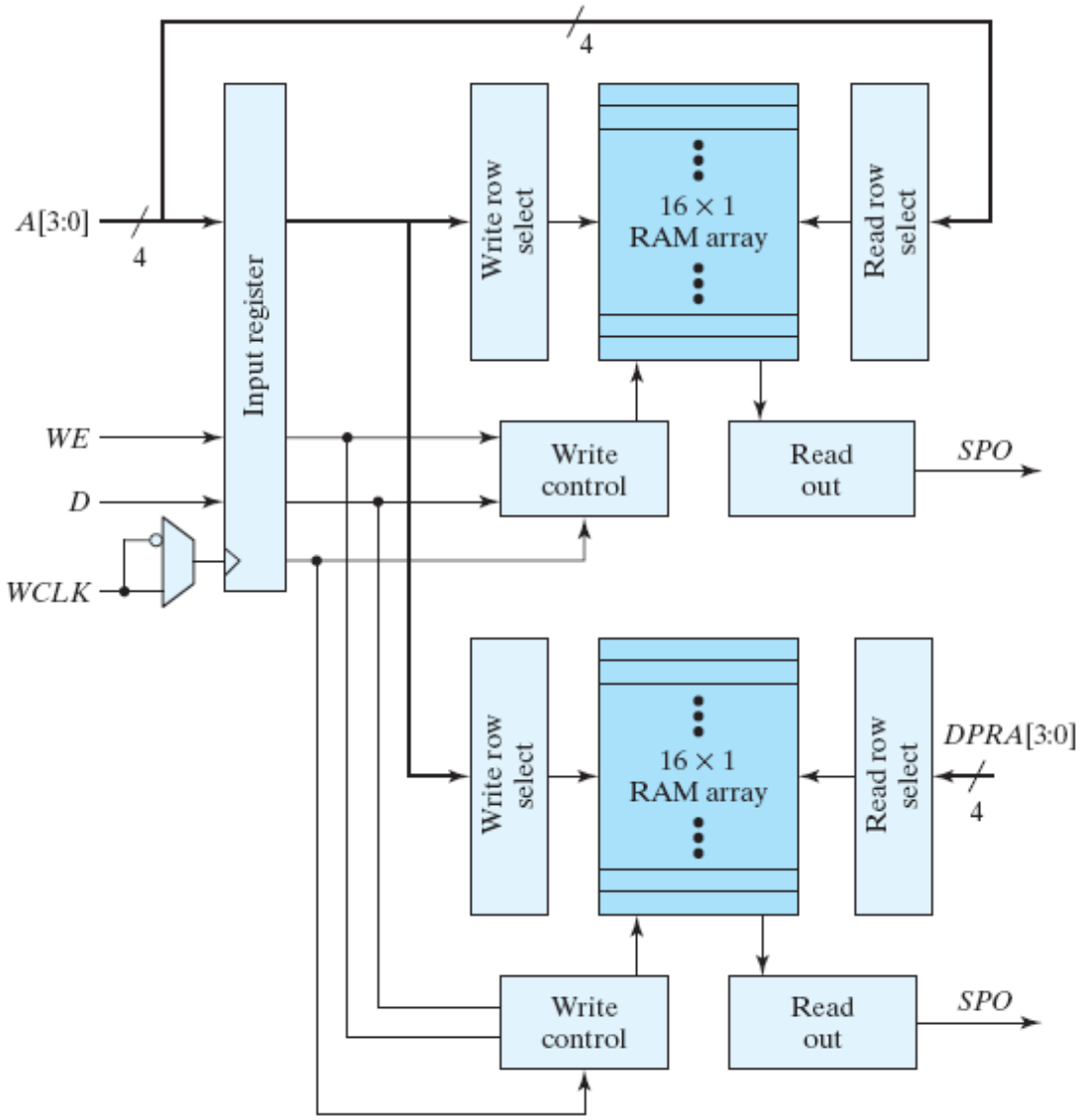
7.8 SEQUENTIAL PROGRAMMABLE DEVICES



7.8 SEQUENTIAL PROGRAMMABLE DEVICES



7.8 SEQUENTIAL PROGRAMMABLE DEVICES



7.8 SEQUENTIAL PROGRAMMABLE DEVICES

Table 7.7
Attributes of the Xilinx Spartan XL Device Family

Spartan XL	XCS05/XL	XCS10/XL	XCS20/XL	XCS30/XL	XCS40/XL
System Gates[1]	2K–5K	3K–10K	7K–20K	10K–30K	13K–40K
Logic Cells[2]	238	466	950	1,368	1,862
Max Logic Gates	3,000	5,000	10,000	13,000	20,000
Flip-Flops	360	616	1,120	1,536	2,016
Max RAM Bits	3,200	6,272	12,800	18,432	25,088
Max Avail I/O	77	112	160	192	224

[1] 20–30% of CLBs as RAM.
[2] 1 Logic cell = four-input lookup table + flip-flop.

Table 7.8
Spartan II Device Attributes

Spartan II FPGAs	XC2S15	XC2S30	XC2S50	XC2S100	XC2S150	XC2S200
System Gates[1]	6K–15K	13K–30K	23K–50K	37K–100K	52K–150K	71K–200K
Logic Cells[2]	432	972	1,728	2,700	3,888	5,292
Block RAM Bits	16,384	24,576	32,768	40,960	49,152	57,344
Max Avail I/O	86	132	176	196	260	284

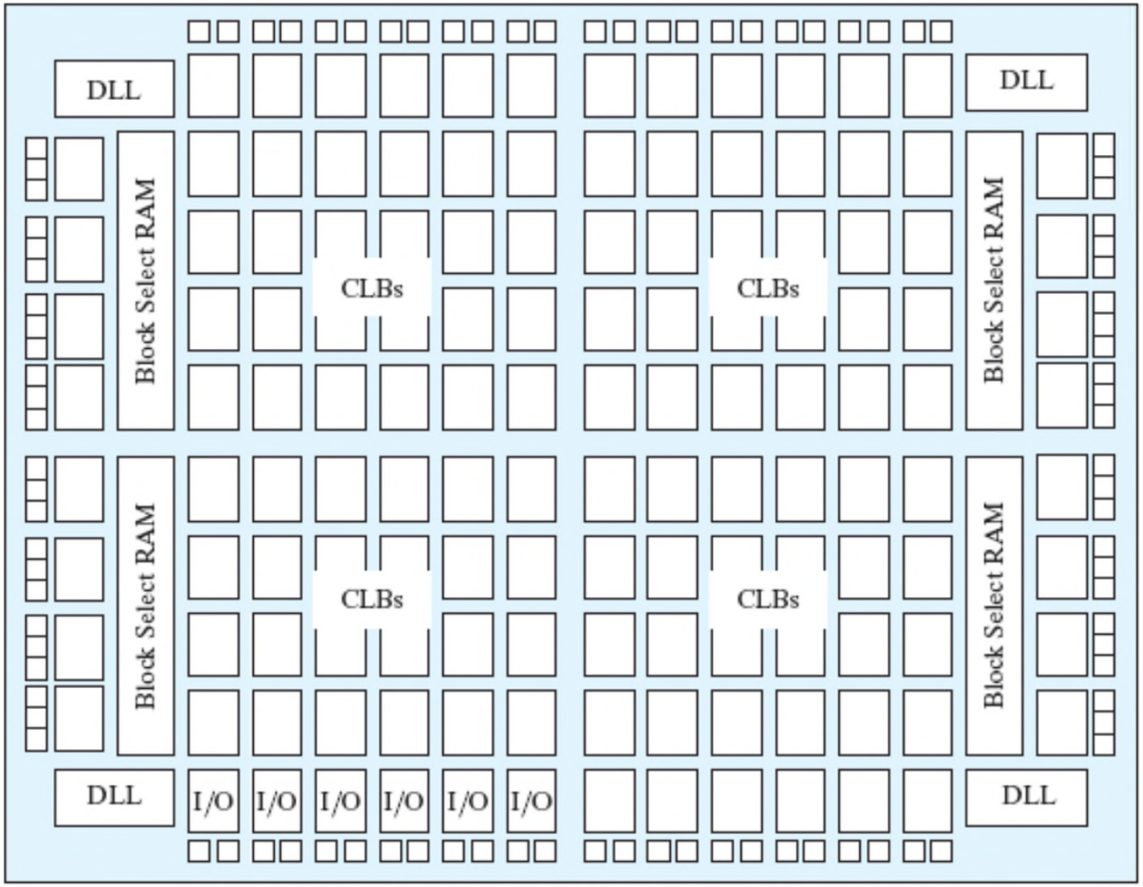
[1] 20–30% of CLBs as RAM.
[2] 1 Logic cell = four-input lookup table + flip-flop.

7.8 SEQUENTIAL PROGRAMMABLE DEVICES

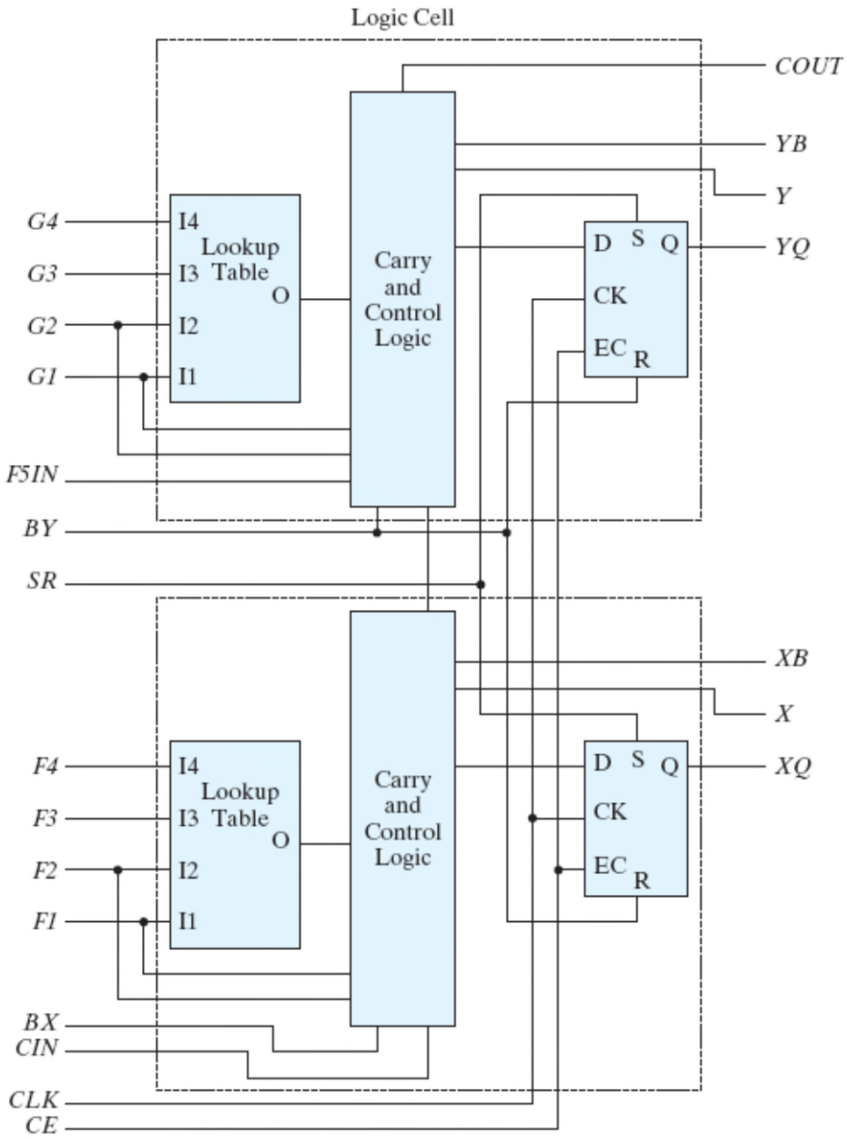
Table 7.9
Comparison of the Spartan Device Families

Part	Spartan	Spartan XL	Spartan II
Architecture	XC4000 Based	XC4000 Based	Virtex Based
Max # System Gates	5K–40K	5K–40K	15K–200K
Memory	Distributed RAM	Distributed RAM	Block + Distributed
I/O Performance	80 MHz	100 MHz	200 MHz
I/O Standards	4	4	16
Core Voltage	5 V	3.3 V	2.5 V
DLLs	No	No	Yes

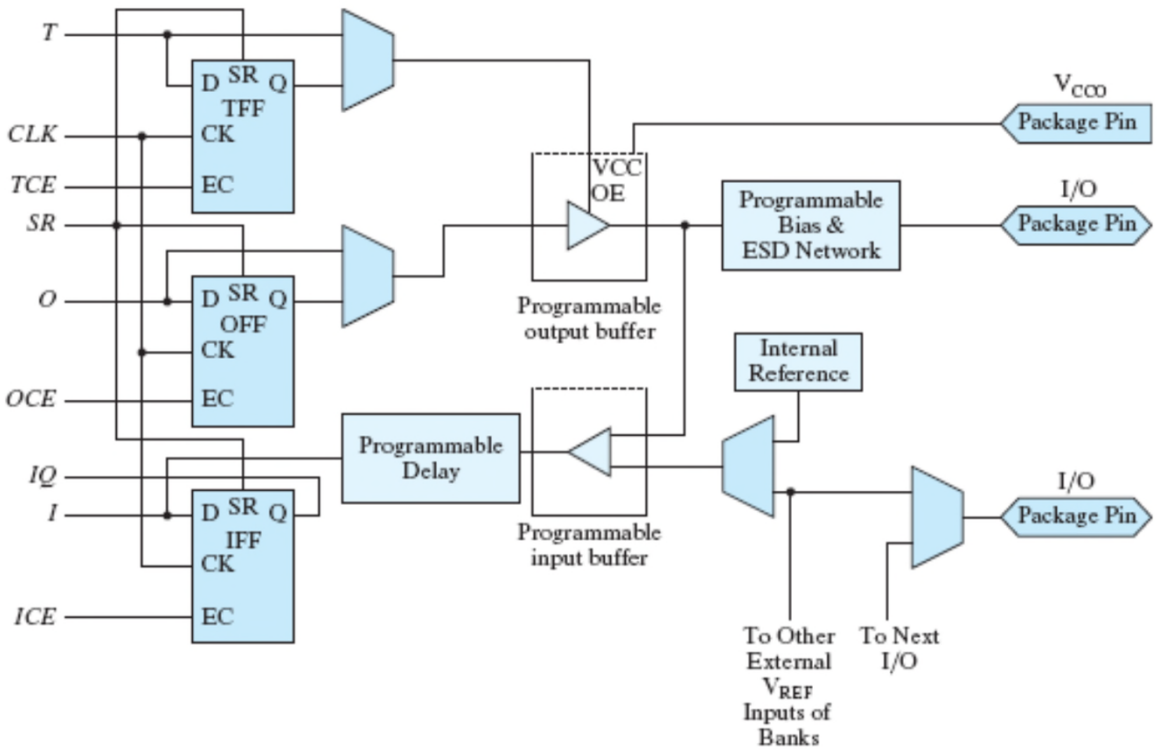
7.8 SEQUENTIAL PROGRAMMABLE DEVICES



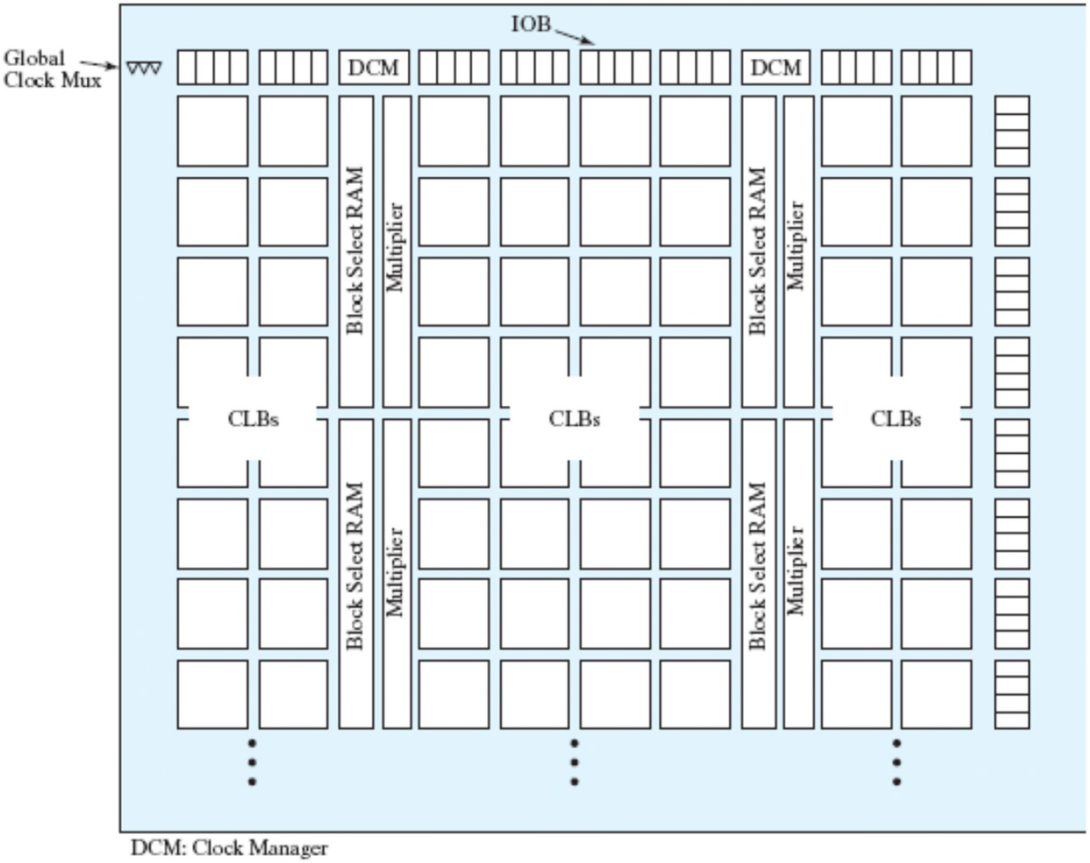
7.8 SEQUENTIAL PROGRAMMABLE DEVICES



7.8 SEQUENTIAL PROGRAMMABLE DEVICES

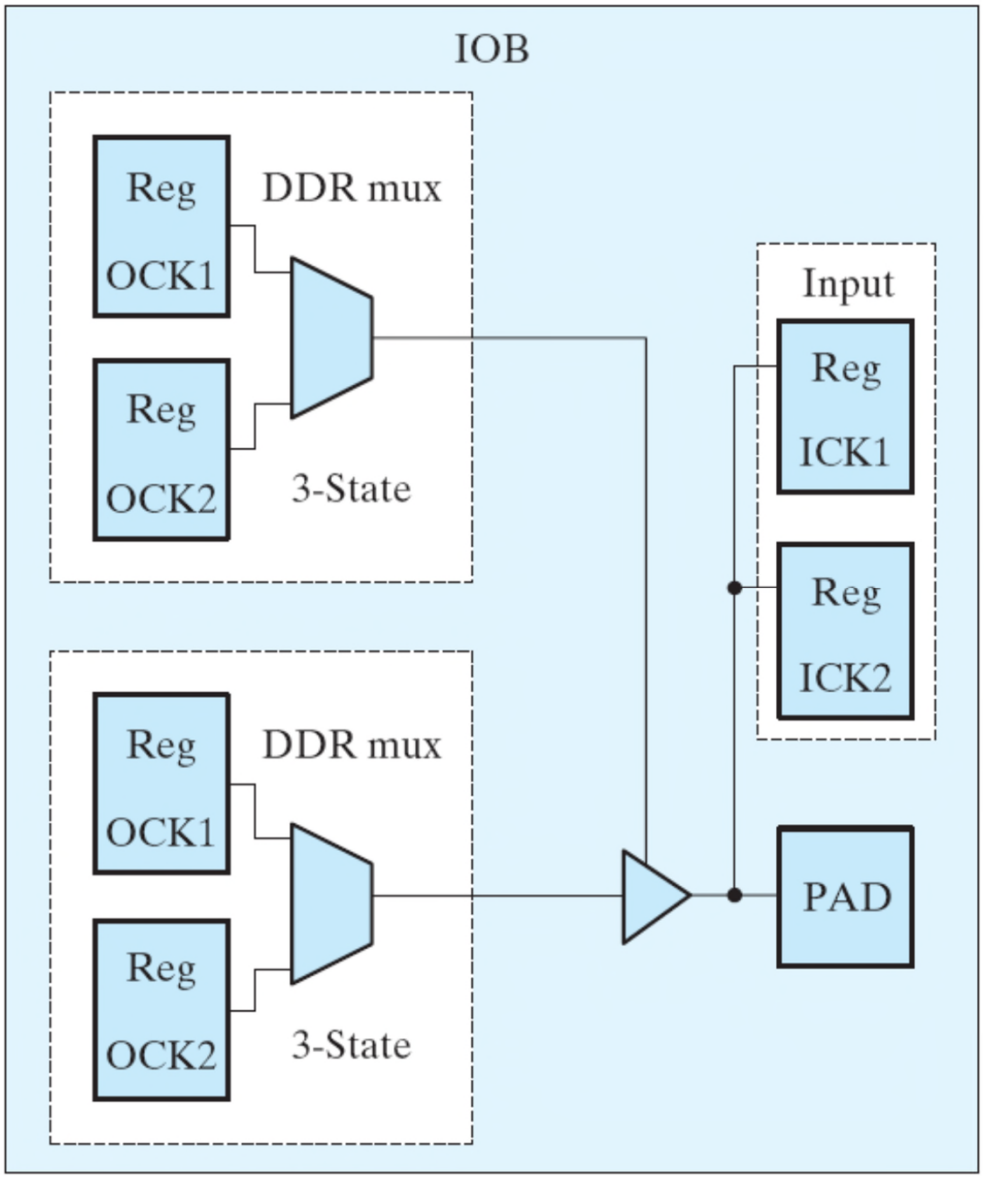


7.8 SEQUENTIAL PROGRAMMABLE DEVICES



DCM: Clock Manager

7.8 SEQUENTIAL PROGRAMMABLE DEVICES



7.8 SEQUENTIAL PROGRAMMABLE DEVICES