UNIT - 2

Programmable Logic Devices ~~& Threshold Logic~~

Introduction

- We have seen various "standard ICs" (such as adders, comparators, encoders, decoders, code converters, multiplexers, demultiplexers, ALU e.t.c) for performing basic digital operations and other functions.
- These standard ICs are also called as "fixed function IC" because each one of them performs a fixed digital operation.
- Problems of using standard (or) fixed-function ICs in logic design are that they require large amount of board space, large power, lack of security, a great deal of time and cost in inserting, soldering, and testing.
- i.e the standard ICs required additional cost, space, power requirements, etc to modify the required
- Additional cost, space, power requirements, e.t.c required to modify the design & to introduce new features.
- To overcome the disadvantages these ICs, two new approaches are there. Those are 1) ASICs (Application specific integrated ckts)
  2) PLD's (Programmable logic devices)

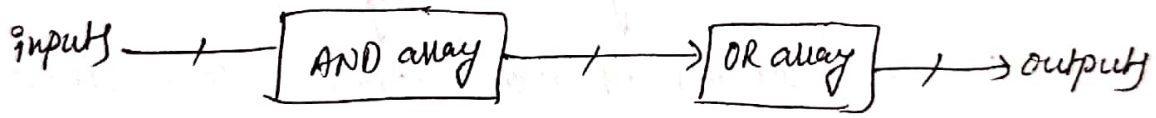
Programmable Logic Devices:

- A Programmable Logic Device is an IC consisting of large no. of gates (Ranging from few hundreds to millions (LSI)) which can be configured by the designer (user) to design different functions circuits according to his own specifications (functions).

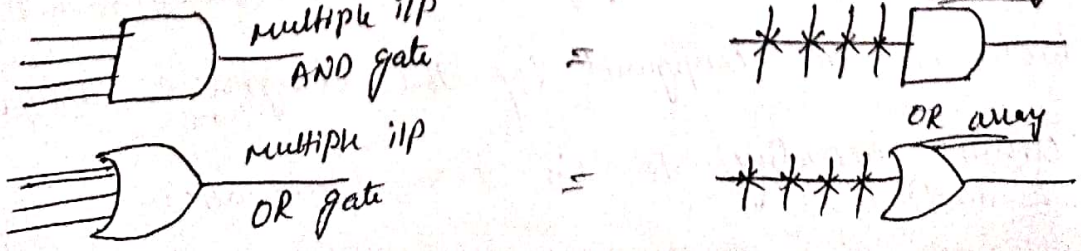
- PLDs are different from standard ICs.
- Advantages of using PLDs are less board space, faster, lower power requirements, less cost, higher reliability (fewer ICs and circuit connections means easier the troubleshooting)
- The IC is said to be programmable because the specific function of the IC for a given application is determined by the selective breaking of some of the interconnections while leaving others intact.

  This process is called 'programming' because it produces the desired circuit pattern interconnecting the gates, flipflops, registers, and so on.

- PLDs are typically built with an array of AND gates (AND-array) and an array of OR gates (OR-array)



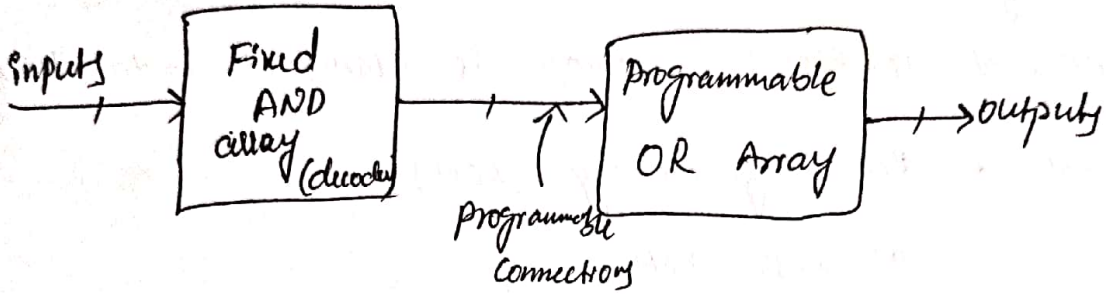
- All the internal logic gates connections of PLDs are configured by a programming process.
- One of the simplest programming technologies is to use fuses. In the original state of the device, all the fuses are intact. By blowing fuses of unwanted connections required functionality is obtained.
- conventional symbols



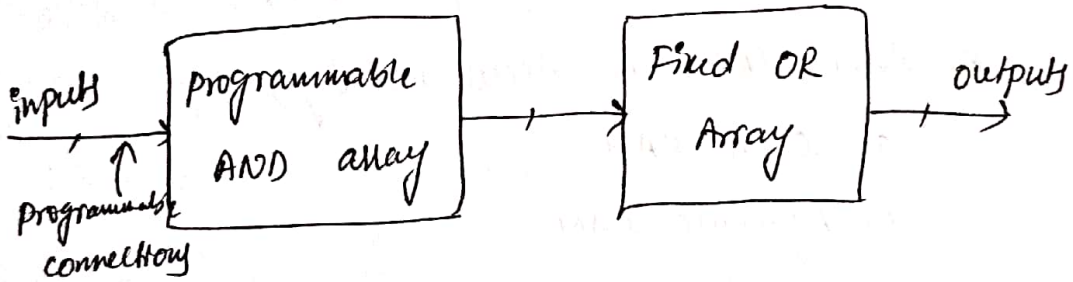
Symbols used in PLDs

- There are 3 fundamental types of PLDs:
  - PROM (Programmable Read only ROM)
  - PAL (Programmable Array Logic)
  - PLA (Programmable Logic Array)

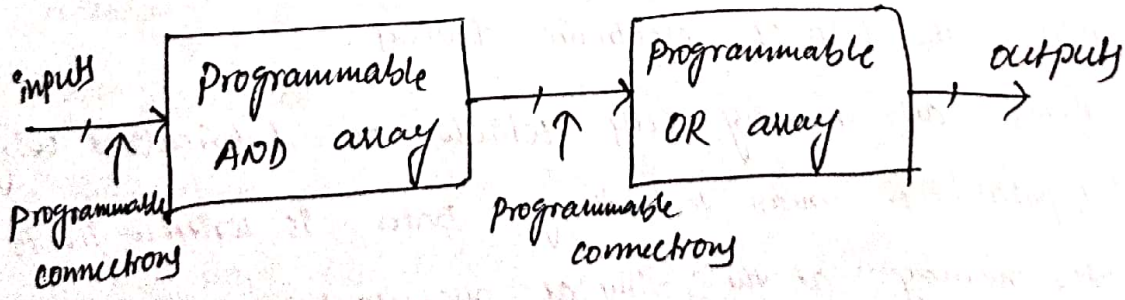
a) Programmable Read only ROM (PROM)



b) Programmable Array Logic (PAL)



c) Programmable Logic Array (PLA)



- A 4th type of PLD, which is discussed later, is the Complex Programmable Logic Device (CPLD), Eg: FPGA (Field Programmable Gate Array).

Memory :

A memory is a storage device which is used to store as well as to make it available for the further processing.

Types of Memory : Memory is classified in to two types :) Read only memory (ROM)

a) mask ROM
b) PROM (Programmable ROM)
c) EPROM (erasable PROM)
d) EEPROM (Electrically EPROM)

2) RAM (Random Access memory)

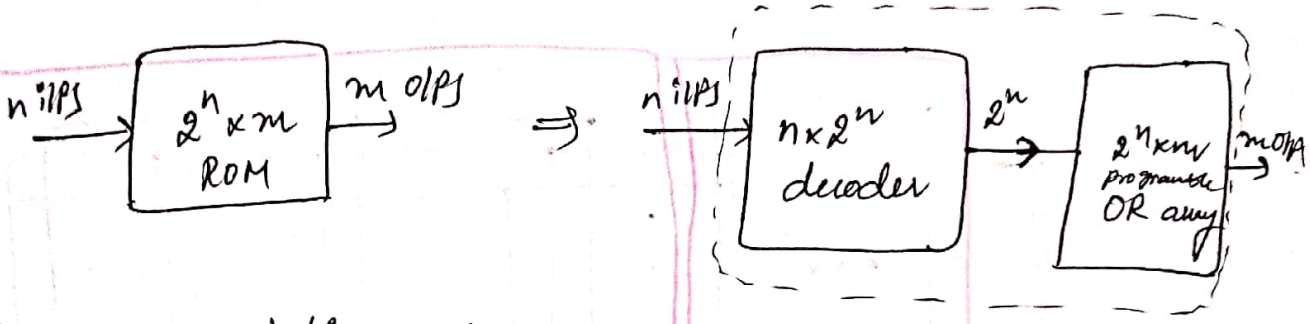
a) Static RAM
b) Dynamic RAM

ROM :

- ROM is used to store binary information permanently with the help of electronic devices.
- ROMs are memory cells which are fabricated using bipolar & mos technology. Data is written in to the memory at the time of manufacturer according to the users requirement.
- ROMs are widely used to implement complex combinational circuits directly from their truth tables.
- PROM is one of the types of ROM.

→ PROM : (Programmable Read-only Memory)

- A PROM is a device that includes both the decoder and the OR gates with in a single IC package.
- The block diagram of PROM is given below.



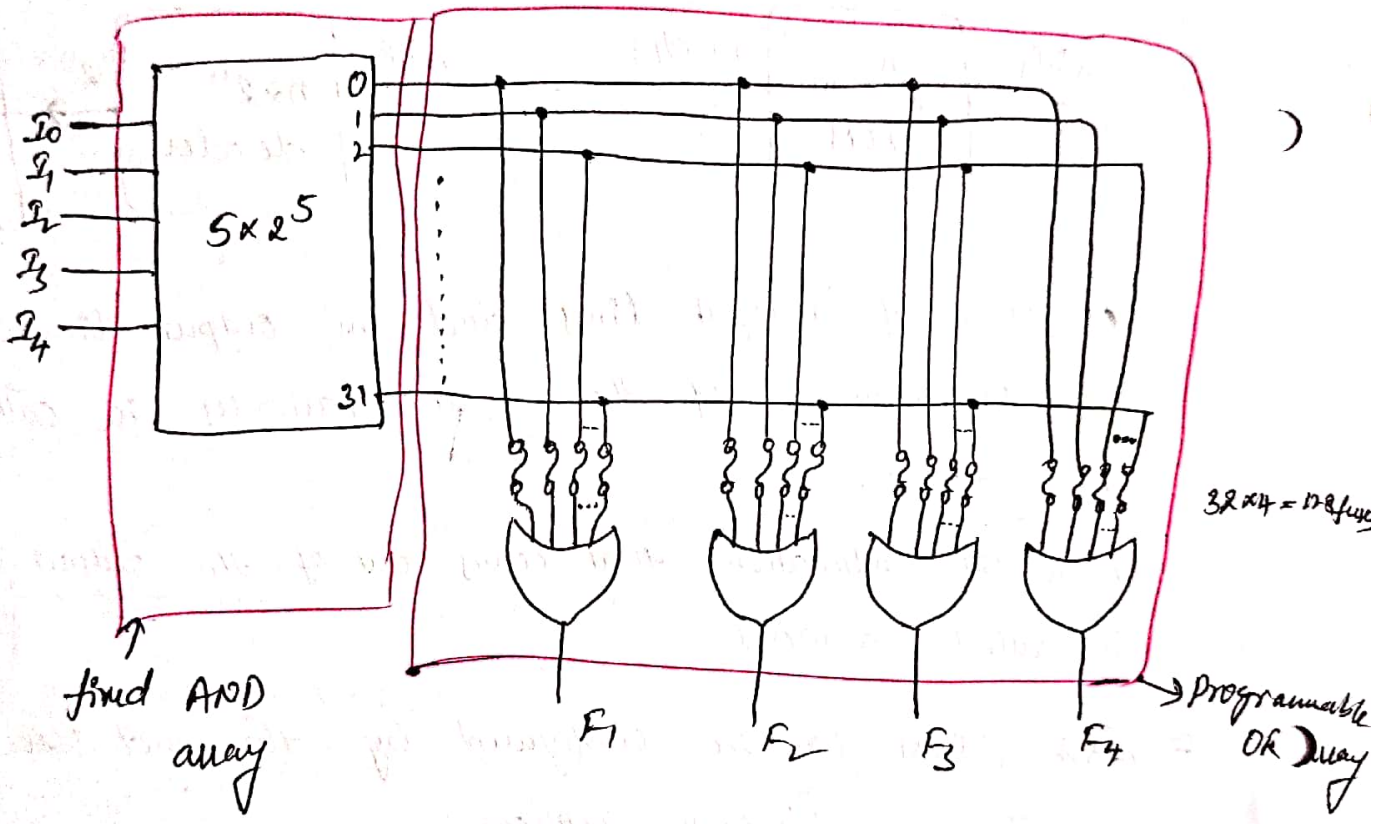
It consist of 'n' input lines and 'm' output lines.
Each bit combination of the input variables is called an address.
Each bit combination that comes out of the output lines is called a word.

- PROM can be configured by the end user according to his own application

Ex: consider 32 × 4 PROM. It consists of 32 words of 4 bits each. This means that there are 4 o/p lines and particular word from 32 words presently available on the output lines is determined from the 5 input lines ($2^5 = 32$).
There are only 5 inputs in a 32×4 PROM and with 5 variables, we can specify 32 addresses & minterms.
Fo each address input, there is a unique selected word.

Thus, if the input address is 00000, word number '0' is selected and applied to the output lines. If the input address is 00001, word number '1' is selected, applied to the output lines. If the input address is 11111, word number 31 is selected and applied to the output lines.

The internal logic construction of a 32×4 PROM is shown below.



- The PROM is a two level implementation in sum of minterms form (SOP). i.e PROM is capable of realizing any switching function as it generates all minterms.

- NOTE: No advantage with minimization of a switching function using a PROM.

- We can use the PROM for the realization of switching functions.

Combinational logic implementation using PROM:

- By observing the logic diagram of the PROM, we can realize that each output provides the sum of all the minterms of n input variables.
- For an n-input, m-output combinational circuit, we need $2^n \times m$ PROM.

Ex: 1) Implement the following Boolean functions using PROM.

$$F_1(A,B) = \Sigma m(1,2,3)$$
$$F_2(A,B) = \Sigma m(0,2)$$

Ans: If the given functions are not expressed in canonical SOP form convert them into CSOP form.

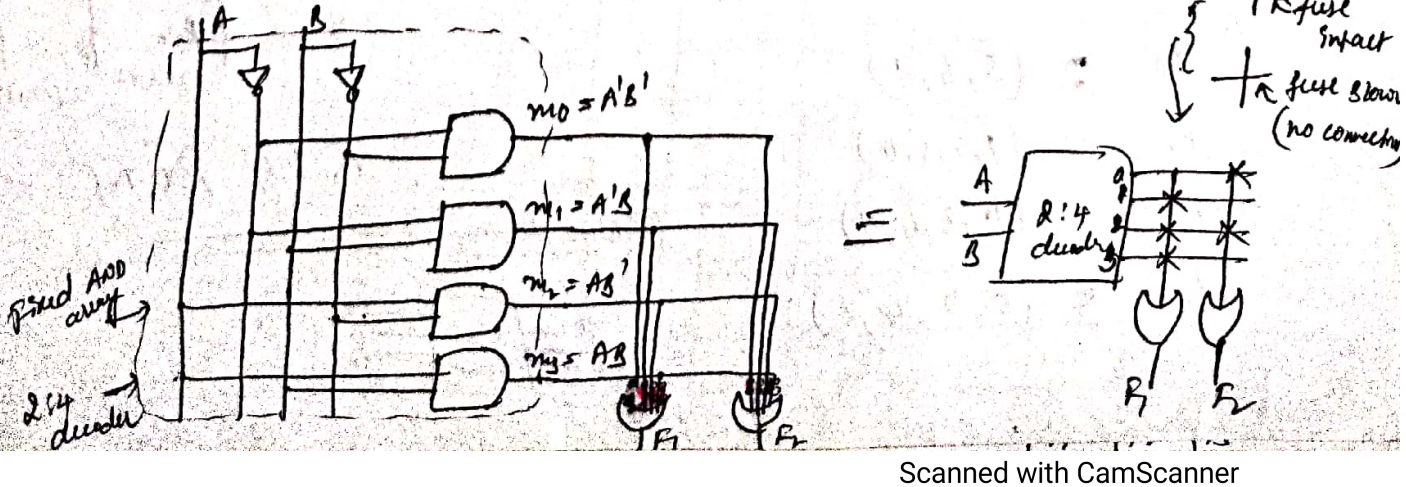
Here the given functions are already in CSOP form.

So directly take, $F_1(A,B) = \Sigma m(1,2,3)$

$$F_2(A,B) = \Sigma m(0,2)$$

The given functions indicate that the circuit has 2 i/ps (A,B) and 2 outputs (F_1, F_2).

∴ These boolean functions can be implemented by 4×2 PROM (or) 4×2 ROM.

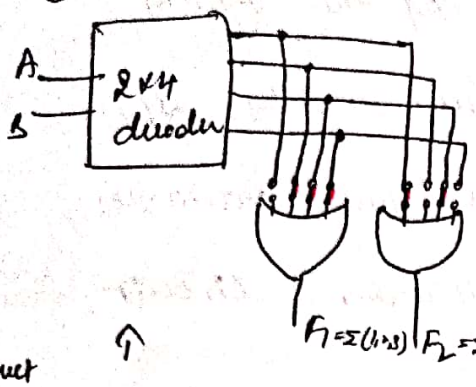


Other way Representation of above ckt is given below.

① Truth table

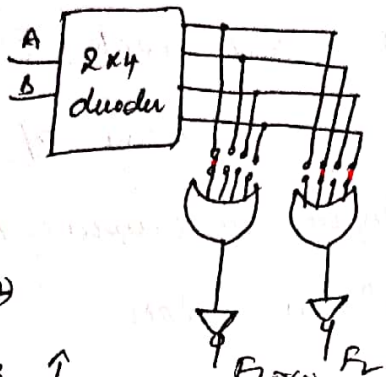
A	B	F_1	F_2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0

②



Q: Construct 4:2 ROM with AND-OR gates

③



Q: Construct 4:2 ROM with AND-OR-Inverter

Ex: 2) Design a combinational ckt using ROM that circuit accepts 3 bit binary number and generate its equivalent Excess-3 code.

A: Let us derive the truth table for the given ckt condition.

①

	i/p 3-bit Binary code			o/p Excess-3 code			
	A	B	C	P_0	P_1	P_2	P_3
0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	0
2	0	1	0	0	1	0	1
3	0	1	1	0	1	1	0
4	1	0	0	0	1	1	1
5	1	0	1	1	0	0	0
6	1	1	0	1	0	0	1
7	1	1	1	1	0	1	0

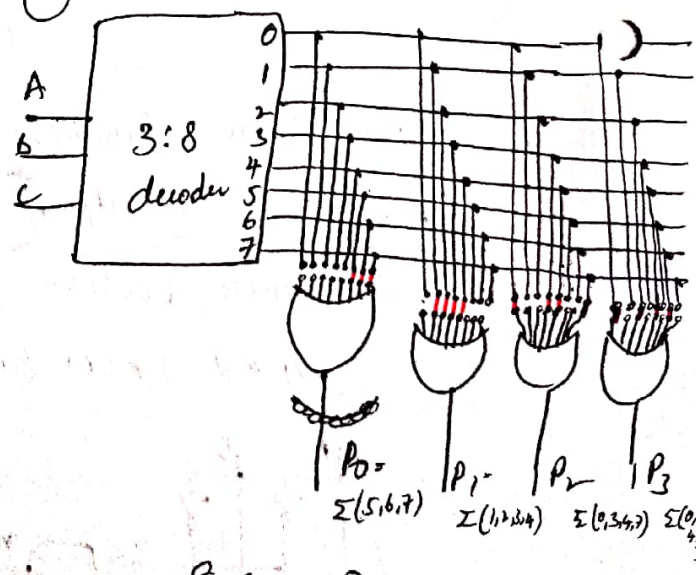
② $P_0 = \Sigma(5,6,7)$

$P_1 = \Sigma(1,2,3,4)$

$P_2 = \Sigma(0,3,4,7)$

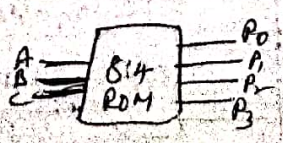
$P_3 = \Sigma(0,2,4,6)$

③ $P_0 = \Sigma(5,6,7)$



8:4 ROM

④ Block diagram



Ex:3) Realize the following three switching functions using a single ROM & PROM. Mention the size of ROM

$$f_1(A,B,C) = AB + \bar{B}C$$

$$f_2(A,B,C) = (A+\bar{B}+C)(\bar{A}+B)$$

$$f_3(A,B,C) = A + BC$$

Ans

Step 1: Since the given functions are not expressed in canonical SOP form convert them into CSOP form

$$f_1(A,B,C) = AB(C+C') + B'C(A+A')$$

$$= ABC' + ABC + A'B'C + A'B'C$$

$$= \Sigma(1,5,6,7)$$

Similarly $f_2 = \Pi(2,4,5) = \Sigma(0,1,3,6,7) = m_0 + m_1 + m_3 + m_6 + m_7$

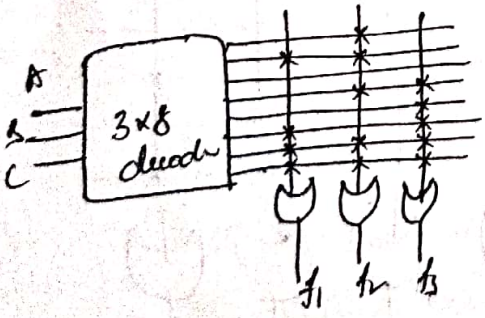
Similarly $f_2 = (A+B'+C)(A'+B+C\cdot C') = \Pi(2,4,5) = \Sigma(0,1,3,6,7)$

$$f_3 = A(B+B')(C+C') + (A+A')BC = \Sigma(3,4,5,6,7)$$

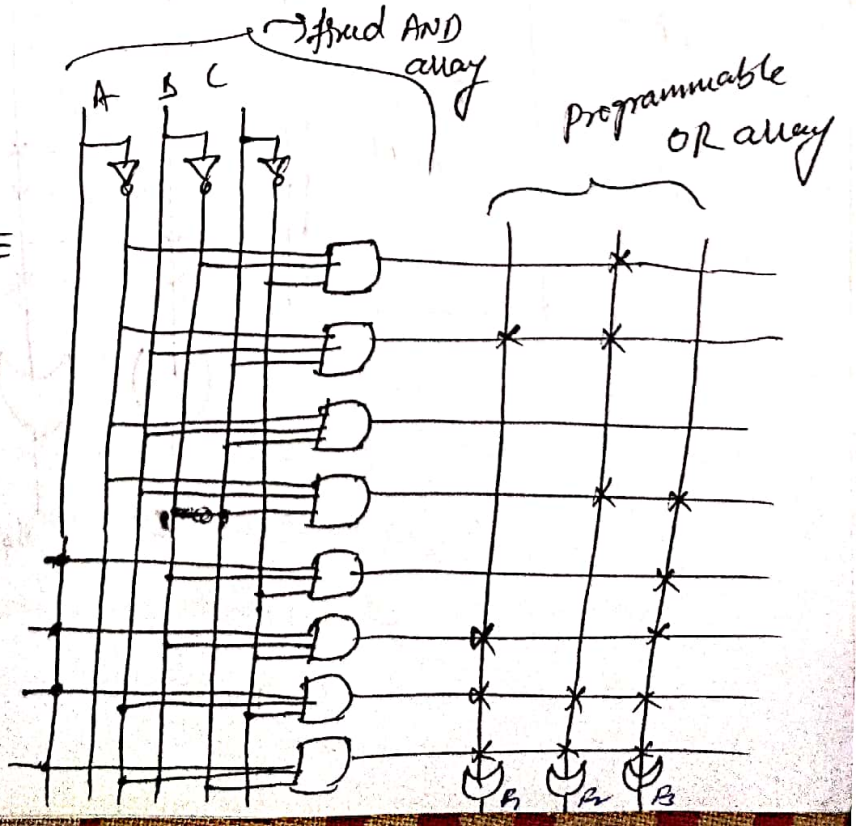
Step 2: PROM (or) ROM Realization of f_1, f_2, f_3

Size of ROM/PROM: 3 I/Ps, $2^3 = 8$ Product terms, 3 O/Ps.

$\therefore$ 8×3 ROM



≡



Ex: 4) Design a combinational ckt using a ROM. The ckt accepts a 3-bit number and generates an output binary number equal to the square of the input number.

A:

	i/ps			o/ps						Decimal
	A	B	C	B_5	B_4	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	1	1
2	0	1	0	0	0	0	1	0	0	4
3	0	1	1	0	0	1	0	0	1	9
4	1	0	0	0	1	0	0	0	0	16
5	1	0	1	0	1	1	0	0	1	25
6	1	1	0	1	0	0	1	0	0	36
7	1	1	1	1	1	0	0	0	1	49

$B_5 = \Sigma(6,7)$

$B_4 = \Sigma(4,5,7)$

$B_3 = \Sigma(3,5)$

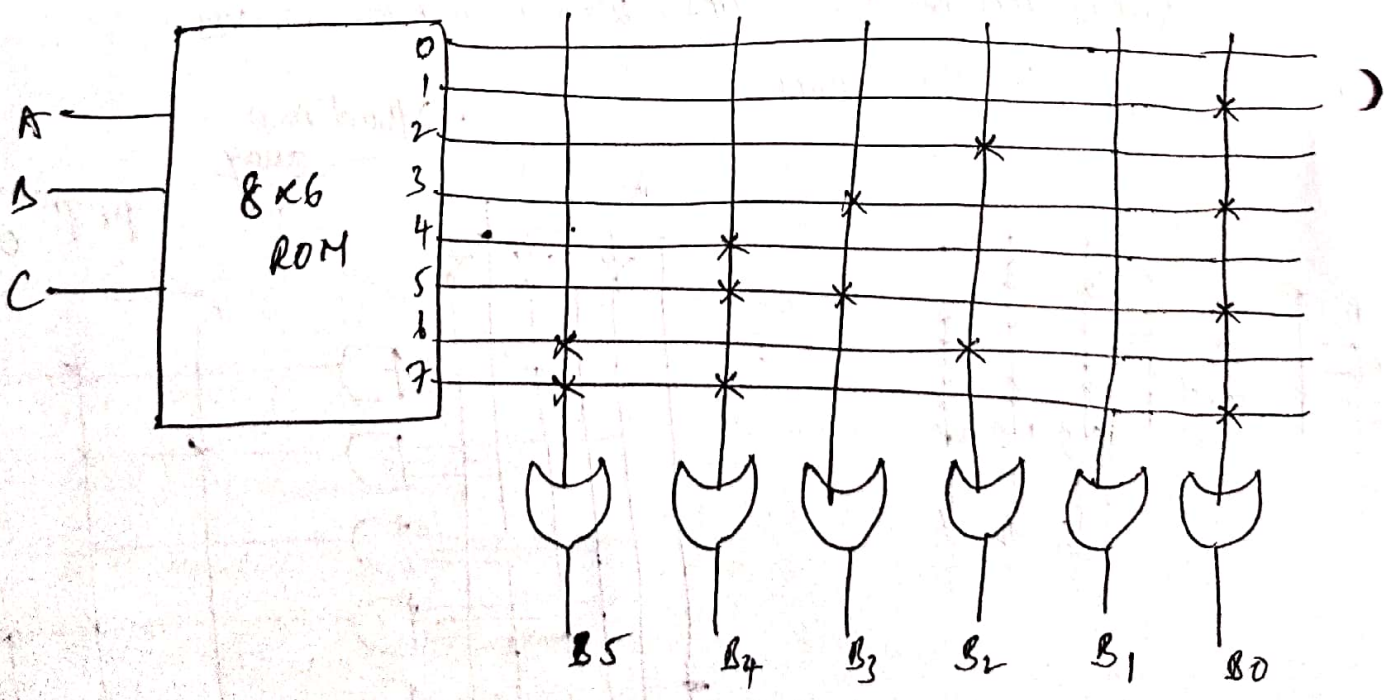
$B_2 = \Sigma(2,6)$

$B_1 = 0$

$B_0 = \Sigma(1,3,5,7)$

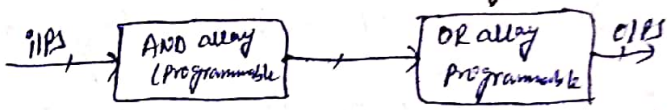
Size of ROM: 3 i/ps → $2^3 = 8$ product terms, 6 o/ps

8 x 6 ROM



→ PLA : (Programmable Logic Array)

- A PLA is similar to PROM, know But it does not provide full decoding of the variables and does not generates all the minterms as in the PROM (i.e it eliminates/ burns the unwanted fuses (minterms)).
- The PLA replaces decoder by group of AND gates, each of which can be programmed to generate a product term of the input variables.
- In PLA, both AND and OR gates have fuses at the inputs, ∴ in PLA both AND and OR gates are Programmable



- PLA → MPLA (MASK PLA)
  → FPLA (Field PLA)
- A block diagram of the (F)PLA is shown in below figure.

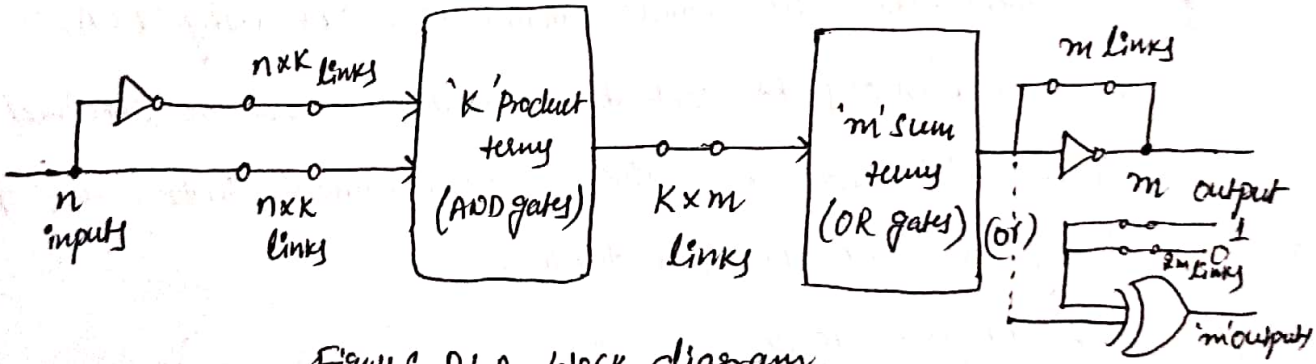


Figure: PLA block diagram

It consists of 'n' inputs, 'm' outputs, 'K' product terms, and m sum terms.

The product terms constitute a group of 'K AND gates' and the sum terms constitute a group of 'm OR gates.

- The size of PLA's are specified by the no. of i/Ps, to the no. of product terms and the no. of o/Ps

is equal to the no. of sum terms.

- Total no. of programmable links, if the o/p of each OR gate goes to an inverter gate is $= (2n \times k) + (k \times m) + m$
- Total no. of programmable links, if the o/p of each OR gate goes to an EX OR gate is $= (2n \times k) + (k \times m) + 2m$

Eg: TTL IC 82S100 is a typical PLA with 16 inputs, 48 product terms and 8 outputs.

- **Implementation of combination Logic Ckt using PLA:**
  - Like ROM, PLA can be a) Mask Programble (MPLA) 2) Field Programmable (FPLA)
  - The use of a PLA must be considered for combinational circuits that have a large number of inputs and outputs.
  - It is superior to a ROM for circuits that have a large no. of don't-care conditions.
  - To implement the combinational Logic Ckt using PLA, Logic expressions for content information must be obtained first, then minimized, and finally programmed into the PLA using a "PLA program table".
  - **PLA Program Table**

    PLA program table specifies product terms and sum terms of information that will be stored in PLA.

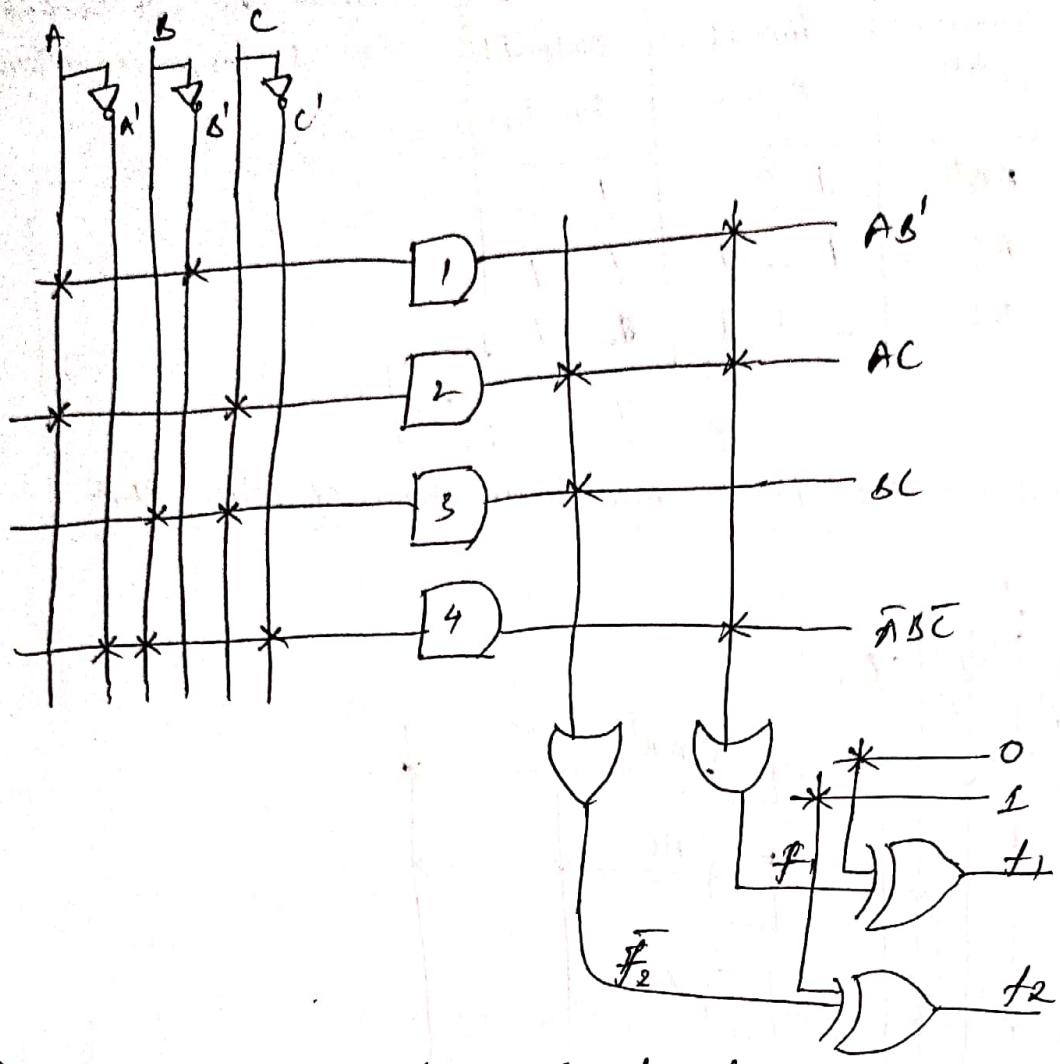
    Eg: Program table for $f_1 = A\bar{B} + AC + \bar{A}B\bar{C}$

    $f_2 = AC + BC$

S.NO.	Product term	I/Ps	O/Ps	
		A B C	f_1	f_2
1	$A\bar{B}$	1 0 –	1	–
2	AC	1 – 1	1	1
3	BC	– 1 1	–	1
4	$\bar{A}B\bar{C}$	0 1 0	1	–

$T \rightarrow f_1$ is in normal form
$C \rightarrow f_2$ is in complemented form

Implementation



Eg: A combinational circuit is defined by the functions:

$F_1(A,B,C) = \Sigma(4,5,7)$

$F_2(A,B,C) = \Sigma(3,5,7)$

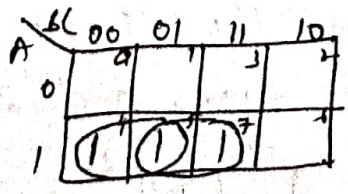
Implement the circuit with a PLA having 3 inputs 3 product terms and two O/Ps.

Ans: minimize the functions using k-maps.

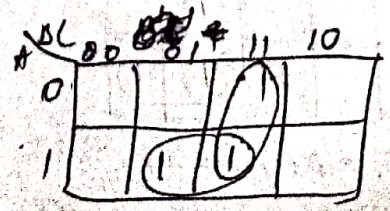
(a) Truth table

A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

(b) k-map simplification



$F_1 = AB' + AC$



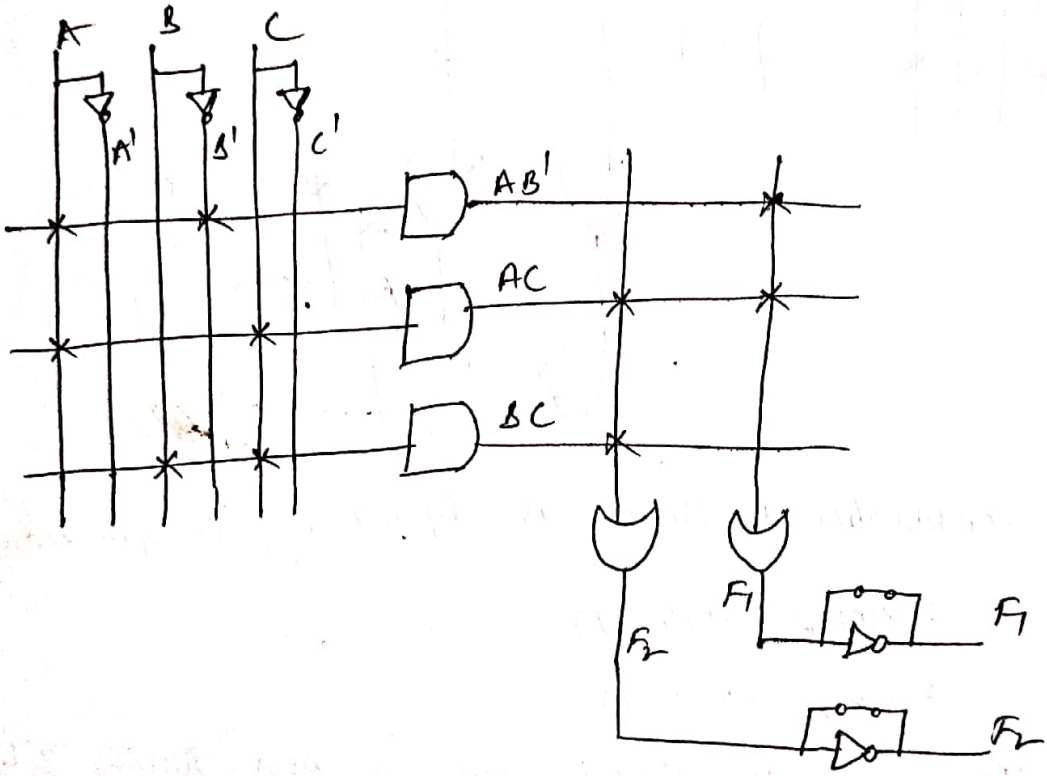
$F_2 = AC + BC$

PLA Program table :

In above expressions we have 1 common term (AC). So we have 3 (AB', AC, BC) distinct product terms

S.No.	Product term	inputs A B C	outputs F_1 F_2
1	AB'	1 0 –	1 –
2	AC	1 – 1	1 1
3	BC	– 1 1	– 1
		T T	T/C

Combinational ckt for practical implementation of with PLA



Eg: A combinational ckt is defined by the functions:

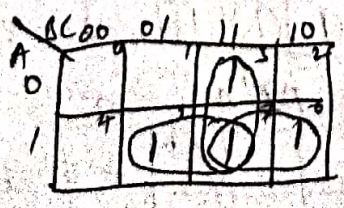
$$F_1(A,B,C) = \Sigma(3,5,6,7)$$

$$F_2(A,B,C) = \Sigma(0,2,4,7)$$

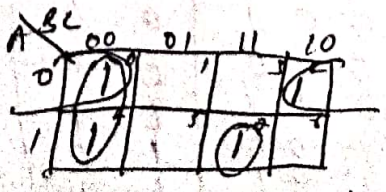
Implement the circuit with a PLA having 3 I/Ps, 4 product terms, and 2 O/Ps.

Ans:

Method - I



$F_1 = AC + AB + BC$

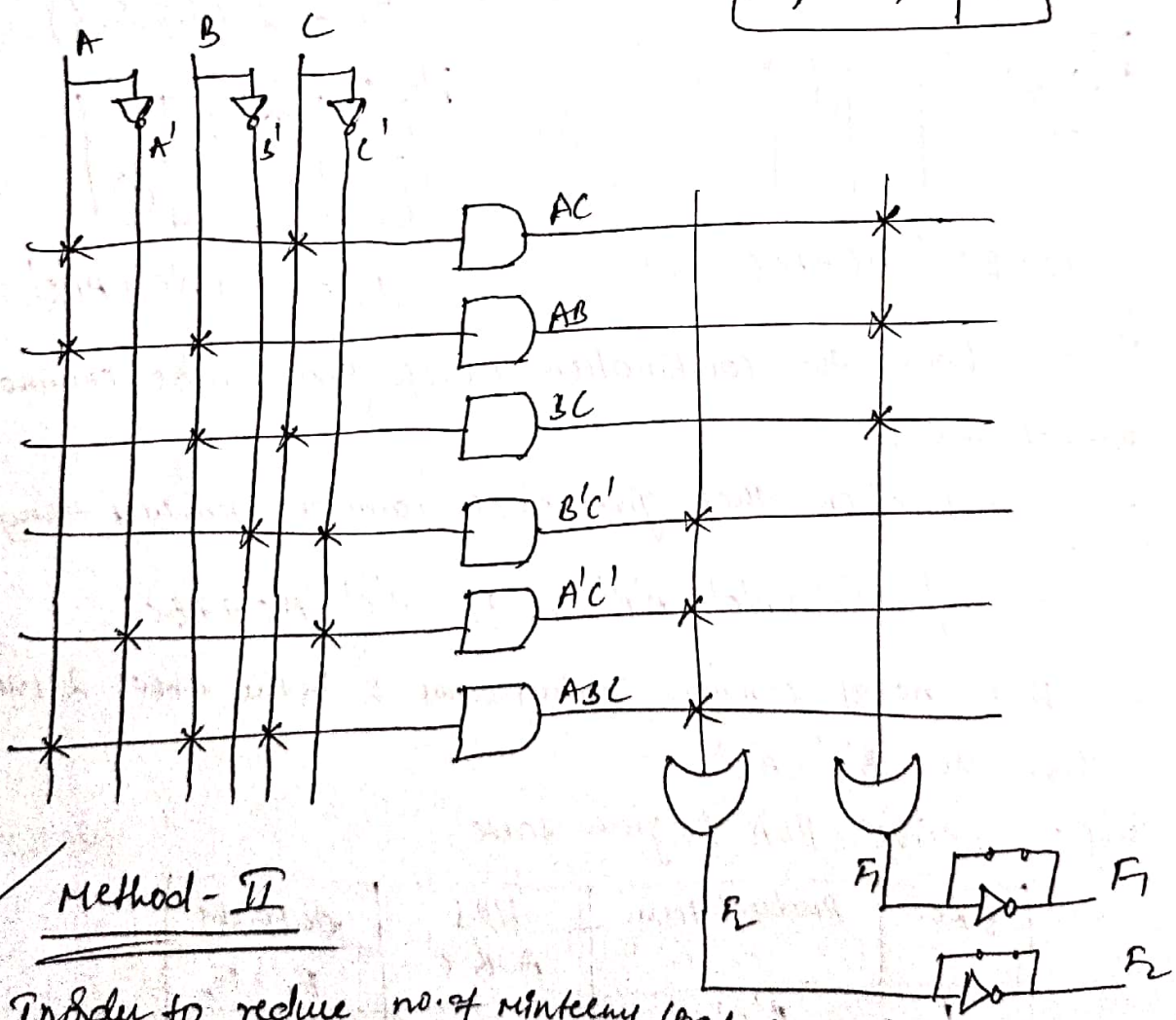


$F_2 = B'C' + A'C' + ABC$

Above two functions are having no common product term. So we have total 6 distinct product terms.

PLA program table :

S.NO.	Product term	inputs A B C	outputs F_1	F_2
1	AC	1 – 1	1	–
2	AB	1 1 –	1	–
3	BC	– 1 1	1	–
4	B'C'	– 0 0	–	1
5	A'C'	0 – 0	–	1
6	ABC	1 1 1	–	1
			T	T	T/C



Method - II

In order to reduce no. of minterms (product terms), we go for this method. Due to this method we are reducing the hardware in practical PLA Implementation.

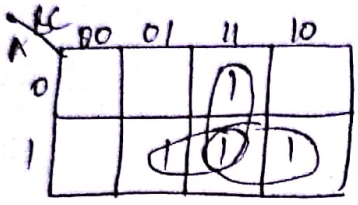
$F_1(A,B,C) = \Sigma(3,5,6,7)$

$F_2(A,B,C) = \Sigma(0,2,4,7)$

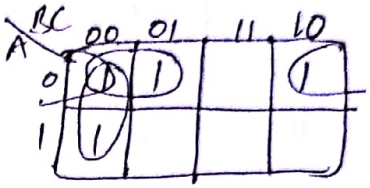
A:

Step 1: Minimize the functions in such a way that there exist more no. of common product terms.

$F_1 = \Sigma(3,5,6,7) \longrightarrow F_1' = \Sigma(0,1,2,4)$

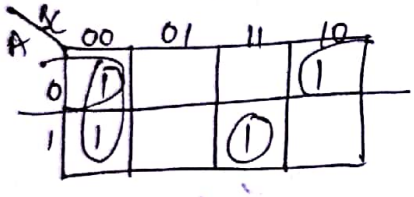


$F_1 = AC + BC + AB$

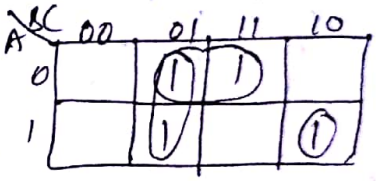


$F_1' = B'C' + A'C' + A'B'$

$F_2 = \Sigma(0,2,4,7) \longrightarrow F_2' = \Sigma(1,3,5,6)$



$F_2 = B'C' + A'C' + ABC$



$F_2' = B'C + A'C + ABC'$

Step 2: Choose the combination which gives more common product terms.

The combination that gives more common product terms

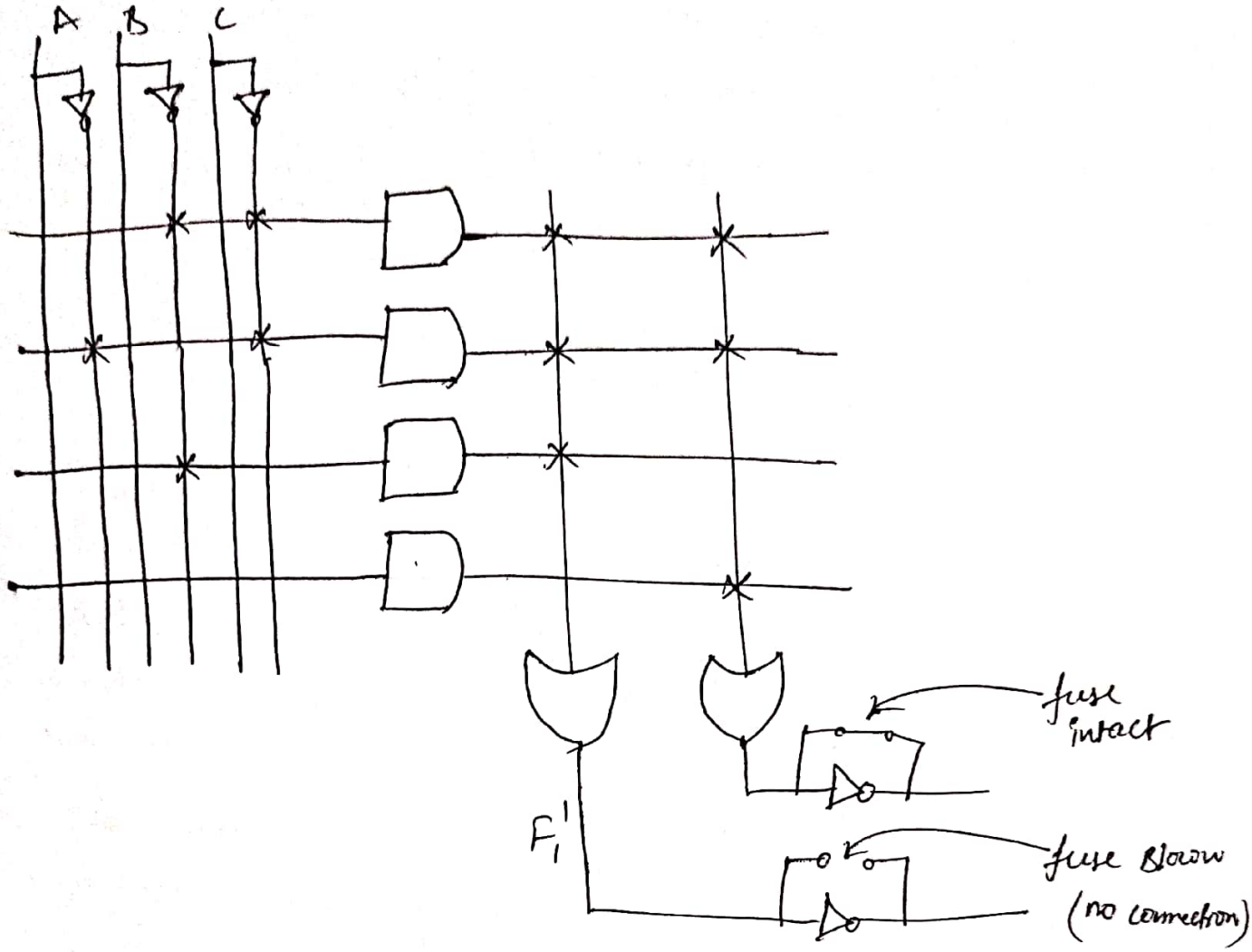
$F_1' = B'C' + A'C' + A'B'$, $F_2 = B'C' + A'C' + ABC$

Here no. of common product terms 2 in the above 2 equations those are $B'C'$, $A'C'$

Step 3: Derive PLA Program table

S.NO.	Product term	I/Ps A B C	Outputs F_1	Outputs F_2
1	$B'C'$	$-\ 0\ 0$	1	1
2	$A'C'$	$0\ -\ 0$	1	1
3	$A'B'$	$0\ 0\ -$	1	-
4	ABC	$1\ 1\ 1$	-	1

Step 4: Implementation of given functions using PLA



Ex: Implement the following boolean functions using PLA

$$f_1(w,x,y,z) = \Sigma(0,1,3,5,9,13)$$

$$f_2(w,x,y,z) = \Sigma(0,2,4,5,7,9,11,15)$$

→ PAL (Programmable Array Logic):

- The programmable array logic (PAL) is a programmable logic device with a fixed OR array and a programmable AND array.
- Because only the AND gates are programmable, the PAL is easier to program, but is not as flexible as the PLA.
- Fixed OR array means no. of product terms in each section is fixed and if the number of terms in the function is too large, it may be necessary to use two (or) more sections to implement one Boolean function.
- The commercial PAL devices has more gates. A typical PAL integrated circuit may have 8 i/ps, 8 o/ps, and 8 sections, each consisting of an 8 wide AND-OR array.

Implementation of combinational logic circuit using PAL:

Let us see the implementation of a combinational circuit using PAL with the help of examples.

Eg:1) Design a combinal circuit using PAL for the following functions.

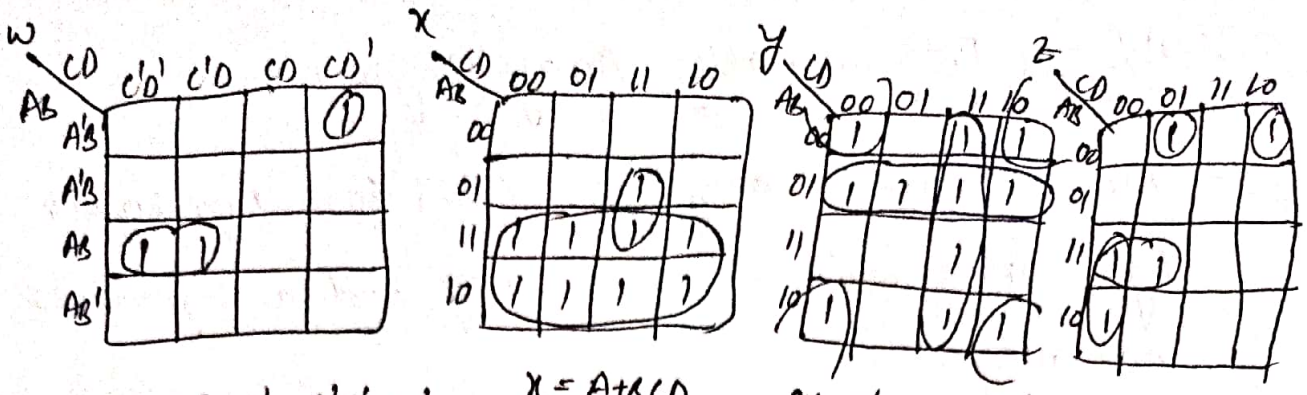
$$w(A,B,C,D) = \Sigma(2,12,13)$$

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

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

$$z(A,B,C,D) = \Sigma(1,2,8,12,13)$$

Ans: Simplifying the four functions to a minimum number of terms results in the 4 using K-map.

① 

$W = ABC' + A'B'CD'$

$X = A + BCD$

$Y = A'B + CD + B'D'$

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

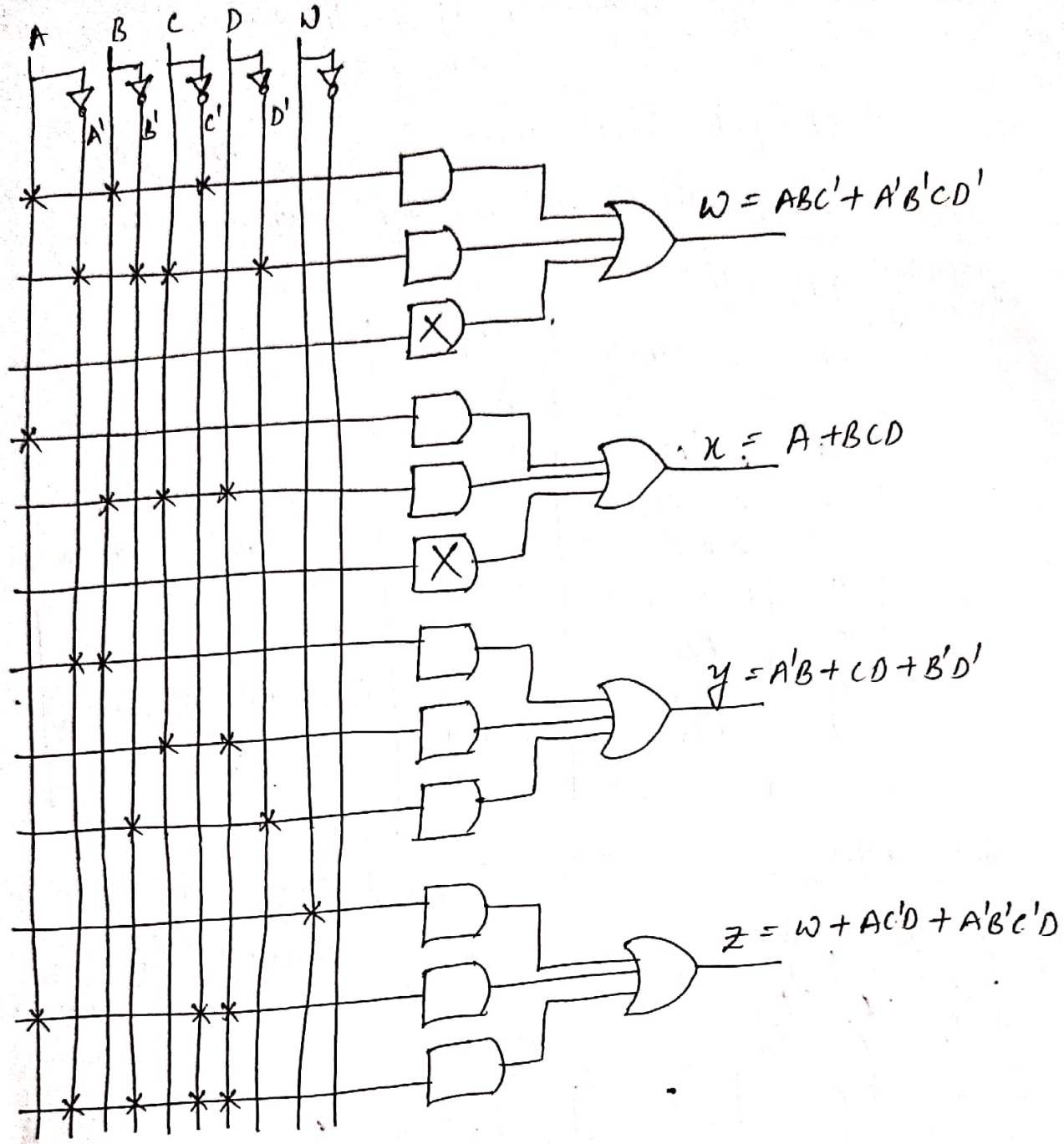
$= W + AC'D' + A'B'C'D$

② Programming table:

The PAL programming table is similar to PLA programming table except that only the inputs of the AND gates need to be programmed.

The above ckts maximum consist of 3 product terms. So we are taking 3 terms each of 4 sections in the table.

product term	i/ps A B C D	W	outputs
1	1 1 0 –	–	$W = ABC' + A'B'CD'$
2	0 0 1 0	–	
3	– – – –	–	
4	1 – – –	–	$X = A + BCD$
5	– 1 1 1	–	
6	– – – –	–	
7	0 1 – –	–	$Y = A'B + CD + B'D'$
8	– – 1 1	–	
9	– 0 – 0	–	
10	– – – –	1	$Z = W + AC'D' + A'B'C'D$
11	1 – 0 0	–	
12	0 0 0 1	–	

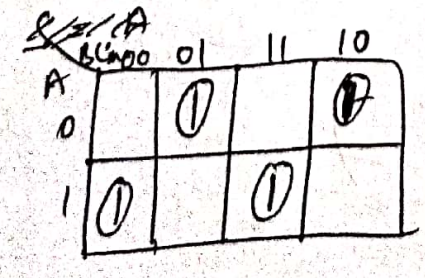


Eg: 7) Implementation of a Full Adder using a PAL

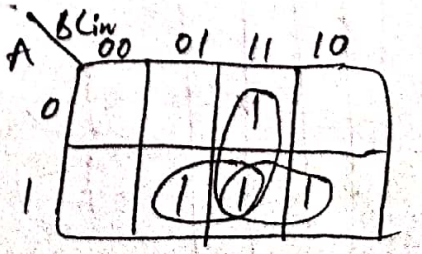
Sum = S = Σ (1,2,4,7),

Carry out = C_{out} = Σ (3,5,6,7)

A:



$S = A'B'C_{in} + A'BC_{in}' + AB'C_{in}' + ABC_{in}$



$C_{out} = AC_{in} + BC_{in} + AB$

From above logic equations, no. of i/ps = 3
no. of o/ps = 2

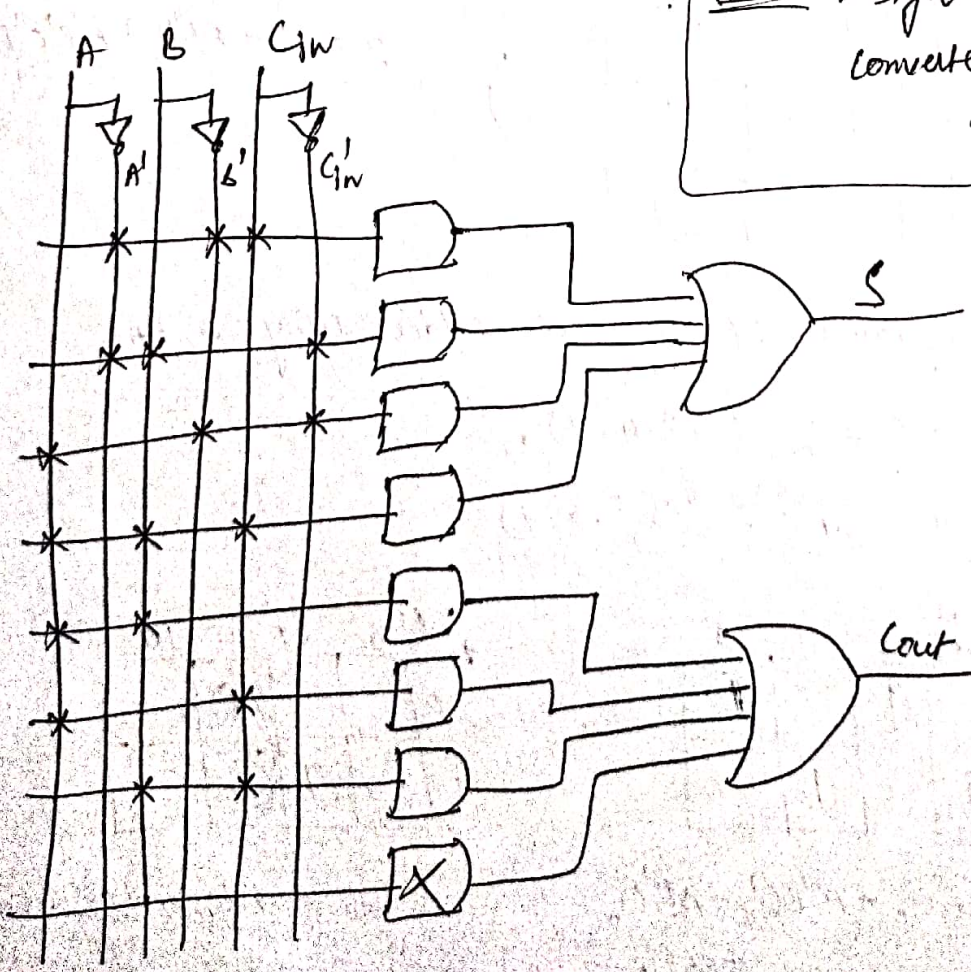
i.e TWO OR gates each driven by four AND gates.

∴ Sum expression is realized using 4 product terms.

Programming table:

S.NO.	Product term	i/ps A B C_{in}	o/ps
1	$A'B'C_{in}$	0 0 1	$S = A'B'C_{in} + A'BC'_{in} + AB'C'_{in} + ABC_{in}$
2	$A'BC'_{in}$	0 1 0	
3	$AB'C'_{in}$	1 0 0	
4	ABC_{in}	1 1 1	
5	AB	1 1 –	$C_{out} = AB + AC_{in} + BC_{in}$
6	AC_{in}	1 – 1	
7	BC_{in}	– 1 1	
8	—	– – –	

Implementation



H.W: Design Excess-3 to BCD converter using
a) ROM b) PAL c) PLA

Table 5.4 Comparison of programmable logic devices

PROM	PLA	PAL
1. AND array is fixed and OR array is programmable.	1. Both AND and OR arrays are programmable.	1. OR array is fixed and AND array is programmable.
2. Cheaper and simple to use.	2. Costliest and more complex than PALs and PROMs.	2. Cheaper and simpler.
3. All minterms are decoded.	3. AND array can be programmed to get desired minterms.	3. AND array can be programmed to get desired minterms.
4. Only Boolean functions in standard SOP form can be implemented using PROM.	4. Any Boolean function in SOP form can be implemented using PLA.	4. Any Boolean function in SOP form can be implemented using PAL.