Chapter 7
Memory and Programmable Logic

7-1. Introduction

- There are two types of memories that are used in digital systems:

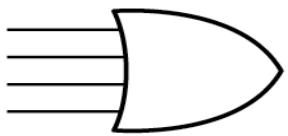
  Random-access memory(RAM): perform both the write and read operations.

  Read-only memory(ROM): perform only the read operation.

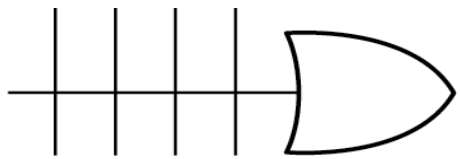
- The read-only memory is a programmable logic device. Other such units are the programmable logic array(PLA), the programmable array logic(PAL), and the field-programmable gate array(FPGA).

Array logic

- A typical programmable logic device may have hundreds to millions of gates interconnected through hundreds to thousands of internal paths.
- In order to show the internal logic diagram in a concise form, it is necessary to employ a special gate symbology applicable to array logic.



(a) Conventional symbol



(b) Array logic symbol

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

7-2. Random-Access Memory

- A memory unit stores binary information in groups of bits called words.

  1 byte = 8 bits

  1 word = 2 bytes

- The communication between a memory and its environment is achieved through data input and output lines, address selection lines, and control lines that specify the direction of transfer.

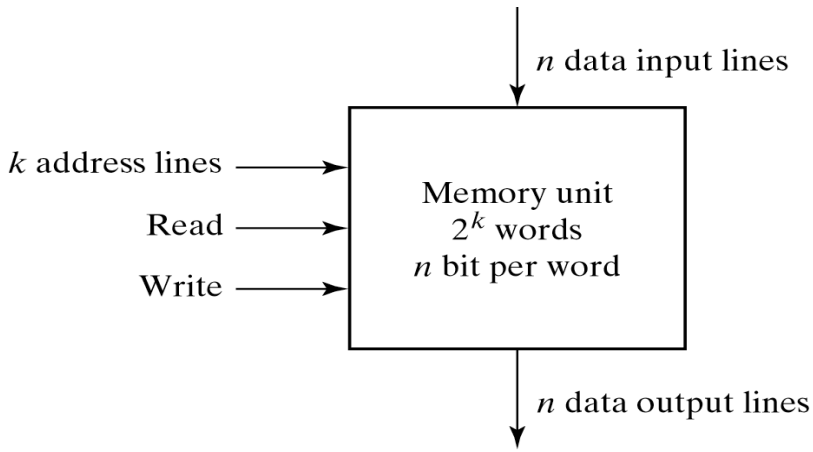


Fig. 7-2 Block Diagram of a Memory Unit

Content of a memory

- Each word in memory is assigned an identification number, called an address, starting from 0 up to 2^k-1, where k is the number of address lines.
- The number of words in a memory with one of the letters K=2^{10}, M=2^{20}, or G=2^{30}.

  64K = 2^{16} 2M = 2^{21}

  4G = 2^{32}

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

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

Write and Read operations

- Transferring a new word to be stored into memory:

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

Write and Read operations

- Transferring a stored word out of memory:

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

- Commercial memory sometimes provide the two control inputs for reading and writing in a somewhat different configuration in table 7-1.

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

Memory description in HDL

A memory of 1024 words with 16-bits per word is declared as

reg [15:0] memword[0:1023];

$\overline{\text{Read/Write}}$ = 1

DataOut ← Mem[Address];

$\overline{\text{Read/Write}}$ =0

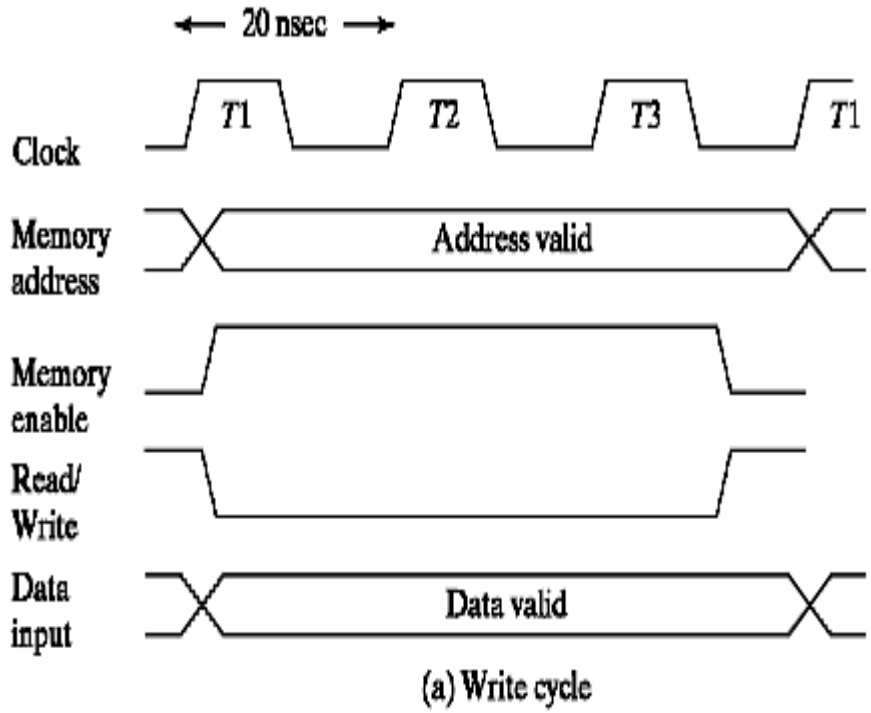
Mem[Address] ← DataIn;

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
```

Timing Waveforms (write)

- The access time and cycle time of the memory must be within a time equal to a fixed number of CPU clock cycles.
- The memory enable and the read/write signals must be activated after the signals in the address lines are stable to avoid destroying data in other memory words.
- Enable and read/write signals must stay active for at least 50ns.



(a) Write cycle

Timing Waveforms (read)

- The CPU can transfer the data into one of its internal registers during the negative transition of T3.

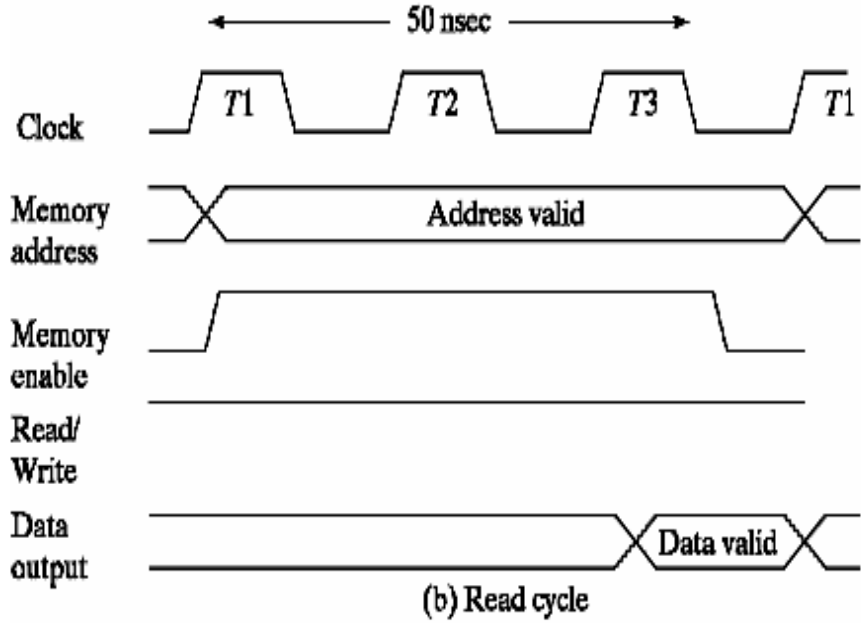


Fig. 7-4 Memory Cycle Timing Waveforms

Types of memories

- In random-access memory, the word locations may be thought of as being separated in space, with each word occupying one particular location.
- In sequential-access memory, the information stored in some medium is not immediately accessible, but is available only certain intervals of time. A magnetic disk or tape unit is of this type.

Types of memories

- In a random-access memory, the access time is always the same regardless of the particular location of the word.
- In a sequential-access memory, the time it takes to access a word depends on the position of the word with respect to the reading head position; therefore, the access time is variable.

Static RAM

- SRAM consists essentially of internal latches that store the binary information.
- The stored information remains valid as long as power is applied to the unit.
- SRAM is easier to use and has shorter read and write cycles.
- Low density, low capacity, high cost, high speed, high power consumption.

Dynamic RAM

- DRAM stores the binary information in the form of electric charges on capacitors.
- The capacitors are provided inside the chip by MOS transistors.
- The capacitors tends to discharge with time and must be periodically recharged by refreshing the dynamic memory.

Dynamic RAM

- DRAM offers reduced power consumption and larger storage capacity in a single memory chip.
- High density, high capacity, low cost, low speed, low power consumption.

Types of memories

- Memory units that lose stored information when power is turned off are said to be volatile.
- Both static and dynamic, are of this category since the binary cells need external power to maintain the stored information.
- Nonvolatile memory, such as magnetic disk, ROM, retains its stored information after removal of power.

7-3. Memory decoding

- The equivalent logic of a binary cell that stores one bit of information is shown below.

  Read/$\overline{\text{Write}}$ = 0, select = 1, input data to S-R latch

  Read/$\overline{\text{Write}}$ = 1, select = 1, output data from S-R latch

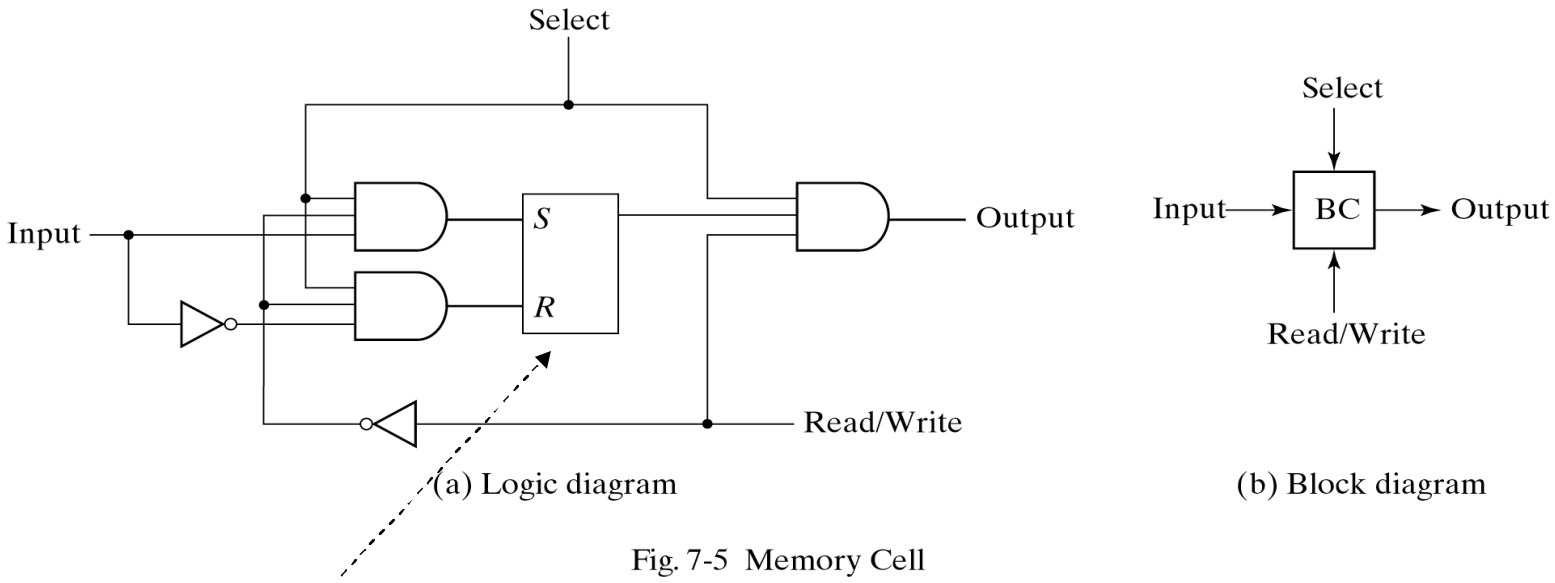


Fig. 7-5 Memory Cell

SR latch with NOR gates

4X4 RAM

- There is a need for decoding circuits to select the memory word specified by the input address.
- 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.

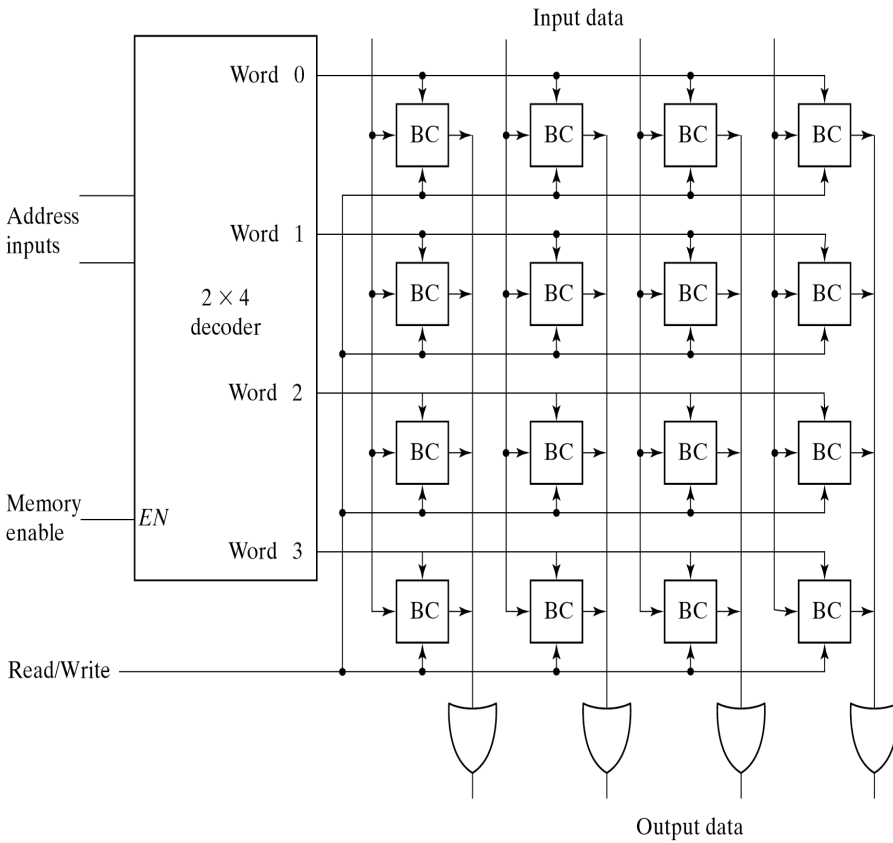


Fig. 7-6 Diagram of a 4 × 4 RAM

- A memory with 2^k words of n bits per word requires k address lines that go into $kx2^k$ decoder.

Coincident decoding

- A decoder with k inputs and 2^k outputs requires 2^k AND gates with k inputs per gate.
- Two decoding in a two-dimensional selection scheme can reduce the number of inputs per gate.
- 1K-word memory, instead of using a single 10X1024 decoder, we use two 5X32 decoders.

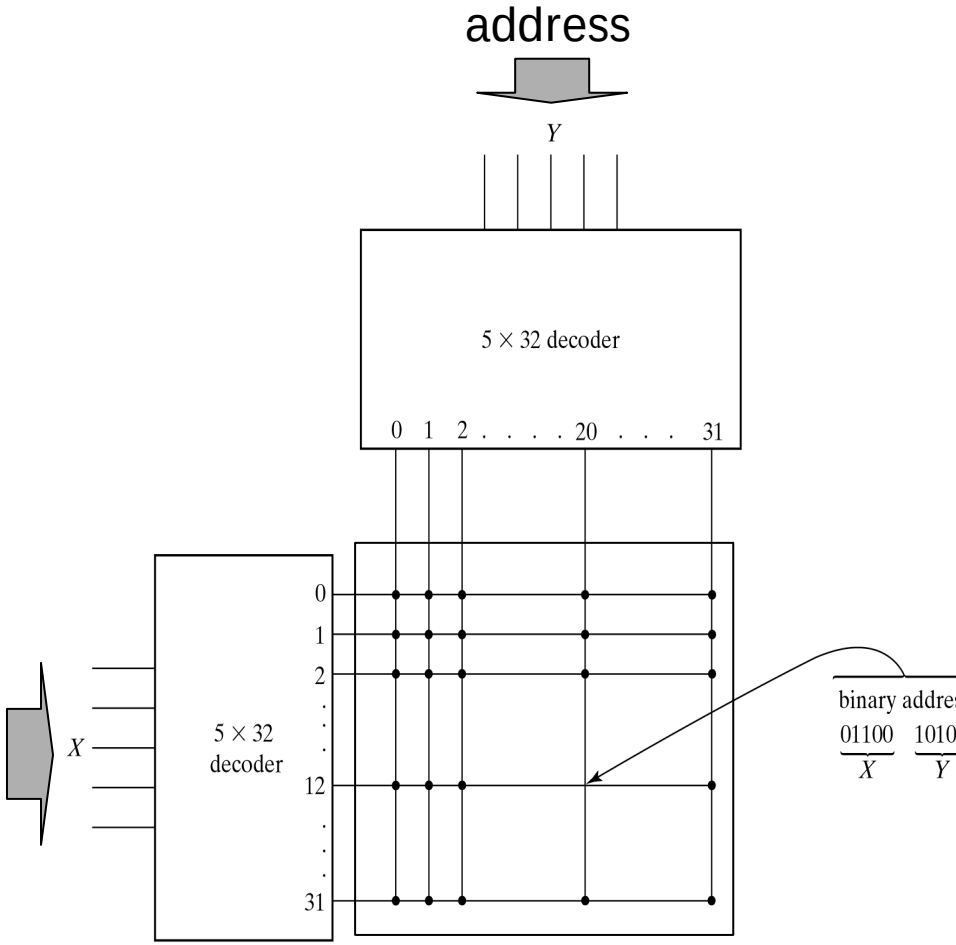


Fig. 7-7 Two-Dimensional Decoding Structure for a 1K-Word Memory

Address multiplexing

- DRAMs typically have four times the density of SRAM.

- The cost per bit of DRAM storage is three to four times less than SRAM. Another factor is lower power requirement.

Address multiplexing

- Address multiplexing will reduce the number of pins in the IC package.

- In a two-dimensional array, the address is applied in two parts at different times, with the row address first and the column address second. Since the same set of pins is used for both parts of the address, so can decrease the size of package significantly.

Address multiplexing for 64K DRAM

- After a time equivalent to the settling time of the row selection, RAS goes back to the 1 level.
- Registers are used to store the addresses of the row and column.
- CAS must go back to the 1 level before initialing another memory operation.

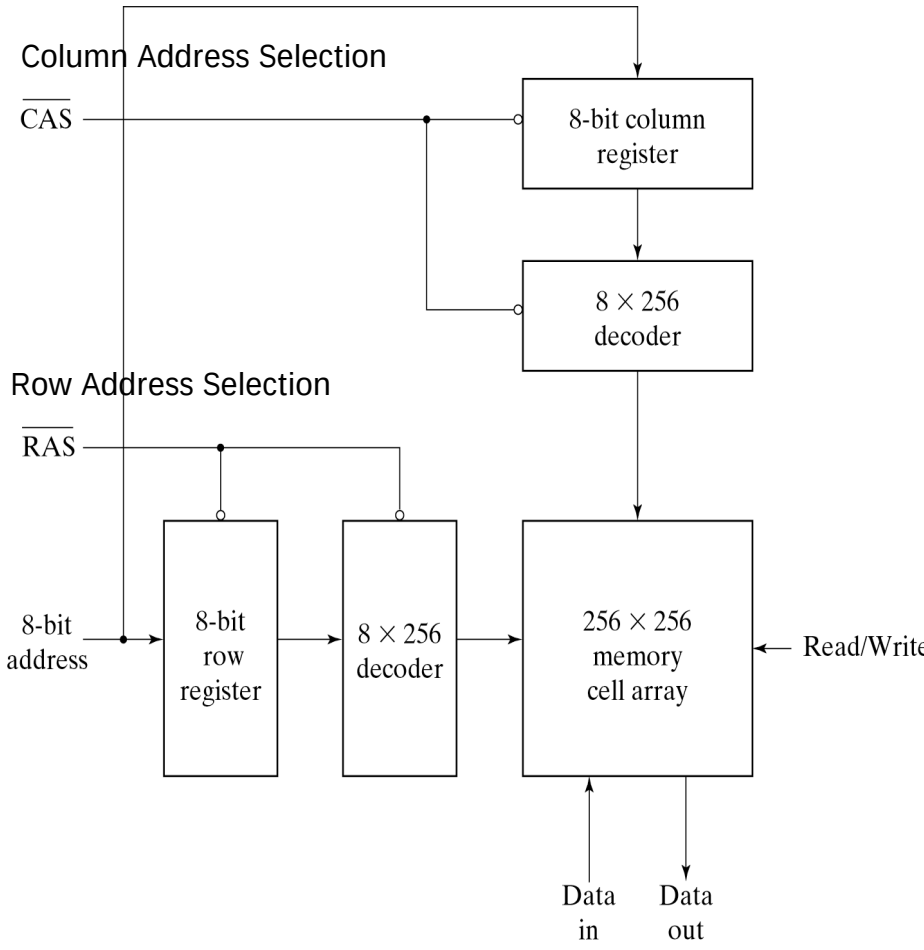


Fig. 7-8 Address Multiplexing for a 64K DRAM

7-4. Error detection and correction

- It is protecting the occasional errors in storing and retrieving the binary information.
- Parity can be checked the error, but it can’t be corrected.
- An error-correcting code generates multiple parity check bits that are stored with the data word in memory.

Hamming Code

- One of the most common used in RAM was devised by R. W. Hamming (called Hamming code).
- In Hamming code:

  k = parity bits in n-bit data word,

  forming a new word of n + k bits. Those positions numbered as a power of 2 are reserved for the parity bits.

  the remaining bits are the data bits.

Hamming Code

Ex. Consider the 8-bit data word 11000100. we include four parity bits with it and arrange the 12 bits as follows:

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) = $1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$

P_2 = XOR of bits(3,6,7,10,11) = $1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$

P_4 = XOR of bits(5,6,7,12) = $1 \oplus 0 \oplus 0 \oplus 0 = 1$

P_8 = XOR of bits(9,10,11,12) = $0 \oplus 1 \oplus 0 \oplus 0 = 1$

Hamming Code

- The data is stored in memory together with the parity bit as 12-bit composite word.

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 read from memory, the parity is checked over the same combination of bits including the parity bit.

C_1 = XOR of bits(3,5,7,9,11)

C_2 = XOR of bits(3,6,7,10,11)

C_4 = XOR of bits(5,6,7,12)

C_8 = XOR of bits(9,10,11,12)

Error-Detection

- A 0 check bit designates an even parity over the checked bits and a 1 designates an odd parity.
- Since the bits were stored with even parity, the result,

  $C = C_8C_4C_2C_1 = 0000$, indicates that no error has occurred.
- If $C \neq 0$, then the 4-bit binary number formed by the check bits gives the position of the erroneous bit.

Example

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

- Evaluating the XOR of the corresponding bits, get the four check bits

	C_8	C_4	C_2	C_1
For no error:	0	0	0	0
with error in bit 1:	0	0	0	1
with error in bit 5:	0	1	0	1

Hamming Code

- The Hamming Code can be used for data words of any length.
- Total bit in Hamming Code is n + k bits, the syndrome value C consists of k bits and has a range of 2^k value between 0 and 2^k - 1. the range of k must be equal to or greater than n + k, giving the relationship

$$2^k - 1 = n + k$$

Table 7-2
Range of Data Bits for k Check Bits

Number of Check Bits, *k*	Range of Data Bits, *n*
3	2-4
4	5-11
5	12-26
6	27-57
7	58-120

Single-Error correction, Double-Error detection

- The Hamming Code can detect and correct only a single error.
- By adding another parity bit to the coded word, the Hamming Code can be used to correct a single error and detect double errors. Becomes $001110010100P_{13}$.

$$001110010100\ P_{13} \rightarrow 001110010100\ 1$$

$$P = XOR(\ 001110010100\ 1\ )$$

if P = 0, the parity is correct (even parity), but if P = 1, then the parity over the 13 bits is incorrect (odd parity).

the following four cases can occur:

Single-Error correction, Double-Error detection

1. If C = 0 and P = 0, no error occurred
2. If C ≠ 0 and P = 1, a single error occurred that can be corrected
3. If C ≠ 0 and P = 0, a double error occurred that is detected but that cannot be corrected
4. If C = 0 and P = 1, an error occurred in the P_{13} bit

7-5. Read-Only Memory

- A block diagram of a ROM is shown below. It consists of k address inputs and n data outputs.
- The number of words in a ROM is determined from the fact that k address input lines are needed to specify 2^k words.

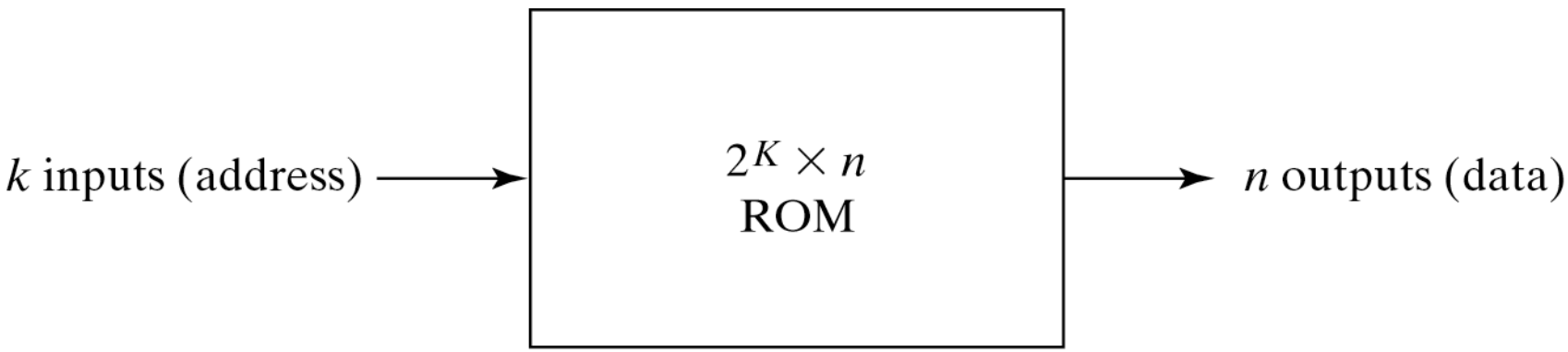


Fig. 7-9 ROM Block Diagram

Construction of ROM

- Each output of the decoder represents a memory address.
- Each OR gate must be considered as having 32 inputs.
- A 2^k X n ROM will have an internal k X 2^k decoder and n OR gates.

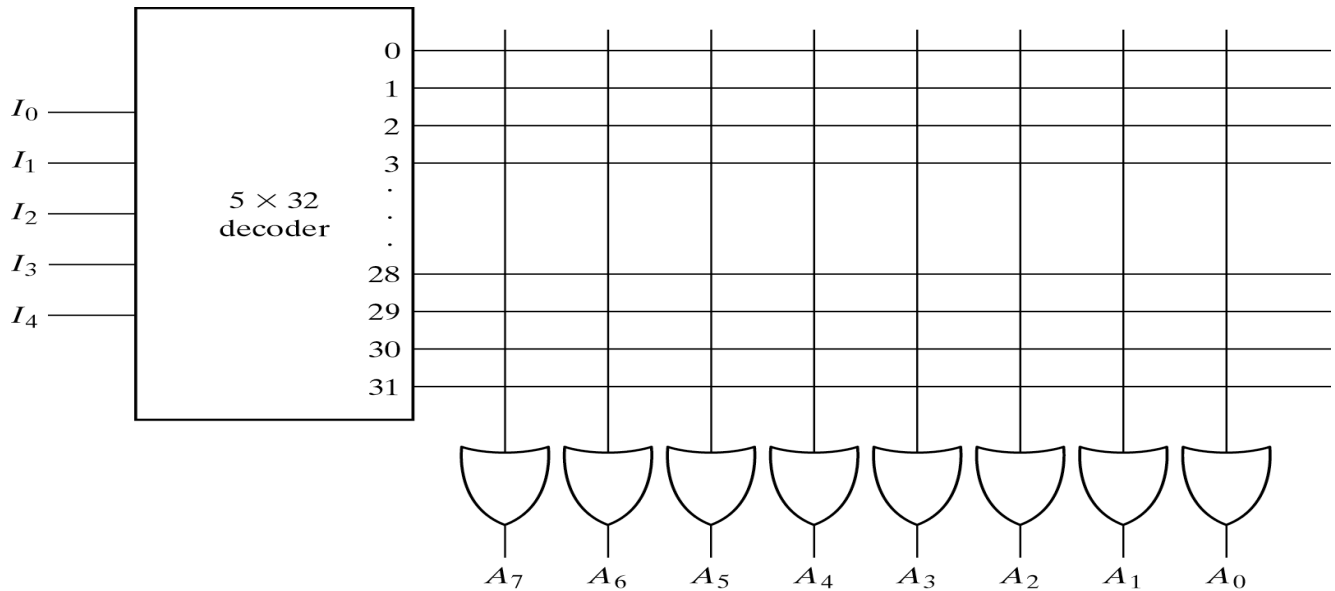


Fig. 7-10 Internal Logic of a 32 × 8 ROM

Truth table of ROM

- A programmable connection between to lines is logically equivalent to a switch that can be altered to either be close or open.
- Intersection between two lines is sometimes called a cross-point.

Table 7-3
ROM Truth Table (Partial)

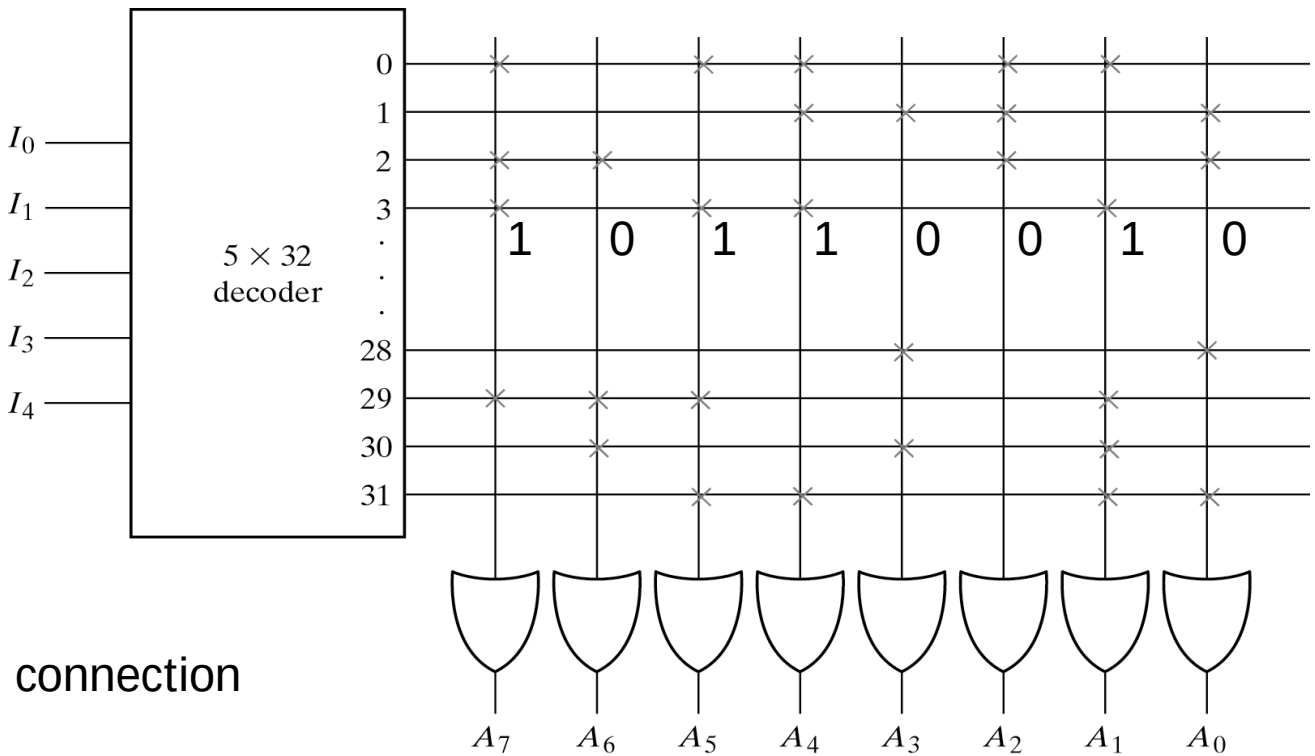
Inputs					Outputs							
I4	I3	I2	I1	I0	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		⋮					⋮					
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

Programming the ROM

In Table 7-3, 0 → no connection

1 → connection

Address 3 = 10110010 is permanent storage using fuse link



X : means connection

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

Combinational circuit implementation

- The internal operation of a ROM can be interpreted in two way: First, a memory unit that contains a fixed pattern of stored words. Second, implements a combinational circuit.
- Fig. 7-11 may be considered as a combinational circuit with eight outputs, each being a function of the five input variables.

$$A_7(I_4, I_3, I_2, I_1, I_0) = \Sigma(0,2,3_{\ldots},29)$$

Sum of minterms

In Table 7-3, output A_7

Example

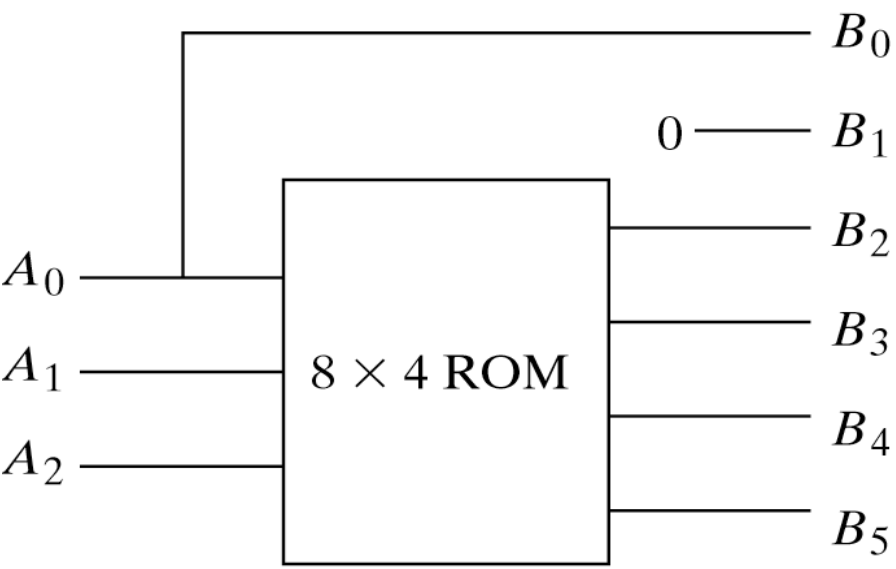
- Design a combinational circuit using a ROM. The circuit accepts a 3-bit number and generates an output binary number equal to the square of the input number.

Derive truth table first

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

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

Example



(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

Types of ROMs

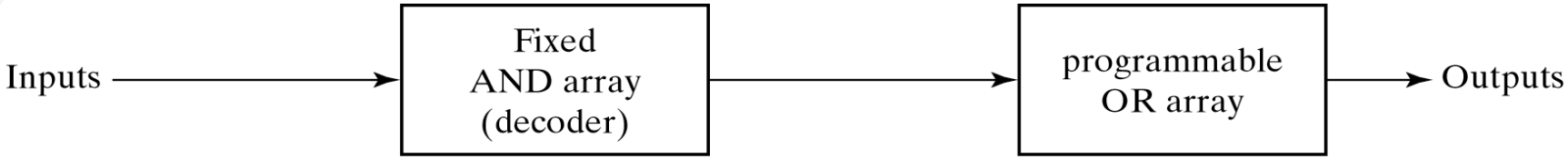
- The required paths in a ROM may be programmed in four different ways.

1. Mask programming: fabrication process
2. Read-only memory or PROM: blown fuse /fuse intact
3. Erasable PROM or EPROM: placed under a special ultraviolet light for a given period of time will erase the pattern in ROM.
4. Electrically-erasable PROM(EEPROM): erased with an electrical signal instead of ultraviolet light.

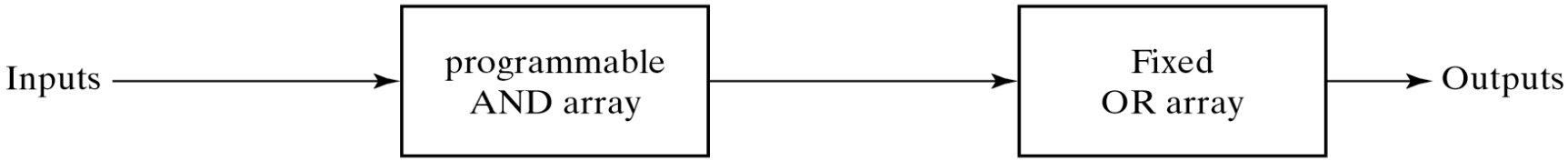
Combinational PLDs

- A combinational PLD is an integrated circuit with programmable gates divided into an AND array and an OR array to provide an AND-OR sum of product implementation.
- PROM: fixed AND array constructed as a decoder and programmable OR array.
- PAL: programmable AND array and fixed OR array.
- PLA: both the AND and OR arrays can be programmed.

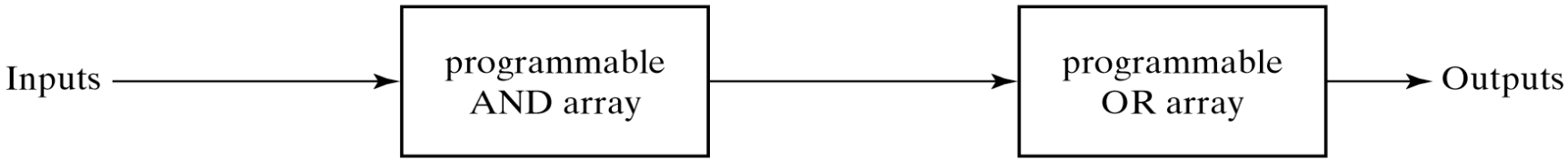
Combinational PLDs



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL)



(c) Programmable logic array (PLA)

Fig. 7-13 Basic Configuration of Three PLDs

7-6. Programmable Logic Array

- Fig.7-14, the decoder in PROM is replaced by an array of AND gates that can be programmed to generate any product term of the input variables.
- The product terms are then connected to OR gates to provide the sum of products for the required Boolean functions.
- The output is inverted when the XOR input is connected to 1 (since $x \oplus 1 = x'$). The output doesn't change and connect to 0 (since $x \oplus 0 = x$).

PLA

F1 = AB'+AC+A'BC'

F2 = (AC+BC)'

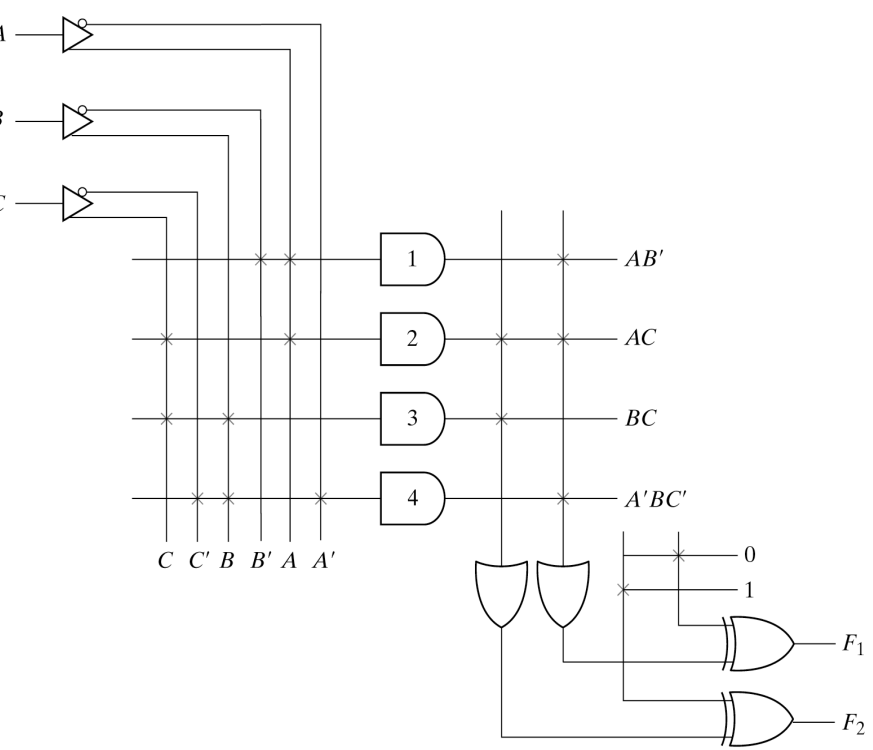


Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs

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	–

Programming Table

1. First: lists the product terms numerically
2. Second: specifies the required paths between inputs and AND gates
3. Third: specifies the paths between the AND and OR gates
4. For each output variable, we may have a T(ture) or C(complement) for programming the XOR gate

Simplification of PLA

- Careful investigation must be undertaken in order to reduce the number of distinct product terms, PLA has a finite number of AND gates.
- Both the true and complement of each function should be simplified to see which one can be expressed with fewer product terms and which one provides product terms that are common to other functions.

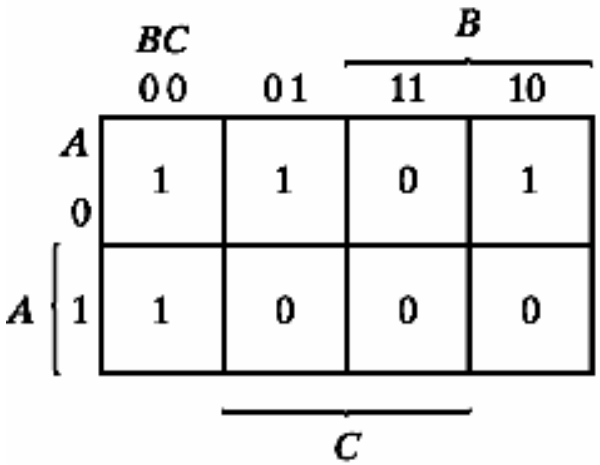
Example 7-2

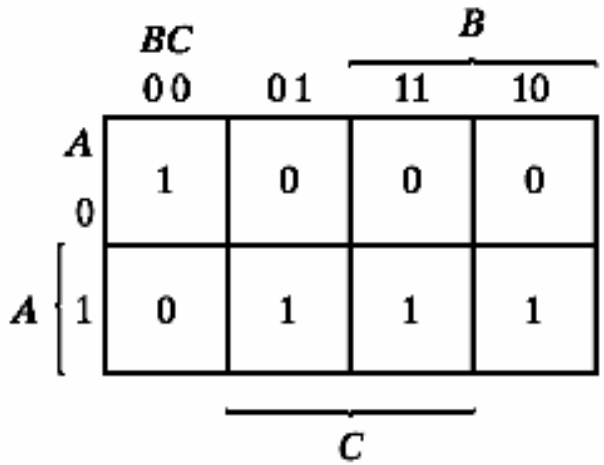
Implement the following two Boolean functions with a PLA:

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

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

The two functions are simplified in the maps of Fig.7-15





1 elements — $F_1 = A'B' + A'C' + B'C'$

0 elements — $F_1 = (AB + AC + BC)'$

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

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

PLA table by simplifying the function

- Both the true and complement of the functions are simplified in sum of products.
- We can find the same terms from the group terms of the functions of F_1, F_1', F_2 and F_2' which will make the minimum terms.

F1 = (AB + AC + BC)'

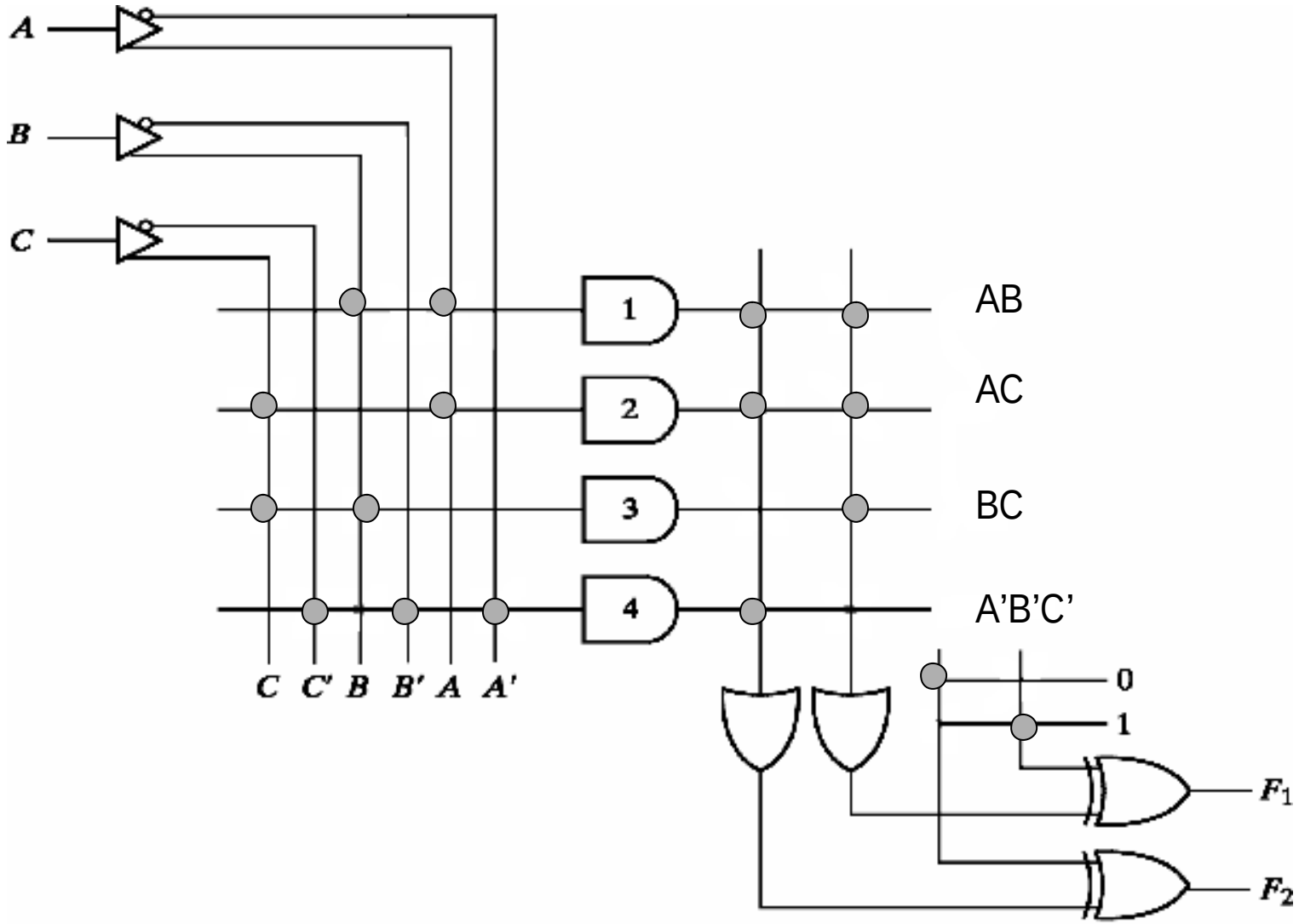
F2 = AB + AC + A'B'C'

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

PLA implementation



7-7. Programmable Array Logic

- The PAL is a programmable logic device with a fixed OR array and a programmable AND array.

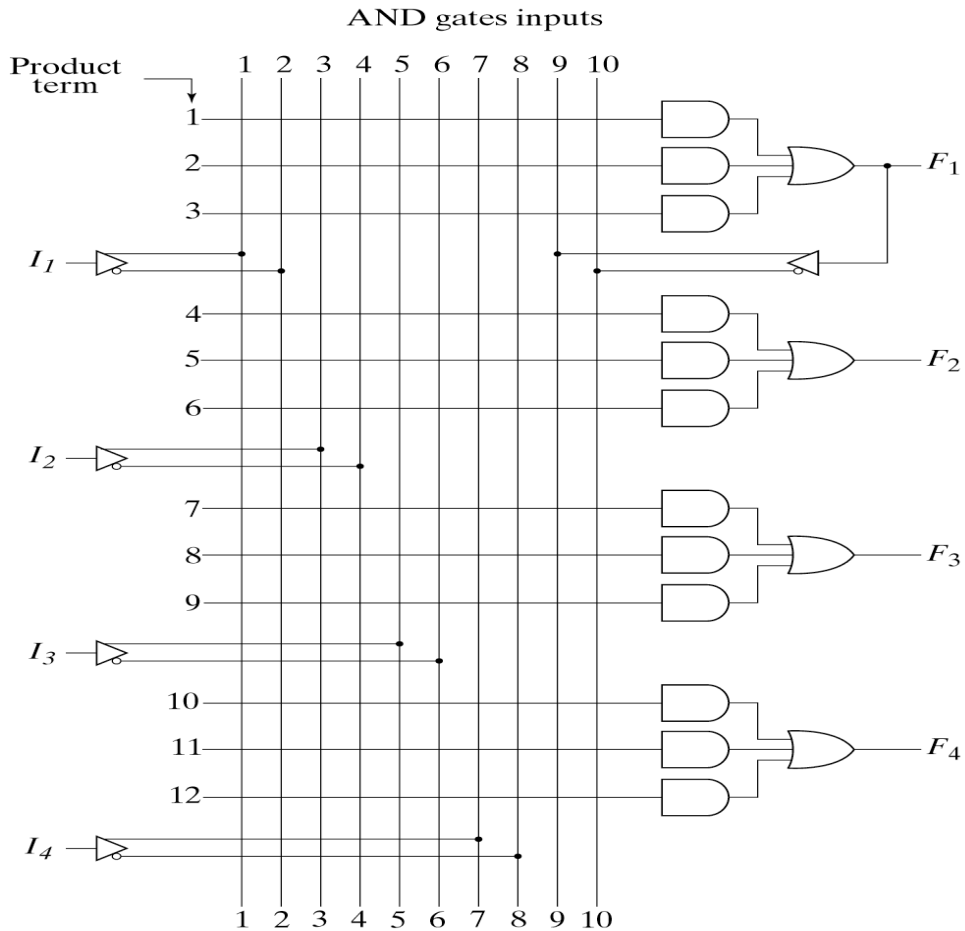


Fig. 7-16 PAL with Four Inputs, Four Outputs, and Three-Wide AND-OR Structure

PAL

- When designing with a PAL, the Boolean functions must be simplified to fit into each section.
- Unlike the PLA, a product term cannot be shared among two or more OR gates. Therefore, each function can be simplified by itself without regard to common product terms.
- The output terminals are sometimes driven by three-state buffers or inverters.

Example

w(A, B, C, D) = ? (2, 12, 13)

x(A, B, C, D) = ? (7, 8, 9, 10, 11, 12, 13, 14, 15)

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

z(A, B, C, D) = ? (1, 2, 8, 12, 13)

Simplifying the four functions as following Boolean functions:

w = ABC' + A'B'CD'

x = A + BCD

w = A'B + CD + B'D'

w = ABC' + A'B'CD' + ACD' + A'B'C'D = w + ACD' + A'B'C'D

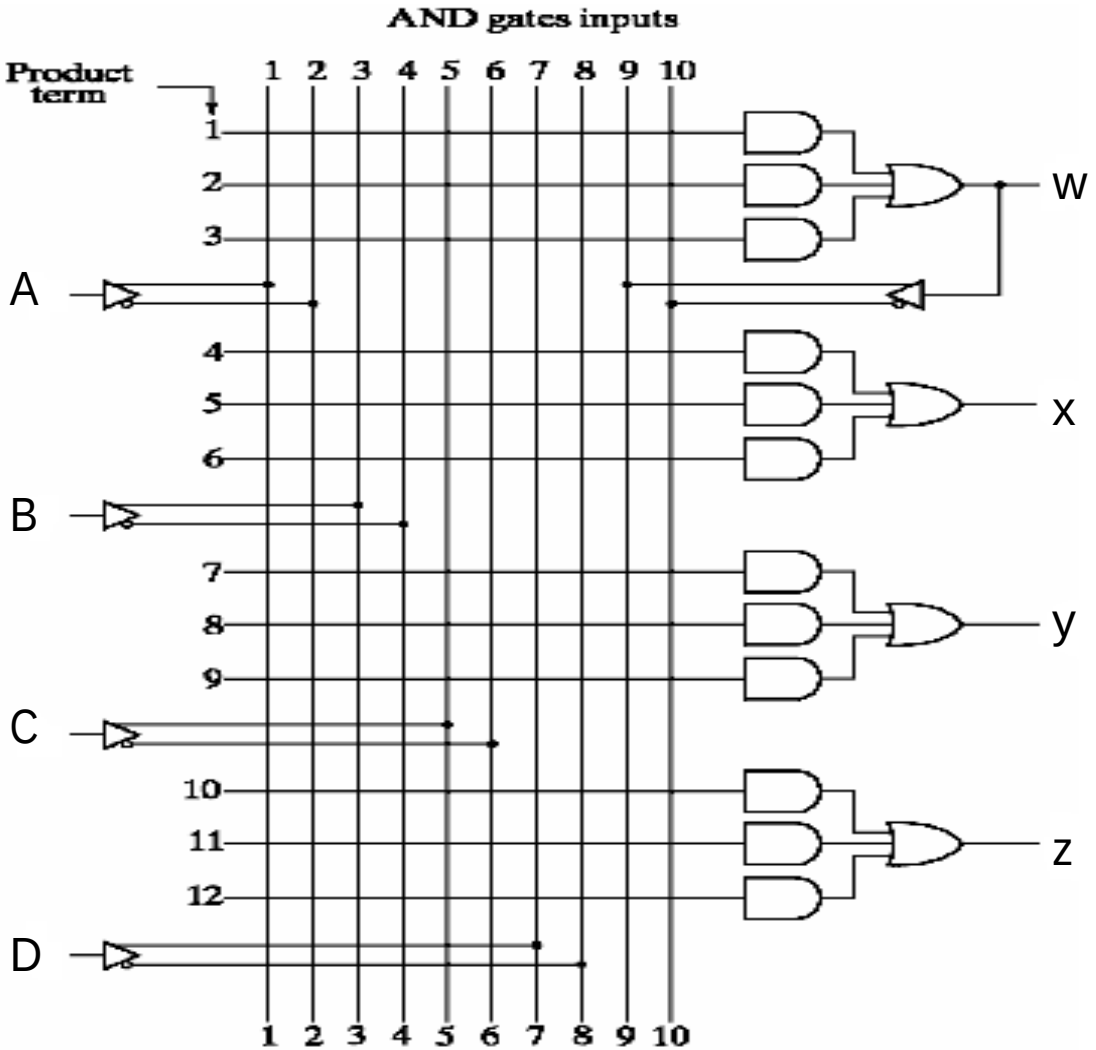
PAL Table

- z has four product terms, and we can replace by w with two product terms, this will reduce the number of terms for z from four to three.

Table 7-6
PAL Programming Table

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

PAL implementation



Fuse map for example

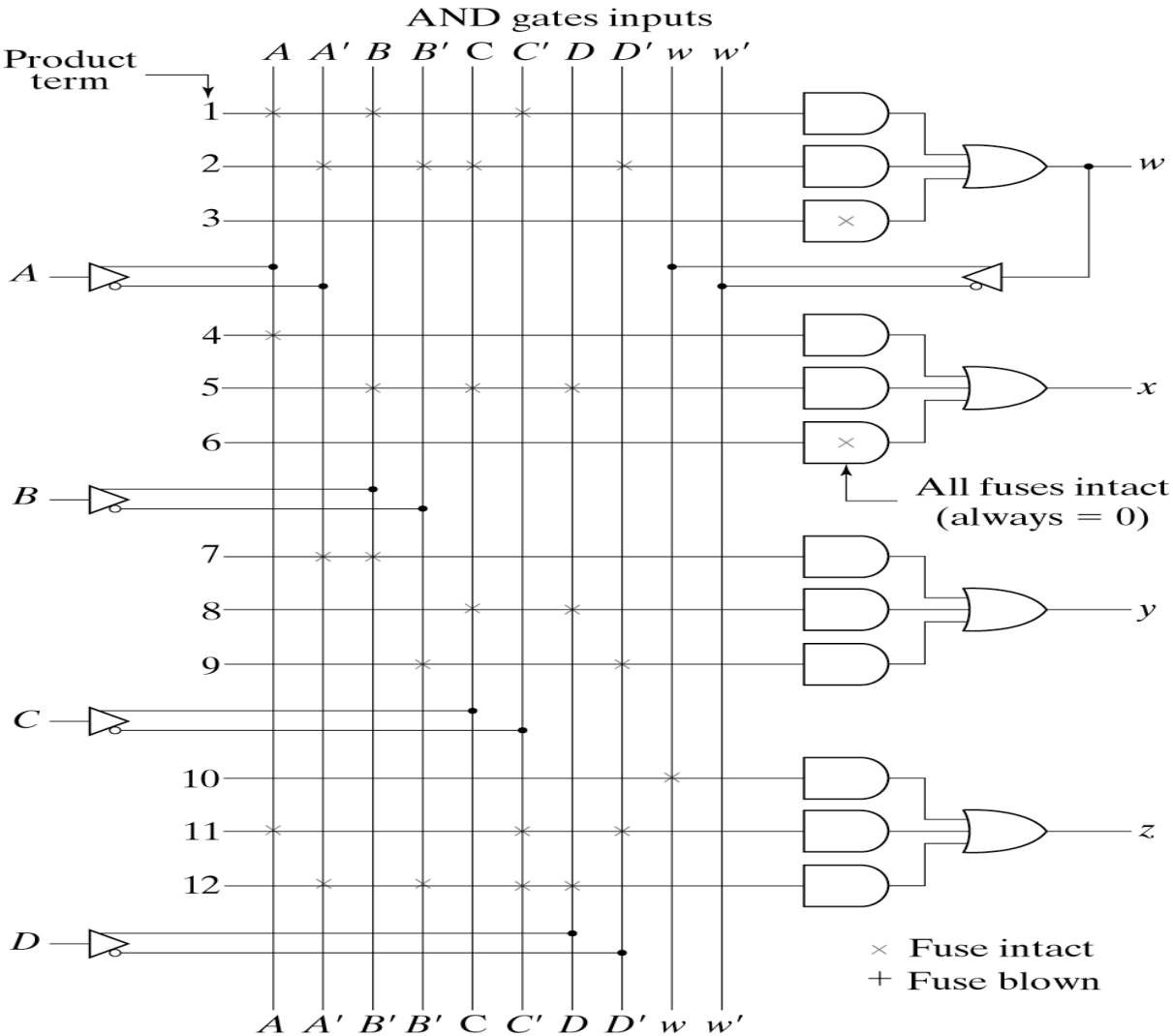


Fig. 7-17 Fuse Map for PAL as Specified in Table 7-6

7-8. Sequential Programmable Devices

- Sequential programmable devices include both gates and flip-flops.
- There are several types of sequential programmable devices, but the internal logic of these devices is too complex to be shown here.
- We will describe three major types without going into their detailed construction.

Sequential Programmable Devices

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

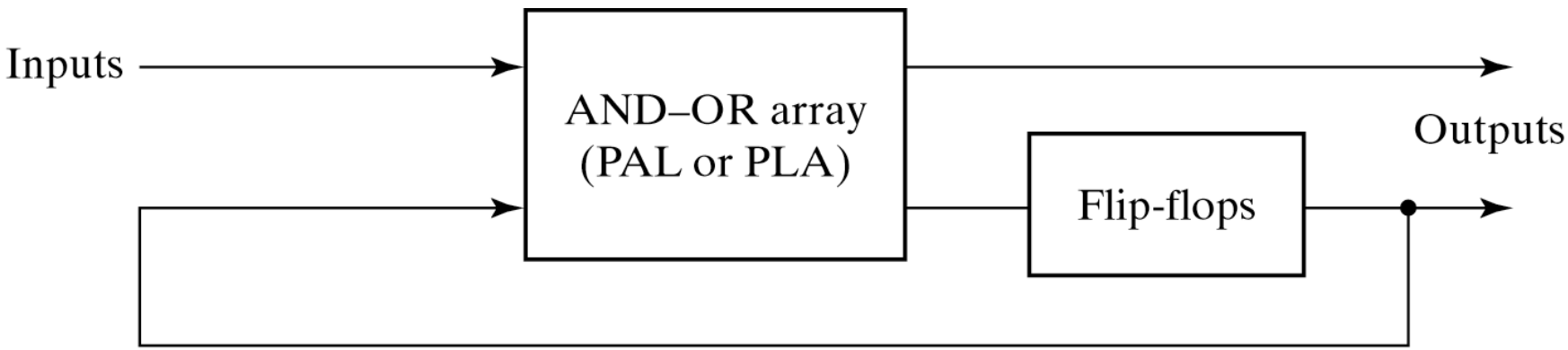


Fig. 7-18 Sequential Programmable Logic Device

FPLS

- The first programmable device developed to support sequential circuit implementation is the field-programmable logic sequencer(FPLS).
- A typical FPLS is organized around a PLA with several outputs driving flip-flops.
- The flip-flops are flexible in that they can be programmed to operate as either JK or D type.
- The FPLS did not succeed commercially because it has too many programmable connections.

SPLD

- Each section of an SPLD is called a macrocell.
- A macrocell is a circuit that contains a sum-of-products combinational logic function and an optional flip-flop.
- We will assume an AND-OR sum of products but in practice, it can be any one of the two-level implementation presented in Sec.3-7.

Macrocell

- Fig.7-19 shows the logic of a basic macrocell.
- The AND-OR array is the same as in the combinational PAL shown in Fig.7-16.

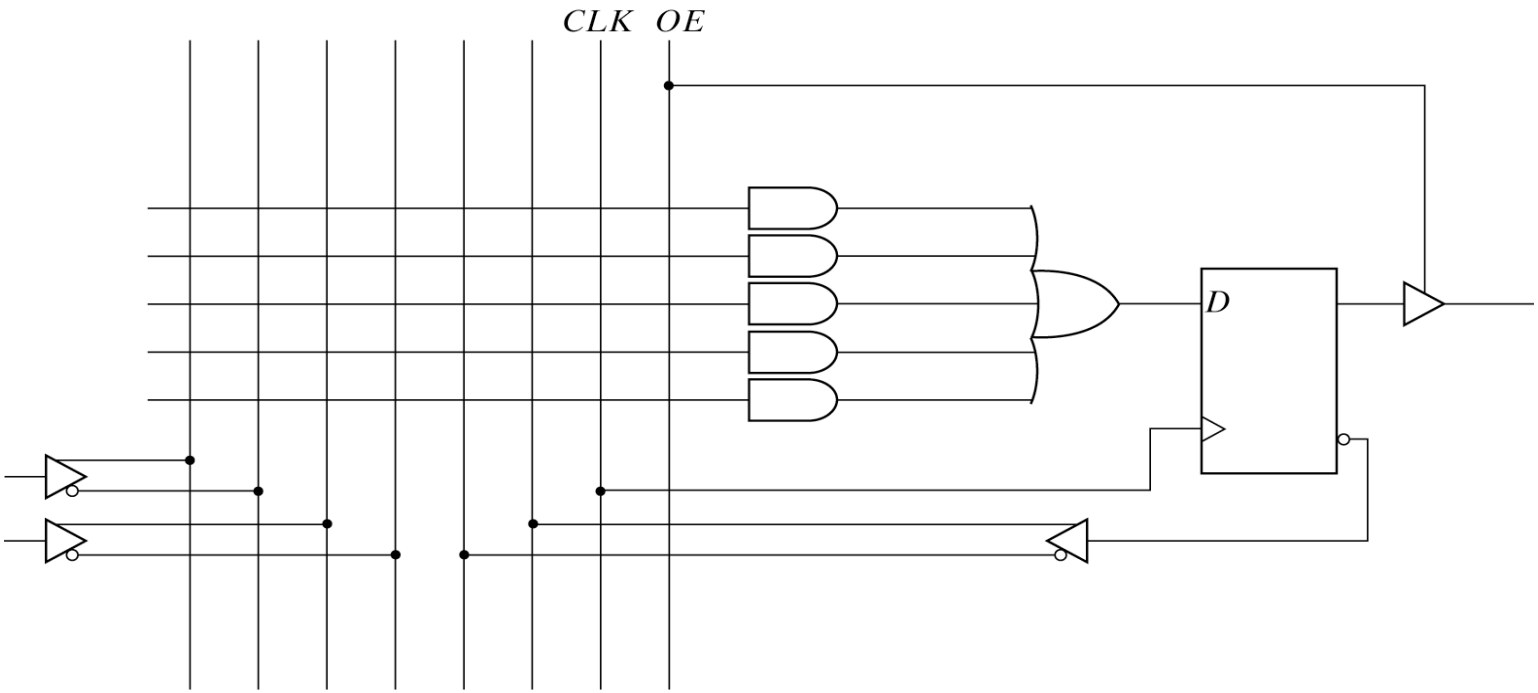


Fig. 7-19 Basic Macrocell Logic

CPLD

- A typical SPLD has from 8 to 10 macrocells within one IC package. All the flip-flops are connected to the common CLK input and all three-state buffers are controlled by the EO input.
- The design of a digital system using PLD often requires the connection of several devices to produce the complete specification. For this type of application, it is more economical to use a complex programmable logic device (CPLD).
- A CPLD is a collection of individual PLDs on a single integrated circuit.

CPLD

- Fig.7-20 shows a general configuration of a CPLD. It consists of multiple PLDs interconnected through a programmable switch matrix. 8 to 16 macrocell per PLD.

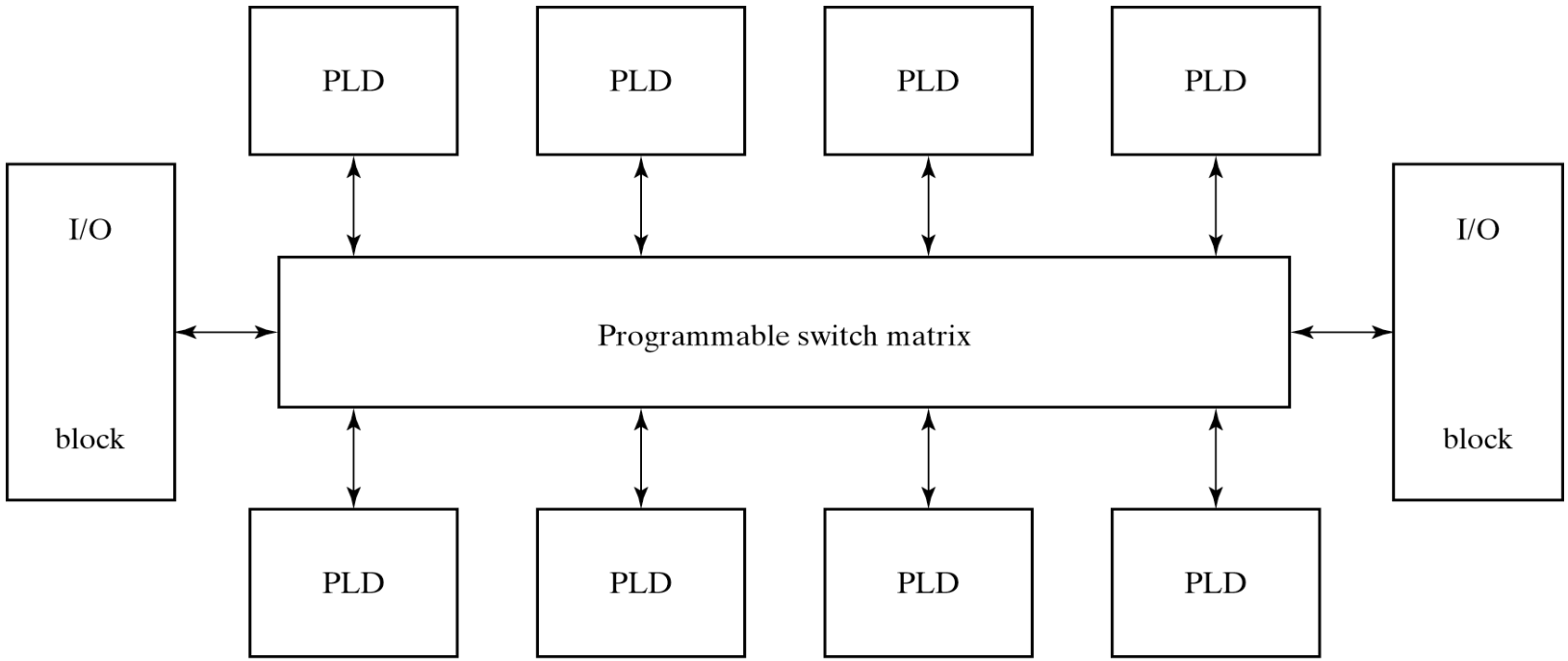


Fig. 7-20 General CPLD Configuration

Gate Array

- The basic component used in VLSI design is the gate array.
- A gate array consists of a pattern of gates fabricated in an area of silicon that is repeated thousands of times until the entire chip is covered with the gates.
- Arrays of one thousand to hundred thousand gates are fabricated within a single IC chip depending on the technology used.

FPGA

- FPGA is a VLSI circuit that can be programmed in the user's location.
- A typical FPGA logic block consists of look-up tables, multiplexers, gates, and flip-flops.
- Look-up table is a truth table stored in a SRAM and provides the combinational circuit functions for the logic block.

Differential of RAM and ROM in FPGA

- The advantage of using RAM instead of ROM to store the truth table is that the table can be programmed by writing into memory.
- The disadvantage is that the memory is volatile and presents the need for the look-up table content to be reloaded in the event that power is disrupted.