7 PROGRAMMABLE LOGIC DEVICES

7.1. LOGIC DEVICES

Logic devices are the circuits which perform specific functions. Logic devices can be classified into two broad categories i.e.,

(*i*) Fixed logic devices
(*ii*) Programmable logic devices

Fixed Logic Devices (FLD)

Devices in which circuits are permanent/fixed. These devices perform one function or set of functions once these devices are manufactured, there function cannot be changed.

Programmable Logic Devices (PLD)

Devices in which circuits are not permanent and can be changed at any time to perform any number of functions. This can be done by using programming.

In This Chapter

- Logic Devices
- Why to go for PLD
- Structure for PLD
- Read Only Memory (ROM)
- Application of ROM
- Designing of ROM
- PLA (Programming Logic Array)
- PAL (Programmable Array Logic)
- FPGA (Field Programmable Gate Array)
- Complex Programmable Logic Devices (CPLDs)
- RAM (Random Access Memory)

7.2. WHY TO GO FOR PLD

In case of fixed logic devices the time required to go from

$$\text{Design} \overset{\text{To}}{\Rightarrow} \text{Prototype} \overset{\text{To}}{\Rightarrow} \text{Final manufacturing}$$

run can take from several months to more than one year and if device doesnot work properly or if the requirement changes then a new design must be developed. This problem is solved in PLDs.

In **case of PLD**, if a user/customer makes a mistake in design/deficiency left out in certain design.

Or

If specification requirements for a project changes.

Or

If a user wants to go for another application.

Or

Whatever may be the reason, then user need to erase the function of that PLD and represent a new function.

These are the reasons why to go for PLDs. Thats why we use PLDs for implementation of combination logic circuits. PLDs allow designers more flexibility to experiment with designs. As these ICs can be reprogrammed in **seconds** and this helps in cost **reduction** also.

Definition of PLD

PLDs are ICs with a large number of gates and flip flops that can be configured/Programmed through software to perform a specific logic function.

PLD as a Black box

It is shown below:

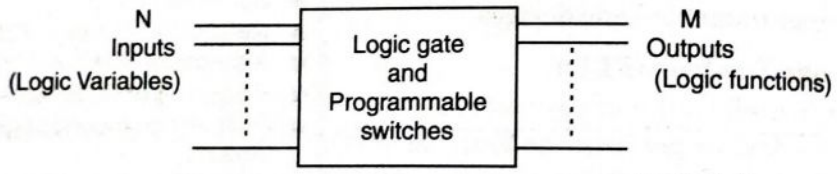


Fig. 7.1.

We can implement any combinational logic circuit with PLD in SOP form which is AND-OR implementation.

7.3. GENERAL STRUCTURE OF PLD (FOR GATE LEVEL DESIGN)

As shown in the PLD black box, it has *N* input variables and *M* output functions. In general structure of PLD, '*N*' input variables are represented by input buffers inverters and '*M*' output functions are represented by OR array.

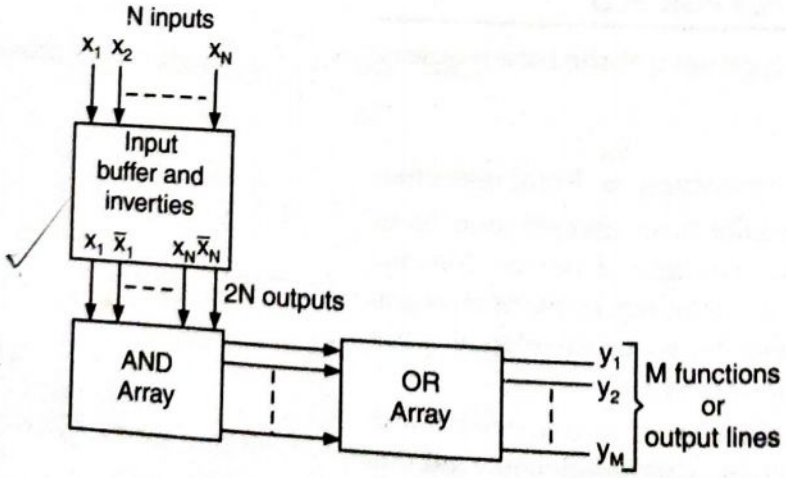


Fig. 7.2.

PLDs consists of three important parts :

(1) Input buffers and inverter

(2) AND array

(3) OR array

(1) Input Buffers and Inverters Block

They are used to provide the true values of the inputs as well as the complemented values of the inputs. For '*N*' input variables *i.e.*, $x_1, x_2 .. x_N$, this block will produce **2 N output lines** *i.e.*, twice of input variables.

(2) AND Array

It takes 2N input variables from input buffers and inverters block and provide product terms of input variables (in both true and complemented form) AND array can provide maximum 2^N product terms (where *N* is the number of input terms) AND array is followed by OR array.

(3) OR array

OR gates present in the OR array logically sum these product terms and thereby generate an SOP expression.

PLD as a part of a logic circuit resides with other chips on a PCB (i.e, printed circuit board). PLD has to be removed from PCB for programming purpose. By placing a socket on PCB makes the removal possible. **PLCC (Plastic leaded chip carrier)** is most commonly used package.

Instead of using a programming unit, it would be easier if a chip could be programmed an PCB itself. This type of programming is known as **In system programming**.

7.4. COMPACT/SHORTHAND GRAPH APPROACH IN PLD

It is represented as:

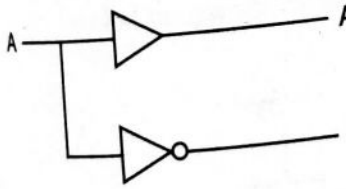


Fig. 7.3.

OR

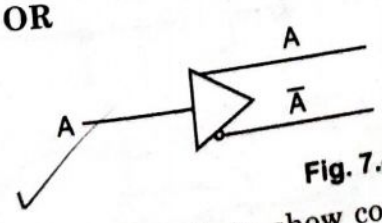


Fig. 7.4.

Let us take an example to show compact/shorthand graph approach in PLDs.

Multiinput AND gate

For this if the output is represented as

$$y = A\bar{B}\bar{C}\,D\bar{E}$$

The shorthand graph approach is given as

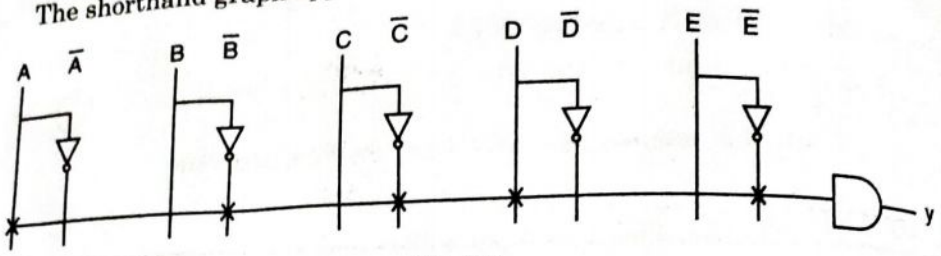


Fig. 7.5.

Here, 'x' indicate the fused link and dots indicate permanent functions which are not Fusible.

Marking 'x' on a PLD diagram is a nice way to visualize which fuses will be blown on left intact.

This function can also be realized by another way. It is shown below.

$$y = A\bar{B}\bar{C}\,D\bar{E}$$

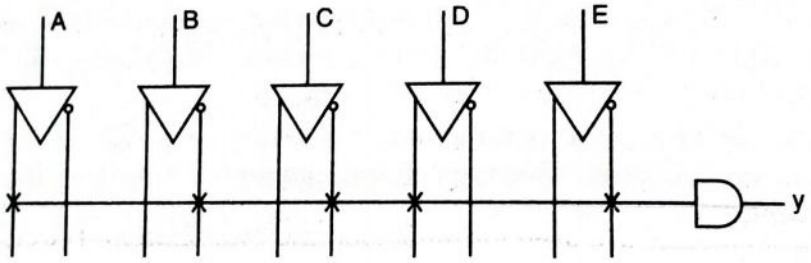


Fig. 7.6.

Multiinput OR gate

Shorthand representation for multiinput OR gate is shown as below

$$y = A + \bar{B}$$

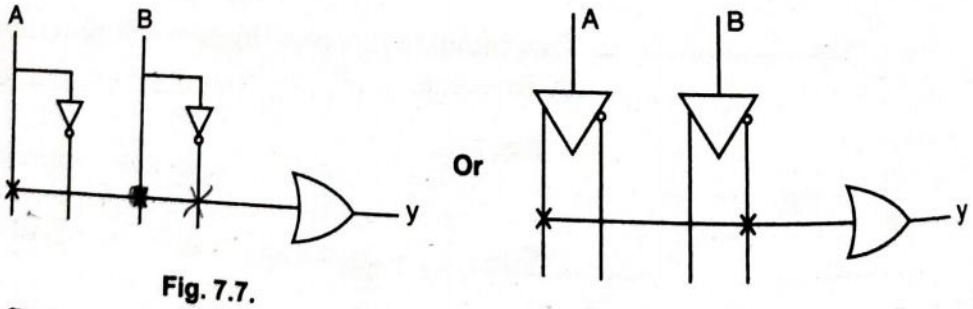


Fig. 7.7.

So for fabricating the PLDs, some of the fuses are programmed to blow out to achieve the desired output from the gate. For example, if desired output of the gate is CD then fuses A and B are to be blown

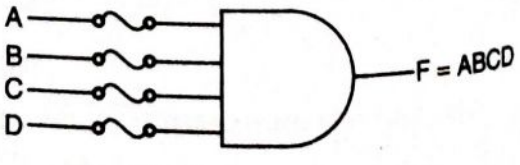


Fig. 7.8. Before programming

Programmable Logic Device

out as shown in figure
programming, the same

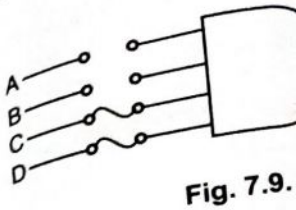


Fig. 7.9.

7.5. ADVANTAGES O

PLDs are those device
are the advantages :

(1) Reduced spa
(2) Reduced pow
(3) Design secu
(4) Compact cir
(5) Short desig
(6) Low develo
(7) Low produc
(8) High switc

7.6. COMBINATI THEIR IMPL

Following technol

(1) ROM (Read

It is presente

Inputs

It stores pe
cannot be altere
"Its a SOP l
It is basically a c
In this case info

(2) PLA (Prog

It is shown

Inputs Progra
Con

out as shown in figure. Therefore, with the blowing of fuses with proper programming, the same gate can generate several Boolean functions.

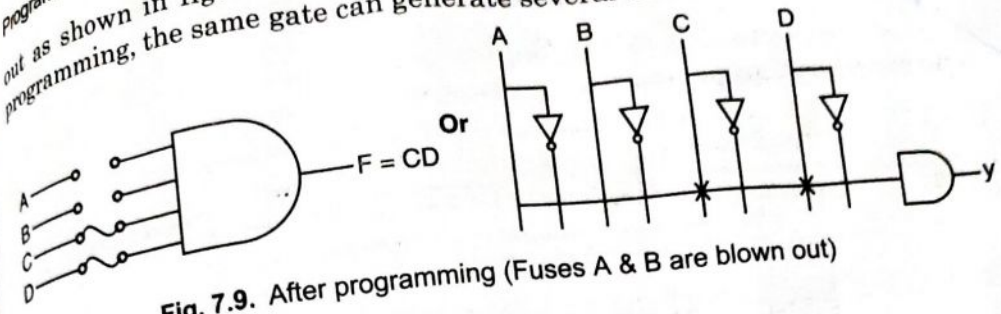


Fig. 7.9. After programming (Fuses A & B are blown out)

7.5. ADVANTAGES OF PLDS

PLDs are those devices in which large circuit is designed on single chip. Following are the advantages :

(1) Reduced space requirements
(2) Reduced power requirements
(3) Design security
(4) Compact circuitry
(5) Short design cycle
(6) Low development cost
(7) Low production cost for large quantity production
(8) High switching speed

7.6. COMBINATIONAL CIRCUITS AND THEIR IMPLEMENTATION TECHNOLOGIES

Following technologies are used for implementing combinational circuit :

(1) ROM (Read only memory)

It is presented as

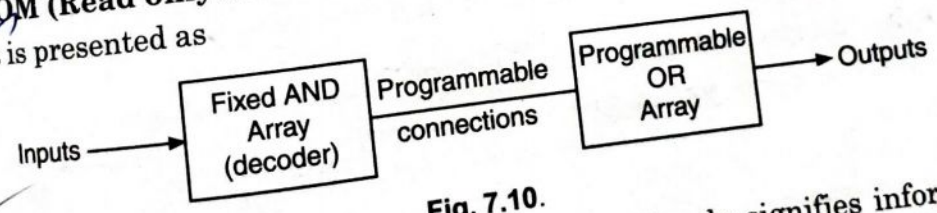


Fig. 7.10.

It stores permanent binary information. Read only signifies information cannot be altered.

"Its a SOP logic device with a fixed AND array and Programmable OR arrays. It is basically a combinational circuit and can be used to implement a logic function. In this case information is specified by designer and physically inserted into PLD.

(2) PLA (Programmable Logic Array)

It is shown below

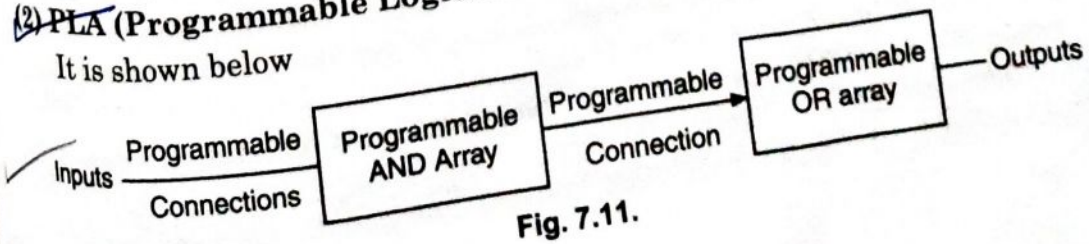


Fig. 7.11.

Its a SOP logic device with a programmable AND array and a programmable OR array.

(3) PAL (Programmable Array Logic)

It is represented as:

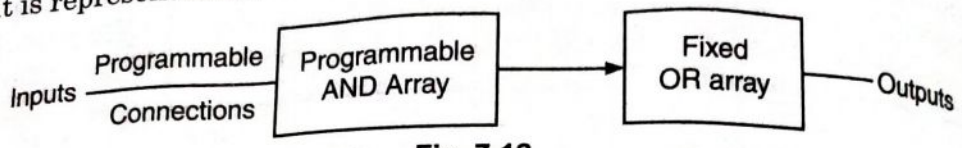


Fig. 7.12.

It is a SOP logic device with a programmable AND array and a fixed OR array.

PLDs can be programmed according to the Boolean functions. The following table shows the programmable array for various devices.

	Devices Type	And Array	OR Array
	Fixed logic devices (Not programmable)	Fixed	Fixed
PLDs	ROM	Fixed	Programmable
	PAL	Programmable	Fixed
	PLA	Programmable	Programmable

7.7 READ ONLY MEMORY (ROM)

A ROM is essentially a memory device for storage purpose in which a fixed set of binary information is stored. An users must first specify the binary information to be stored and then it is embedded in the unit to form the required interconnection pattern. ROM contains special internal links that can be fused or broken. Once a pattern is established for a ROM it remained fixed even if the power supply to the circuit is switched off and then switched on again.

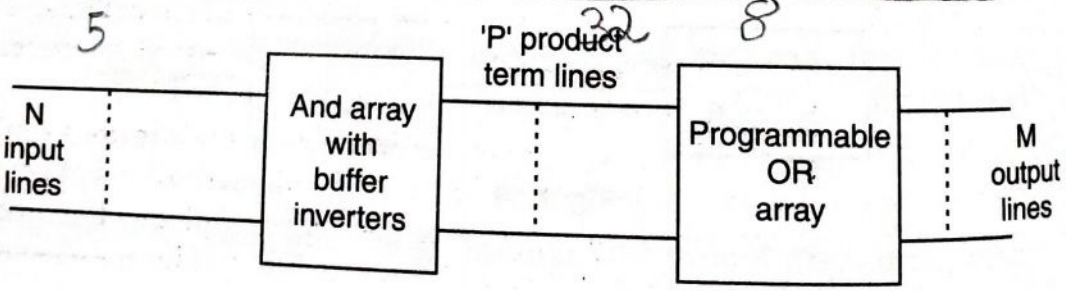


Fig. 7.13. Block diagram of ROM.

It consists of n input lines and m output lines. Each bit combination of input variables is called an address which is denoted by P where $P = 2^n$ and n is the input variable. Each bit combination that is formed at output lines is called a word. Thus an address is essentially a binary number that denotes one of the minterms of n variables and the numbers of bits per word is equal to the number of output lines m.

Therefore, the ROM is denoted by $2^n \times m$ where n is the input variable and m is the output line.

For example, 32×8 ROM includes 8 output line and 32 address lines.

The number of input variables are given as $(2^n) \Rightarrow (32) \Rightarrow (2)^5$. The input is given at 5 line.

In the Fig. 7.4, the block consisting of AND array with buffers/inverters is equivalent to a decoder. The decoder basically is a combinational circuit which generates 2^n number of minterms from n number of input lines. 2^n or p numbers of minterms are realized from n numbers of input variables with help of n numbers of buffers, n numbers of inverters and 2^n numbers of AND gates. Each of the minterms is applied to the inputs of m number of OR gates through fusible links. The logic diagram for ROM is given as

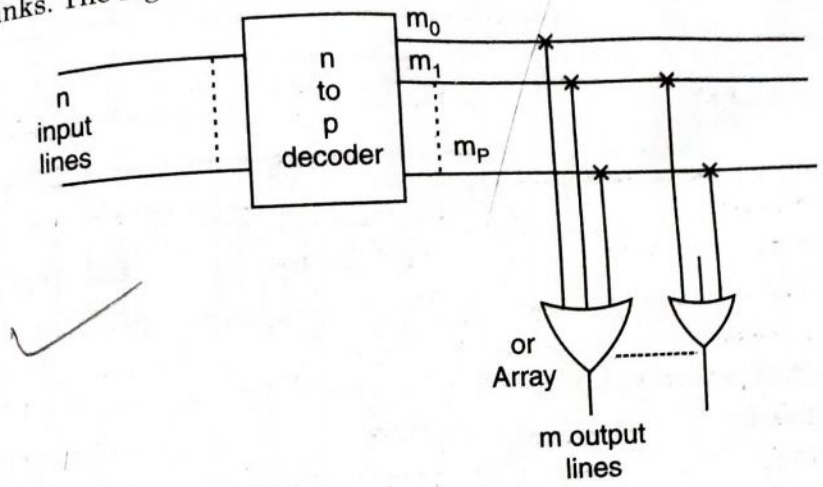


Fig. 7.14.

The advantages of using ROM as a programmable logic device are :

(1) Ease of design since no simplification or minimization of logic function is required.
(2) Design can be changed, modified rapidly.
(3) It is usually faster than discrete SSI/MSI circuit.
(4) Cost is reduced.

Demerit of ROM

The ROM implementation of a function may become quite expensive for functions with a large number of variables because all potential minterms of the functions are implemented whether or not they are needed.

It is removed by using PLA which requires that only the minterms required for a function can be implemented and allows the implementations of several functions simultaneously. Moreover the functions can be implemented directly from their minterms forms (although it is often possible to eliminate some of the minterms, further decreasing the cost of the PLA).

7.7.1. Designing of Circuit Using ROM

EXAMPLE 7.1. Design the following function using ROM

$$F_1 (A, B, C) = \Sigma (0, 1, 3, 6, 7)$$

$$F_2 (A, B, C) = \Sigma (1, 5, 6)$$

Solution: In order to design these functions by using ROM, we need to decide the type of Decoder. ROM is a combination of Decoder and OR gates. Two functions are present and hence we need two OR gates. Only three inputs A, B,

C is present and hence decoder needed is 3 : 8.

The representation of ROM is 8 × 2 which means '2' OR gates and 8 (2^3) means 8 address lines and 3 input variables.

The designing is given as follows :

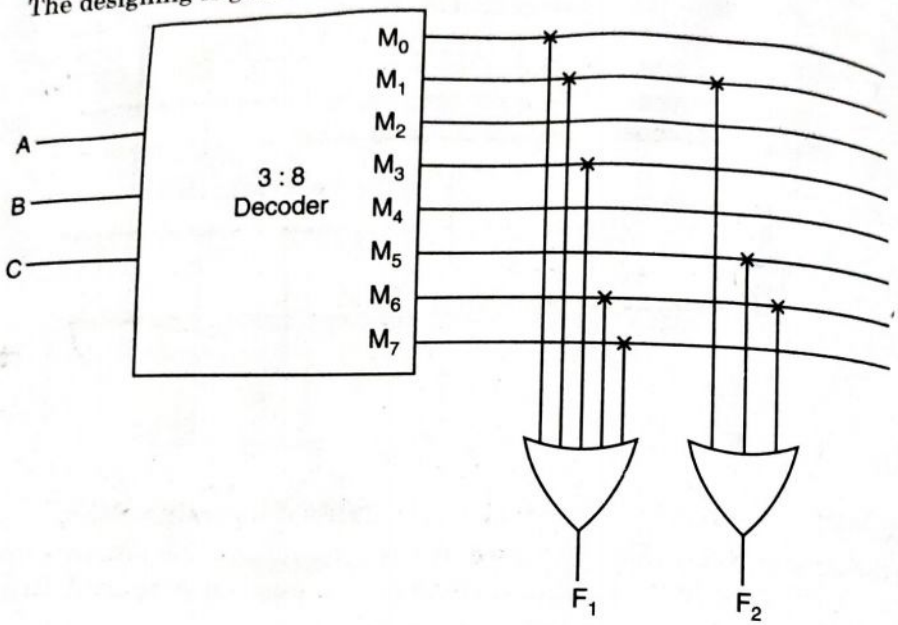


Fig. 7.15.

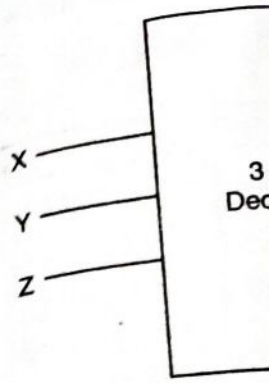
EXAMPLE 7.2. Design the square of three bits numbers using ROM.

Solution: The three bit number has range from 0 to 7 and hence maximum square value is 49 which is represented by 6 bits and hence the output consists of 5–OR gate (as '*e*' donot require any 1 at output). The decoder used is 3 to 8. The truth table representation is given as follows :

Decimal No.	*Input variable*			*Decimal No.*	*Output variables*					
	X	*Y*	*Z*	*No.*	*a*	*b*	*c*	*d*	*e*	*f*
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1
2	0	1	0	4	0	0	0	1	0	0
3	0	1	1	9	0	0	1	0	0	1
4	1	0	0	16	0	1	0	0	0	0
5	1	0	1	25	0	1	1	0	0	1
6	1	1	0	36	1	0	0	1	0	0
7	1	1	1	49	1	1	0	0	0	1

Output for a, b, c, d, e, f, is given as

$$a = \Sigma m\ (6, 7)$$
$$b = \Sigma m\ (4, 5, 7)$$
$$c = \Sigma m\ (3, 5)$$
$$d = \Sigma m\ (2, 6)$$



EXAMPLE 7.3. De

Solution: For Half

The decoder used i

truth table for Hal

The designi

where S denot

EXAMPLE 7.

e = No minterm

$f = \Sigma m\ (1, 3, 5, 7)$

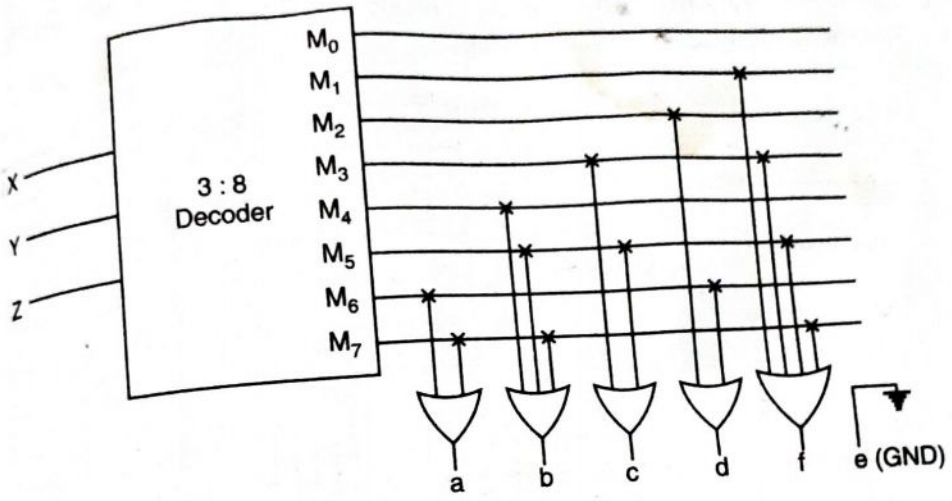


Fig. 7.16.

EXAMPLE 7.3. Design the circuit for Half Adder using ROM.

Solution: For Half Adder, two outputs are present, two OR gates are required. The decoder used in 2 : 4 which states that it has two inputs *i.e.* A and B. The truth table for Half-Adder is given as :

Input Varible		*Outputs*	
A	***B***	***S***	***C***
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The designing of the given circuit is by using 4 × 2 ROM

$S = \Sigma m\ (1, 2)$

$C = \Sigma m\ (3)$

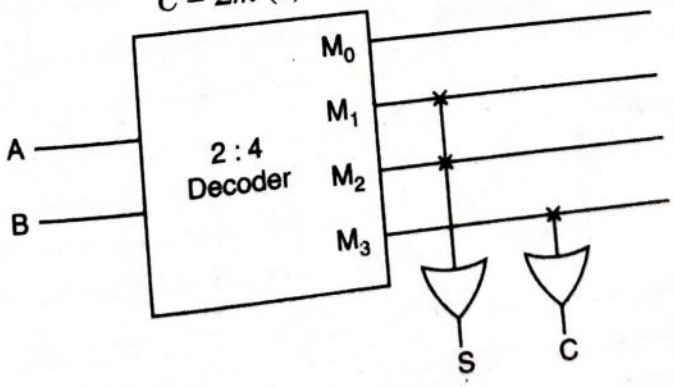


Fig. 7.17.

where S denote sum output and C denotes carry output.

EXAMPLE 7.4. Design a BCD to 2421 Code using ROM.

Solution: It is represented by 10×4 ROM where 10 is the address lines and 4 is the output lines. It can be designed by 4 : 16 decoder and 4–OR gates. The truth table is given as follows :

Decimal No.	BCD Code				2421 Code			
	A	B	C	D	P	Q	R	S
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	1
5	0	1	0	1	1	0	1	0
6	0	1	1	0	1	1	0	1
7	0	1	1	1	1	1	0	0
8	1	0	0	0	1	1	1	0
9	1	0	0	1	1	1	1	1

The output expression for P, Q, R, S is given as

$$P = \Sigma m\ (5, 6, 7, 8, 9)$$
$$Q = \Sigma m\ (4, 6, 7, 8, 9)$$
$$R = \Sigma m\ (2, 3, 5, 8, 9)$$
$$S = \Sigma m\ (1, 3, 5, 7, 9)$$

The designing is given as follows :

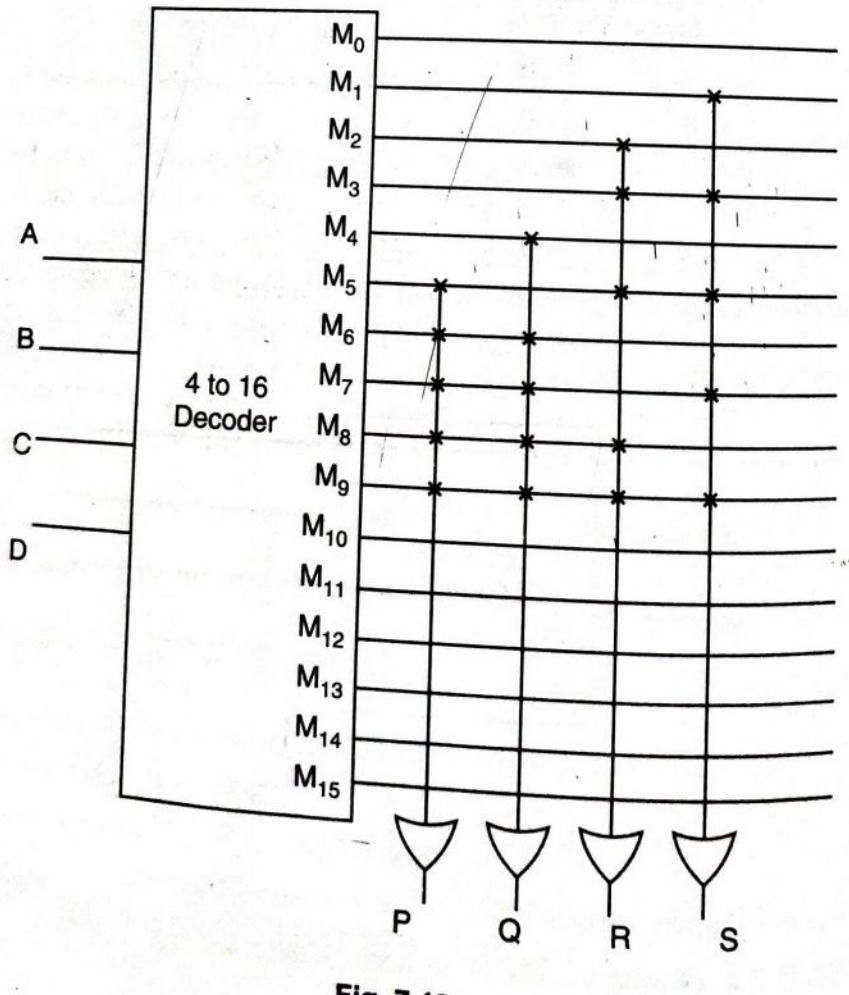


Fig. 7.18.