Presentation Topic

Analog and Digital Electronics

Yogesh K Sharma
yogesh.sharma@viit.ac.in
Department of Computer Engineering

BRACT'S, Vishwakarma Institute of Information Technology, Pune-48

(An Autonomous Institute affiliated to Savitribai Phule Pune University)
(NBA and NAAC accredited, ISO 9001:2015 certified)

Introduction
To
Programmable Logic Devices(PLD's)

Unit - 4

Courtesy: R. P. Jain & J. S. Katre (Modern Digital Electronics)

Course Objectives

1. To learn and understand basic digital circuit design techniques.
2. To study the implementation of digital circuits using combinational logic.
3. To study the implementation of digital circuits using sequential logic.
4. To illustrate the concept of PLD's.
5. To study the implementation of digital circuits using VHDL.
6. To understand basics of Logic Families and IOT circuit boards in development of mini digital circuits.

Course Outcome

1. Simplify Boolean algebraic expressions for designing digital circuits using K- Maps. (Analyzing)
2. Apply digital concepts in designing combinational circuits. (Applying)
3. Apply digital concepts in designing sequential circuits. (Applying)
4. Implement digital circuits using PLA and PAL. (Applying)
5. Design digital circuits using VHDL. (Applying)
6. Design and implementation of Mini digital circuit applications. (Applying)

.

Introduction to PLD's

Introduction to PLD's:

- Introduction to PLD's: - ROM, PAL, PLA, Applications of PLAs to implement combinational and sequential logic circuits.

Programmable Logic Device

- 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.
- We have studied different IC's such as multiplexers, demultiplexers, adders, code converters which are called as fixed function ICs.
- The advantages of designing logic circuits using the fixed function ICs are low development cost & easy testing, & disadvantages is requirement of large board space, large power requirements, no security, additional costs needed for the modification of existing circuit.
- The ICs that are used for designing a specific application is called the application specific integrated circuits (ASICs). They are too complex to design.
- The third approach to the problem of digital circuit design is to use the programmable logic devices (PLDs). It has the advantages of both the approaches discussed earlier.
- PLD are special type of ICs which can be programmed by the users as per their requirements.
- Thus it is possible to implement a combinational or sequential circuit using the PLD ICs.

Programmable Logic Device

Advantages of PLDs:

- Reduced space requirement.
- Reduction in power requirement.
- Increased speed of switching.
- Better security, Higher density, low production cost as compared to ASICS.
- Low development cost, short design cycle i.e. time required to implement the design is short.
- More flexibility to the designer.
- Reprogramming is possible to be done within few seconds.
- Modifications can be carried out within a short span of time.
- An important application of PLDs is that we can prepare prototype ASIC design, & reduces the time required to design ASIC

Programmable Logic Device

Types of PLDs:

- Programmable Read only memories (PROMs).
- Programmable Logic Array (PLAs).
- Programmable Array Logic (PALs).
- Simple Programmable Logic Devices (SPLDs).
- Complex Programmable Logic Devices (CPLDs).
- Field Programmable Gate Arrays (FPGAs).

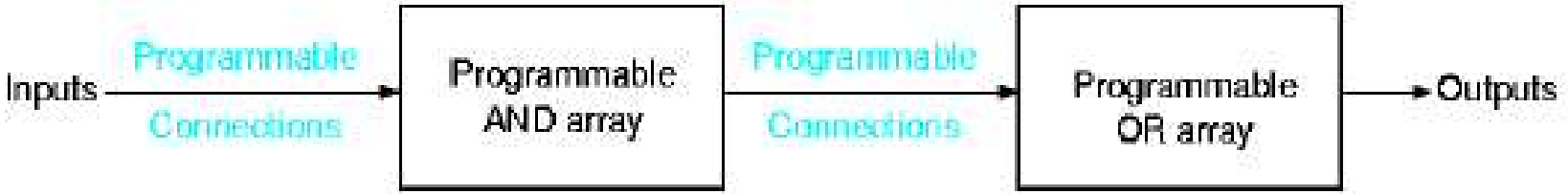
Note:

- Stores permanent binary information (nonvolatile). Can be read only (cannot be altered). Information is specified by designer and physically inserted (embedded) into the PLD.
- Programmable connections are formed by fuses, masks, or antifuses depending on the technology.

PROGRAMMABLE LOGIC DEVICES

(a) Programmable read-only memory (PROM)

(b) Programmable array logic (PAL) device

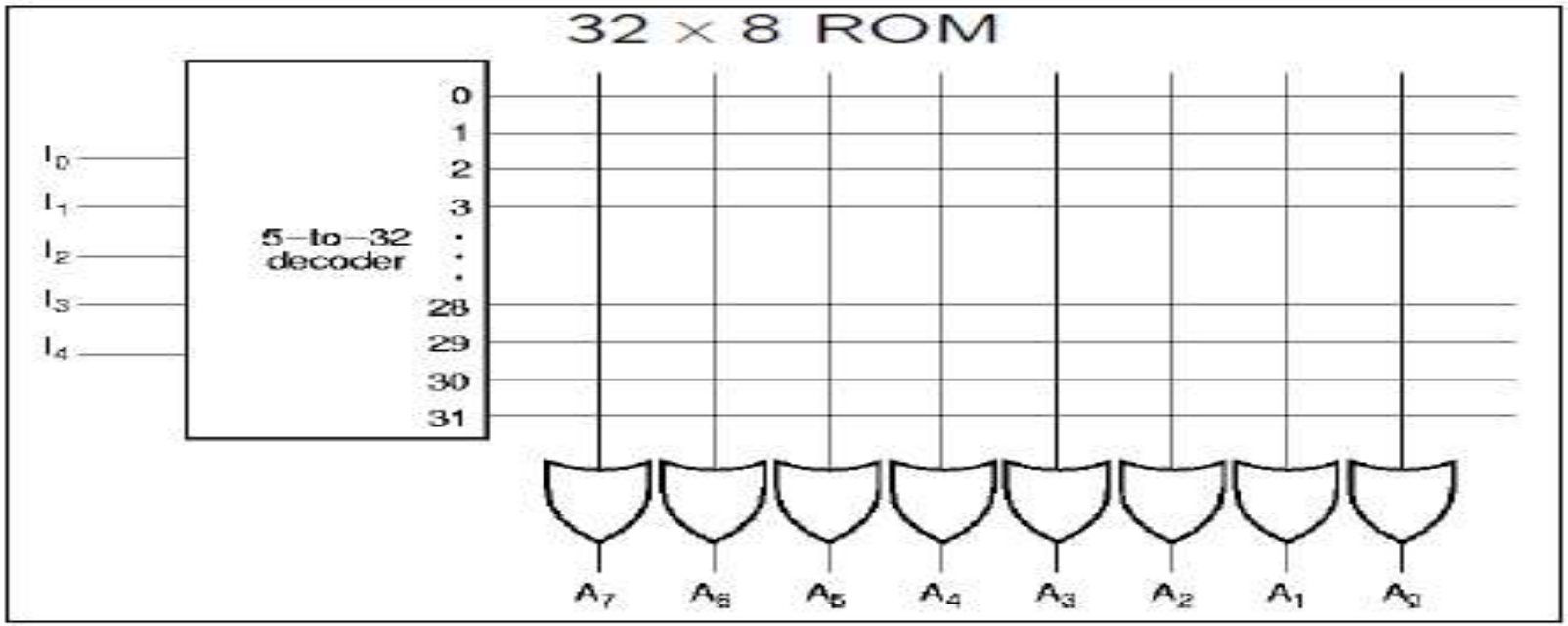


(c) Programmable logic array (PLA) device

ROM as PLD

Read-Only Memory

k inputs (address) $\Rightarrow$ $2^k \times n$ ROM $\Rightarrow$ n outputs (data)



ROM as PLD

- k *x* 2^k decoder to decode input address.
- n OR gates with 2^k input each.
- Decoder output is connected to all n OR gates through fuses.
- ROM -> 2^k *x* n programmable connections.
- EX: 4 x 2 ROM.
- Truth table -> address and content of ROM.
- Programming -> stores truth table in ROM.
- 0 = Open connection = Fuse blown.
- 1 = Closed connection = Fuse intact.

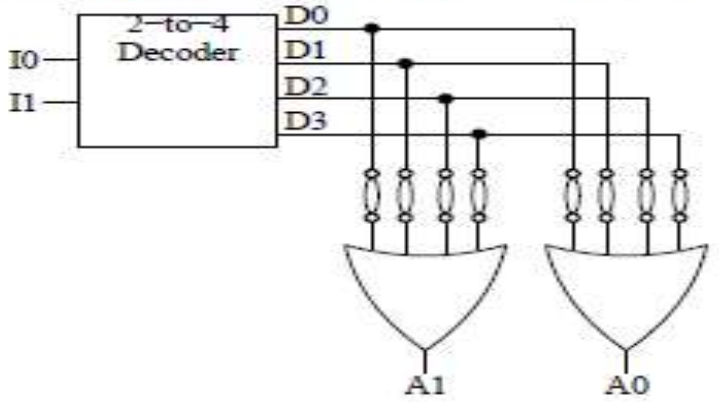
ROM as PLD

Example of 4 × 2 ROM

Truth table

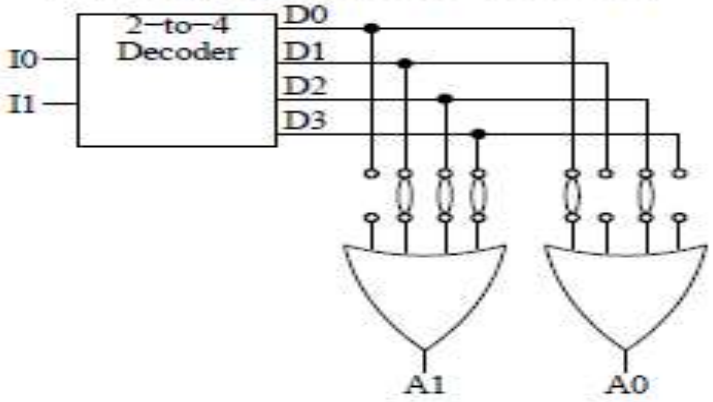
Address		Content	
I_1	I_0	A_1	A_0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0

NonProgrammed ROM

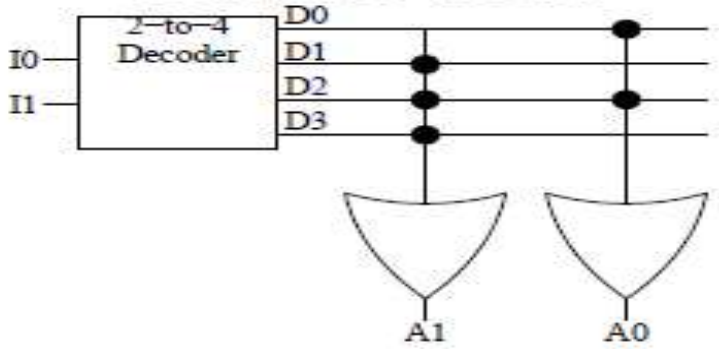


⇓

Programmed ROM



Compact ROM



Function Synthesis with ROM

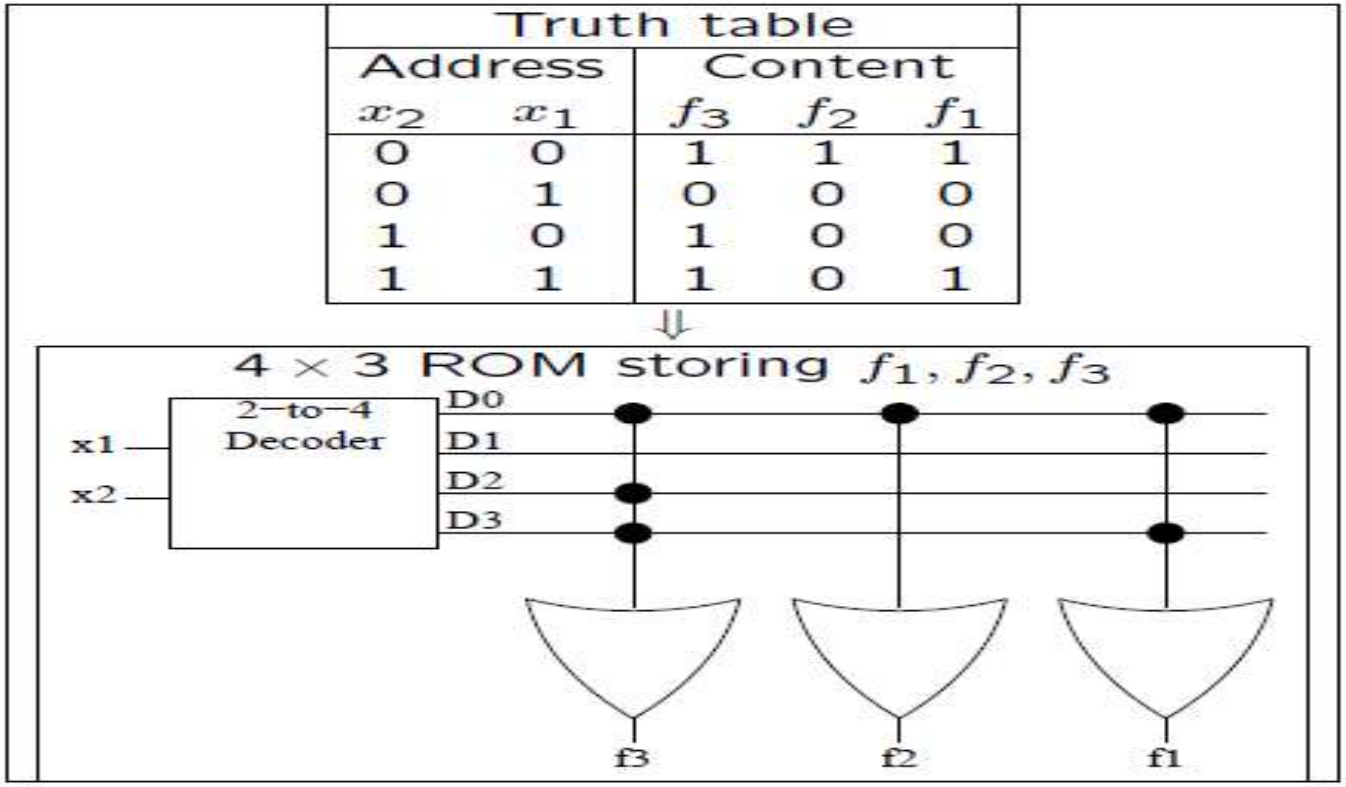
Any set of functions $f_1(x_k, \ldots, x_1)$, $\ldots$, $f_n(x_k, \ldots, x_1)$ can be realized with a $2^k \times n$ ROM

Example: Implement $f_1(x_2, x_1) = \sum m(0, 3)$, $f_2(x_2, x_1) = \overline{x_2 + x_1}$, and $f_3(x_2, x_1) = \prod M(1)$ with a 4×3 ROM

Truth table

Address		Content		
x_2	x_1	f_3	f_2	f_1
0	0	1	1	1
0	1	0	0	0
1	0	1	0	0
1	1	1	0	1

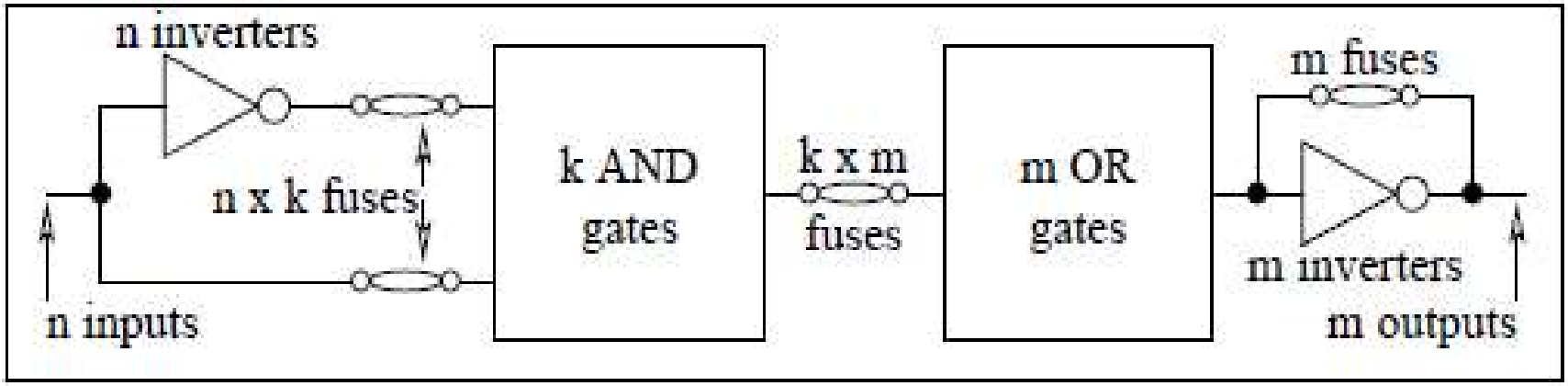
$\Downarrow$



Programmable Logic Array

- A PLD generally consists of programmable array of logic gates. Interconnections are made with the array inputs.
- The outputs are connected to the device pins through inverting or noninverting buffers and flip flops.
- The logic gates used can be two level AND-OR, NAND-NAND or NOR-NOR configuration. Sometimes AND-OR-EXOR configuration is also used.
- There are two types of PLDs namely the PLA i.e. programmable logic arrays and PAL i.e. the programmable array logic. Due to the used of AND matrix followed by the OR matrix, we can use them for the implementation of logic functions in the SOP form.
- A PLA consists of two level of AND-OR circuit on the same chip.
- The AND matrix can be used to implement the product terms in the SOP form and the OR matrix is used for implementing the sum of product term.

Programmable Logic Array

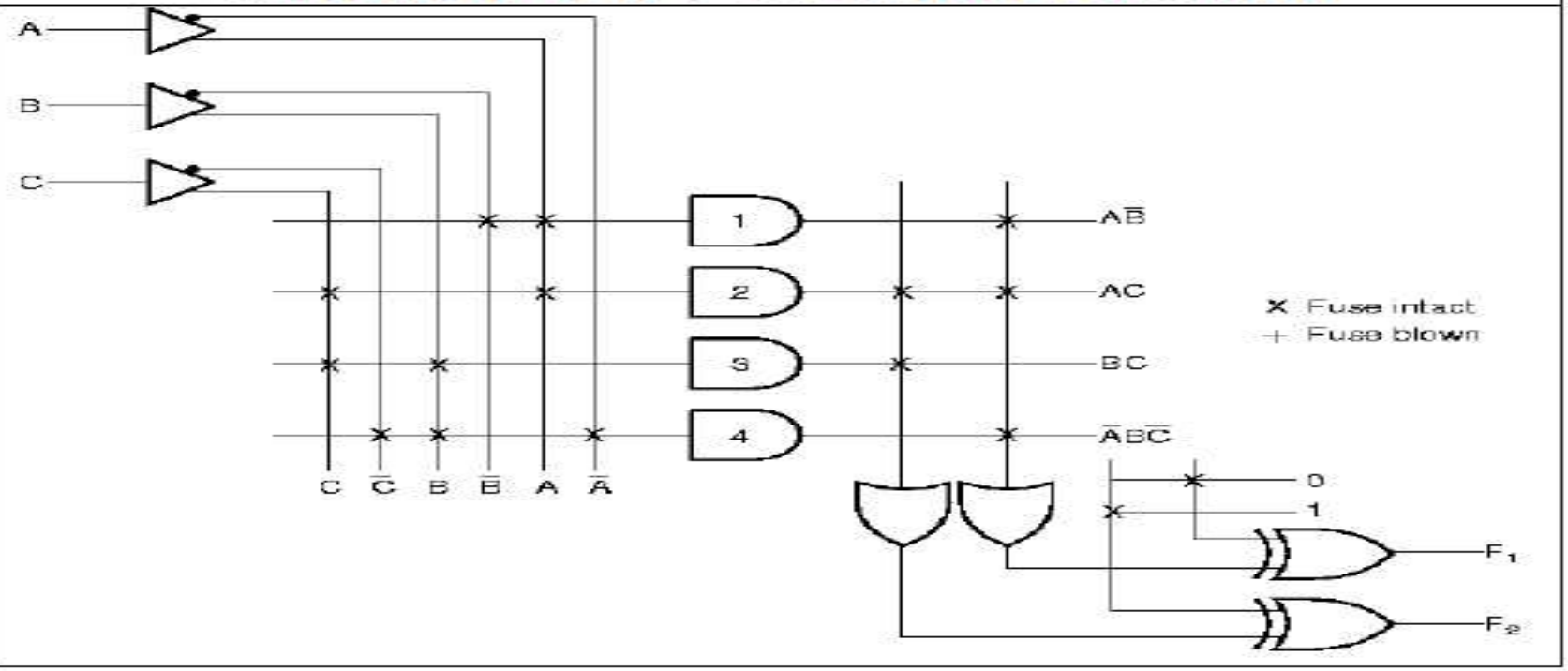


- Behave like a ROM but has different structure:
- Uses ANDs array instead of decoder to produce product terms of inputs
- Has programmable connections before ANDs, between ANDs and ORs, after ORs. That is $2nk + km + m$ fuses
- More flexible than ROM but more difficult to program.
- Logic expressions for content information to be stored in PLA must be obtained first, then minimized, and finally programmed into the PLA using a PLA program table.
- PLA program table specifies product terms and sum terms of information that will be stored in PLA.

Programming a PLA

PLA Program Table						
		Inputs			Outputs	
Term	Term #	A	B	C	F_1	F_2
$A\bar{B}$	1	1	0	_	1	_
AC	2	1	_	1	1	1
BC	3	_	1	1	_	1
$\bar{A}B\bar{C}$	4	0	1	0	1	_
					T	C

Corresponding PLA Implementation



Function Synthesis with PLA

Any set of functions $f_1(x_1, \ldots, x_n)$, $\ldots$, $f_m(x_1, \ldots, x_n)$ can be realized with a PLA

Example Implement $f_1(a, b, c) = \sum m(3, 5, 6, 7)$ and $f_2(a, b, c) = \sum m(0, 2, 4)$ with a PLA

First Simplify $f_1, \bar{f}_1, f_2, \bar{f}_2$, that is

$$f_1(a, b, c) = ab + ac + bc \qquad f_1, f_2 \rightarrow 5 \text{ terms}$$
$$\bar{f}_1(a, b, c) = \bar{a}\bar{b} + \bar{a}\bar{c} + \bar{b}\bar{c} \qquad f_1, \bar{f}_2 \rightarrow 4 \text{ terms}$$
$$f_2(a, b, c) = \bar{a}\bar{c} + \bar{b}\bar{c} \qquad \bar{f}_1, f_2 \rightarrow 3 \text{ terms}$$
$$\bar{f}_2(a, b, c) = ab + c \qquad \bar{f}_1, \bar{f}_2 \rightarrow 5 \text{ terms}$$

Second Select combination of functions that has less terms, that is

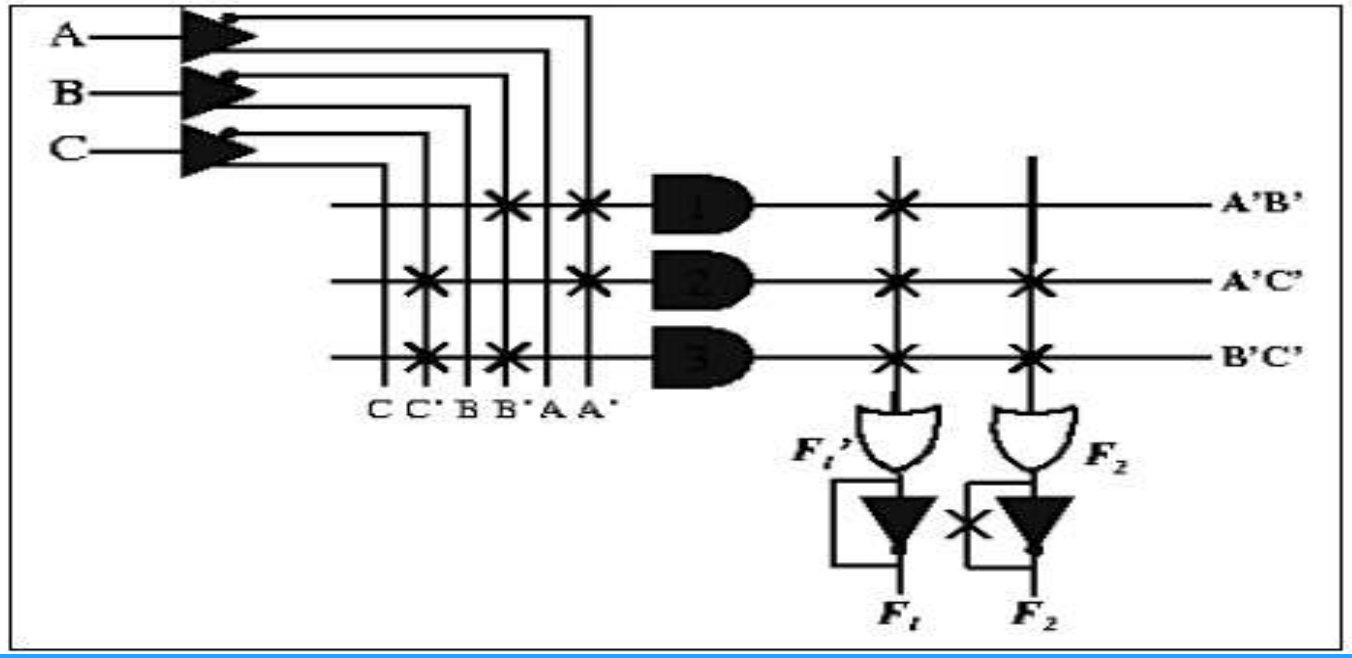
$$f_1 = \overline{\bar{f}_1} = \overline{\bar{a}\bar{b} + \bar{a}\bar{c} + \bar{b}\bar{c}}$$
$$f_2(a, b, c) = \bar{a}\bar{c} + \bar{b}\bar{c}$$

Function Synthesis with PLA

Third Construct a PLA program table from selected functions

Term	Term#	Inputs a	b	c	Outputs f_1	f_2
$\bar{a}\bar{b}$	1	0	0	_	1	_
$\bar{a}\bar{c}$	2	0	_	0	1	1
$\bar{b}\bar{c}$	3	_	0	0	1	1
					C	T

Fourth Construct PLA circuit from PLA program table



Programmable Logic Array

Input Buffers: (See Pg. No. 14-12 J.S. Katre)

- It is used for avoiding the loading of sources connected at the inputs.
- Buffers are of two types namely, inverted buffers and non-inverted buffers.
- One such buffer is used for each of the M input lines.

AND Matrix:

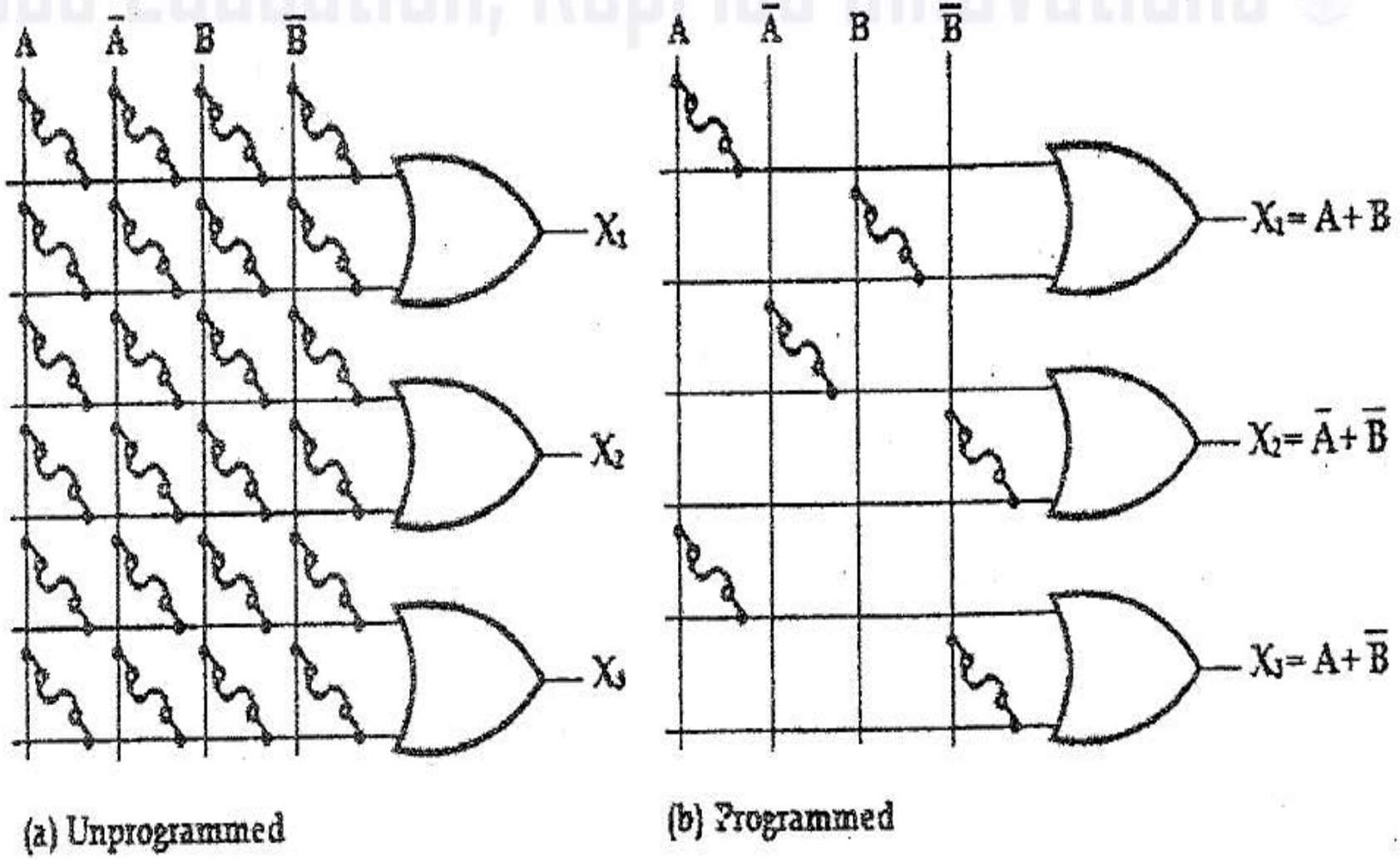
- There are 2M inputs for each AND gate, Po is the output of one of such an AND gate.
- There is a nichrome fuse link connected in series with each input. Thus Po=0 if all the fuse links are intact.
- In an unprogrammed PLA device all the fuse links are intact.
- Thus it consists of 'n' such AND gates formed with the help of diodes. Each AND gate has 2M inputs. The output is thus a product term. So the required product term can be generated by opening the unwanted nichrome fuse links.
- The (x) marks indicate that a connection is present. Each AND gate has 2M inputs which are shown only by a single line.
- When a logic function is to be implemented, we have to program the array. In programming the desired connections are left with the (x) marks and such mark is not used when connection is not required.

Programmable Logic Array(PLA)

OR Matrix:

- The output of AND matrix is used as the inputs to the OR matrix. These are applied to the bases of transistors.
- An OR gate is formed by parallel connected transistors and the load is common and connected in the emitters.
- The required sum terms are obtained by opening the unwanted fuse links.
- It is possible to programme the OR matrix, by open circuiting the unwanted fuse links. The open fuse links are equivalent to a “0” at the input of corresponding OR gate.

The OR Array

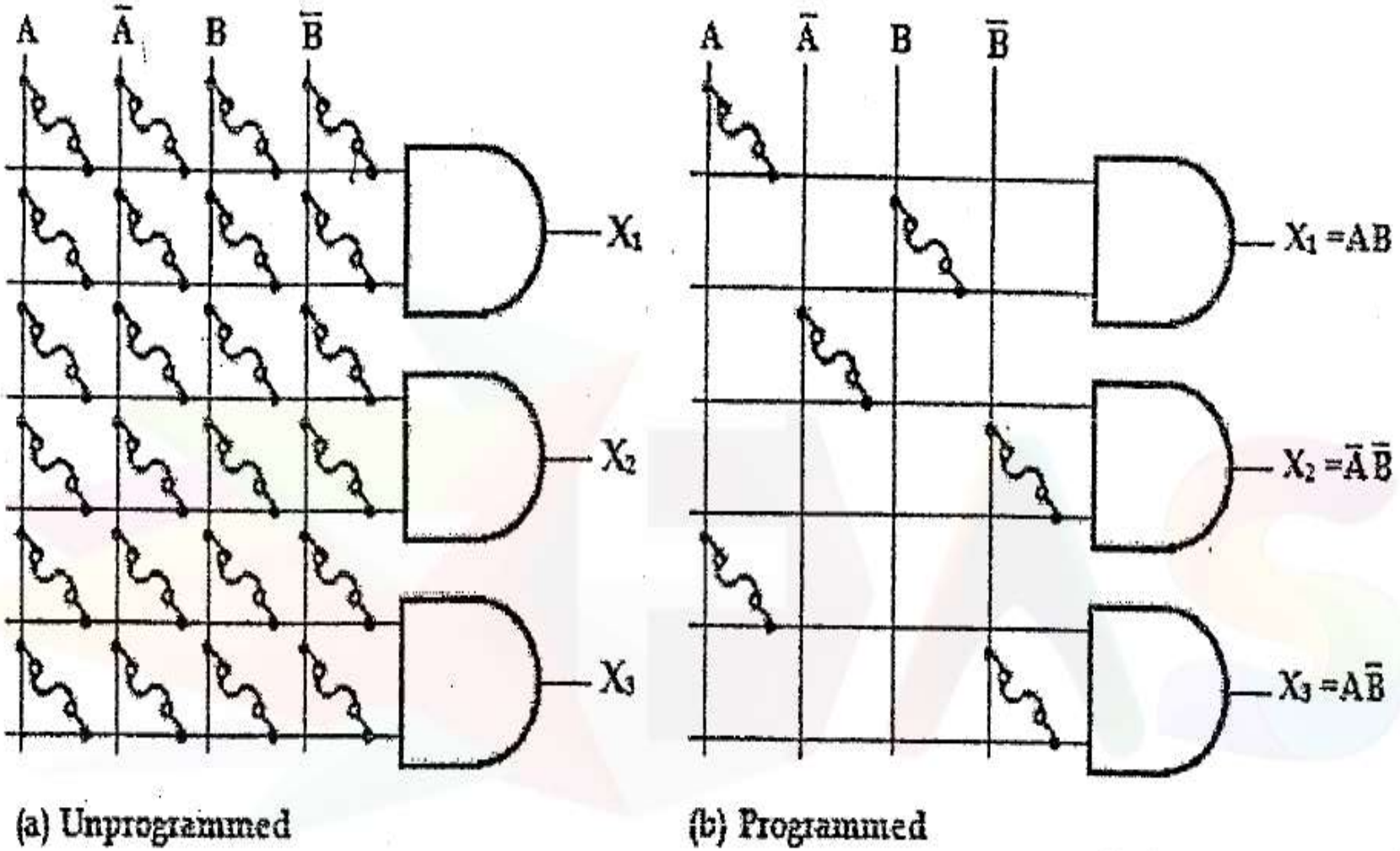


(a) Unprogrammed

(b) Programmed

An example of a basic programmable OR array

The AND Array



An example of a basic programmable AND array

Programmable Logic Array

Invert/Non Invert Matrix:

- The block following the OR matrix is the inverting/non inverting matrix which is basically a programmable buffer.
- It can invert its input if active low output is required. Its input is passed without any inversion, if active high output is required.
- Thus the sum of product terms obtained at the outputs of the OR matrix can be either inverted or passed through as it is.
- For the EX-OR gate, if the fuse link is closed, then one of inputs will be 0. Hence $0 \oplus S = S$. Whereas if the fuse link is opened then that input will be treated as 1. So $1 \oplus S = \overline{S}$. i.e. inversion will take place.

Output Buffers:

- The current sourcing capability of the PLA can be increased by using the output buffer. The PLA outputs are generally TTL compatible.
- In tristate output buffer, the chip enable $\overline{CE}$ input is applied to the output buffers. When $\overline{CE} = 0$, the outputs will be made available.

Programmable Logic Array

Output through Buffers and Flip Flops:

- If the PLA devices are to be used for state machine applications, then the output circuit consists of buffers and flip flops.
- The OR gate outputs are connected to the inputs of the positive edge triggered SR flip flops. The flip flop outputs are then applied to the tristate buffers and the device outputs are obtained as the buffer outputs.

Programming Procedure for PLA:

- PLA programming is similar to the ROM programming.
- Like ROM, it is possible to mask programme the PLA devices.
- According to the requirement specified by the user, the manufacturer prepares a mask and the data patterns are stored on the PLA.
- In field PLA (FPLA), all the nichrome fuse elements are intact at the time of manufacturing. At the time of programming, all the unwanted links are open circuited electrically.
- It is not possible to reprogramme an FPLA.

Programmable Logic Array

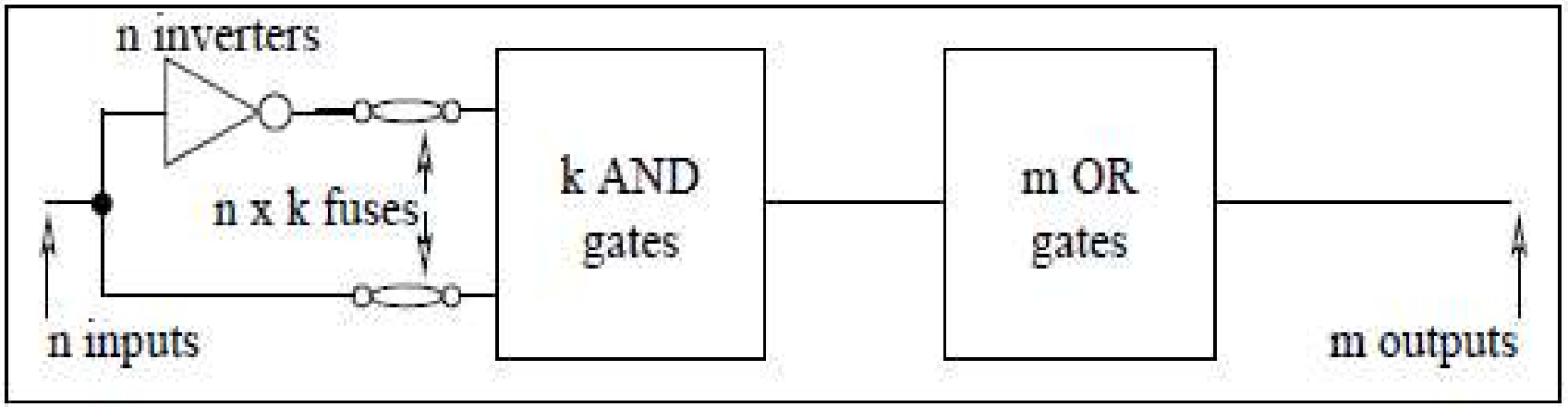
Applications Areas of PLA:

- We can implement the combinational as well as the sequential circuits using PLA.
- For implementing the combinational circuits, the PLA devices with only output buffers are used.
- For implementing the sequential circuits, the PLA devices with flip flop and buffers are included in the output stage.

Programmable Array Logic(PAL)

- Similar to PLA
- Only the connection inputs to ANDs are programmable
- Easier to program than but not as flexible as PLA
- There are feedback connections
- Logic expressions for content information to be stored in PAL must be obtained first, then minimized, and finally programmed into the PAL using a PAL program table

Programmable Array Logic



Field Programmable Logic Array

- FPLA can be programmed by the user.
- Commercial hardware programmer units are available for use in conjunction with certain FPLAs.
- Here we have fused programmable AND array and fused programmable OR array.

Field Programmable Gate Array (FPGA)

- It is an integrated circuit designed to be configured by the customer or designer after manufacturing “field-programmable”.
- It consists of an array of logic blocks with programmable row and column and interconnecting channels surrounded by programmable input output blocks.
- It is generally specified using a hardware description language(HDL).

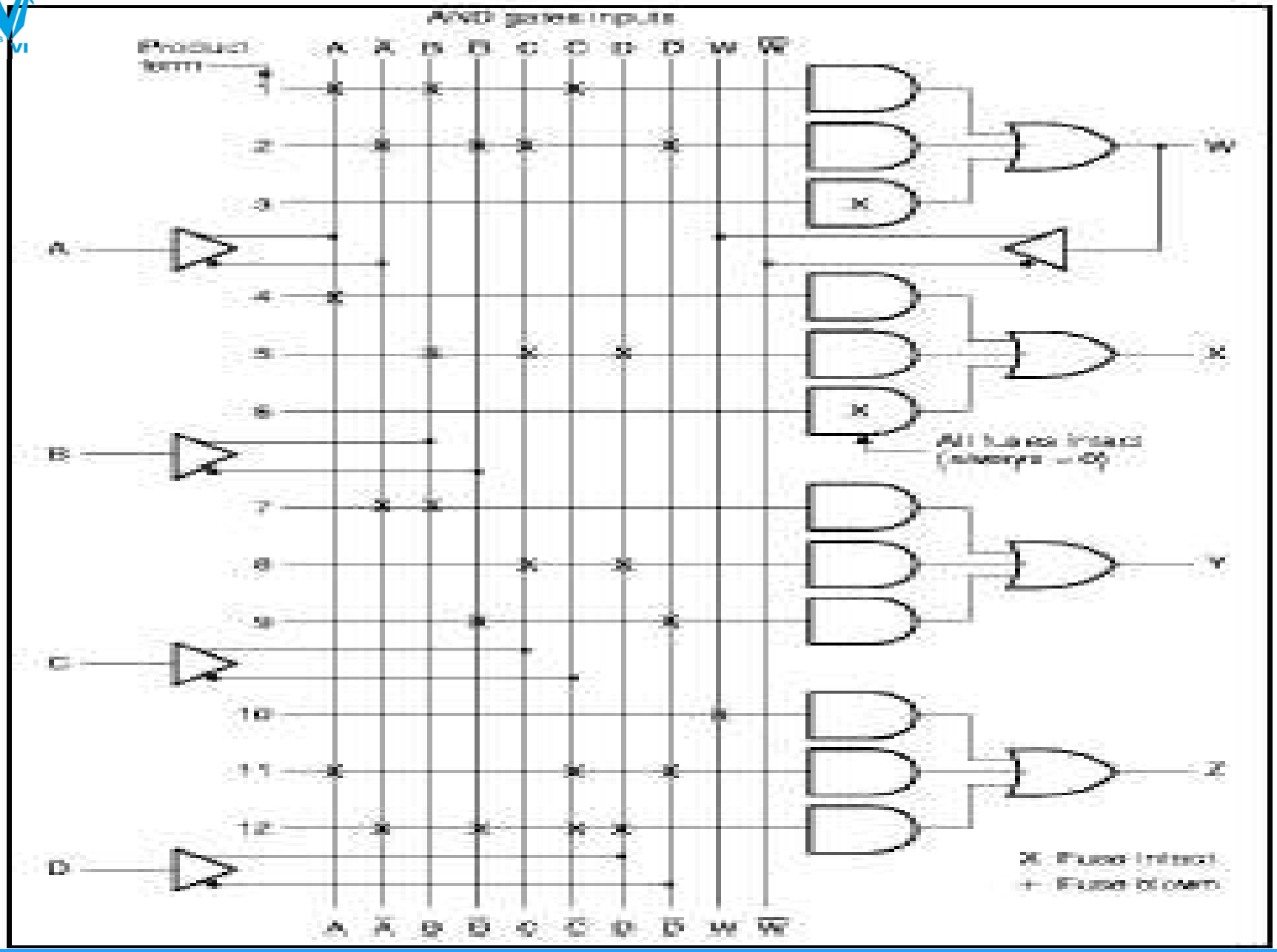
Complex Programmable Logic Devices(CPLD):

- It is a collection of individual PLDs on a single chip, with a programmable interconnection structure that allows the PLDs to get connected as the user wants.

Comparison between PROM, PAL, PLA.

S.No	PROM	PLA	PAL
1	AND array is fixed and OR array is programmable.	Both AND & OR arrays are programmable.	OR array is fixed and AND array is programmable.
2.	Cheaper and simple to use.	Costliest & complex than PAL & PROMs.	Cheaper and simpler.
3.	All minterms are decoded.	AND array can be programmed to get desired minterms.	AND array can be programmed to get desired minterms.
4.	Only Boolean functions in standard SOP form can be implemented using PROM.	Any Boolean functions in standard SOP form can be implemented using PROM.	Any Boolean functions in standard SOP form can be implemented using PROM.

Programming a PAL



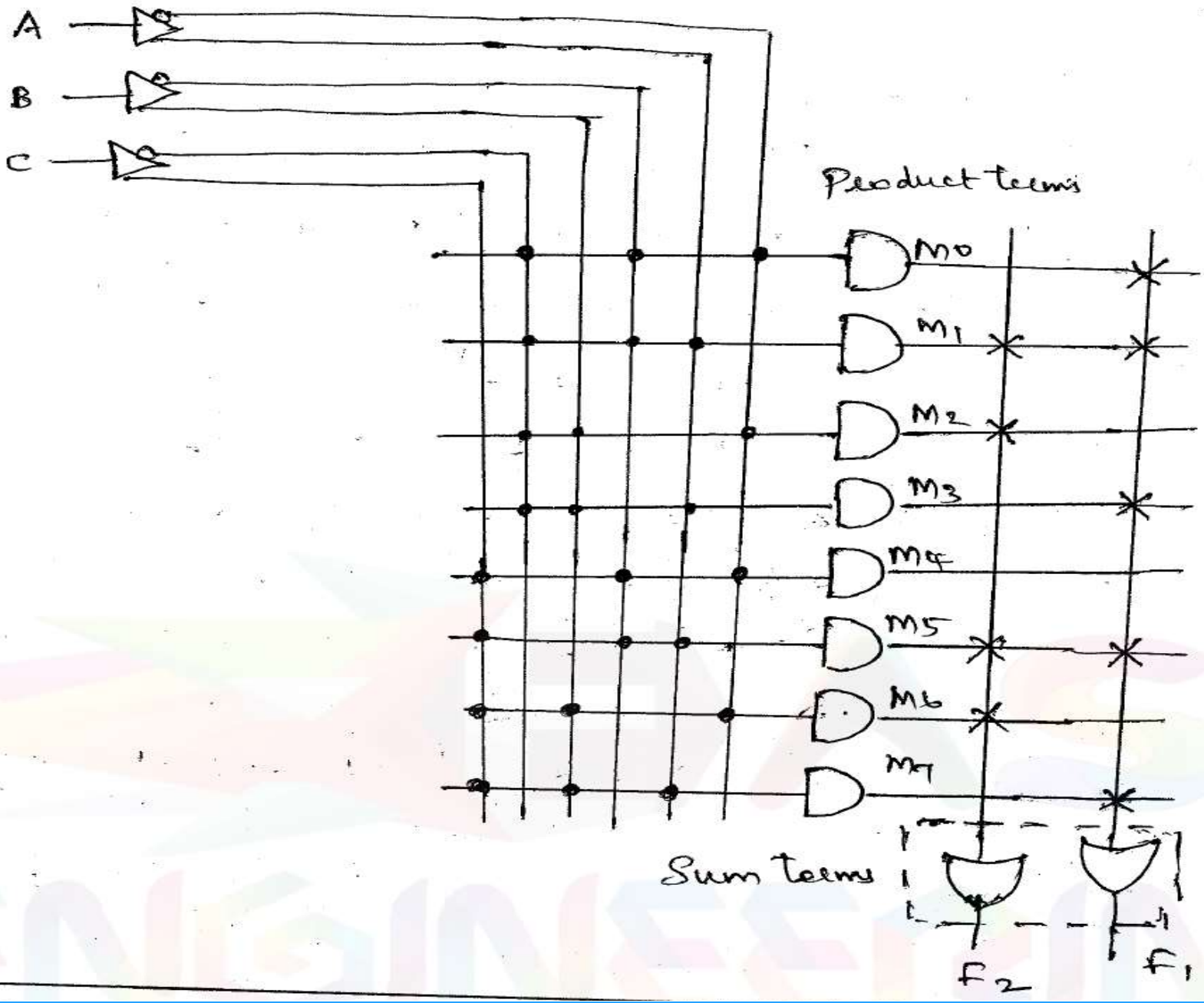
Using PROM realize the following expression.

$f_1(A,B,C) = \Sigma m(0,1,3,5,7)$

$f_2(A,B,C) = \Sigma m(1,2,5,6)$

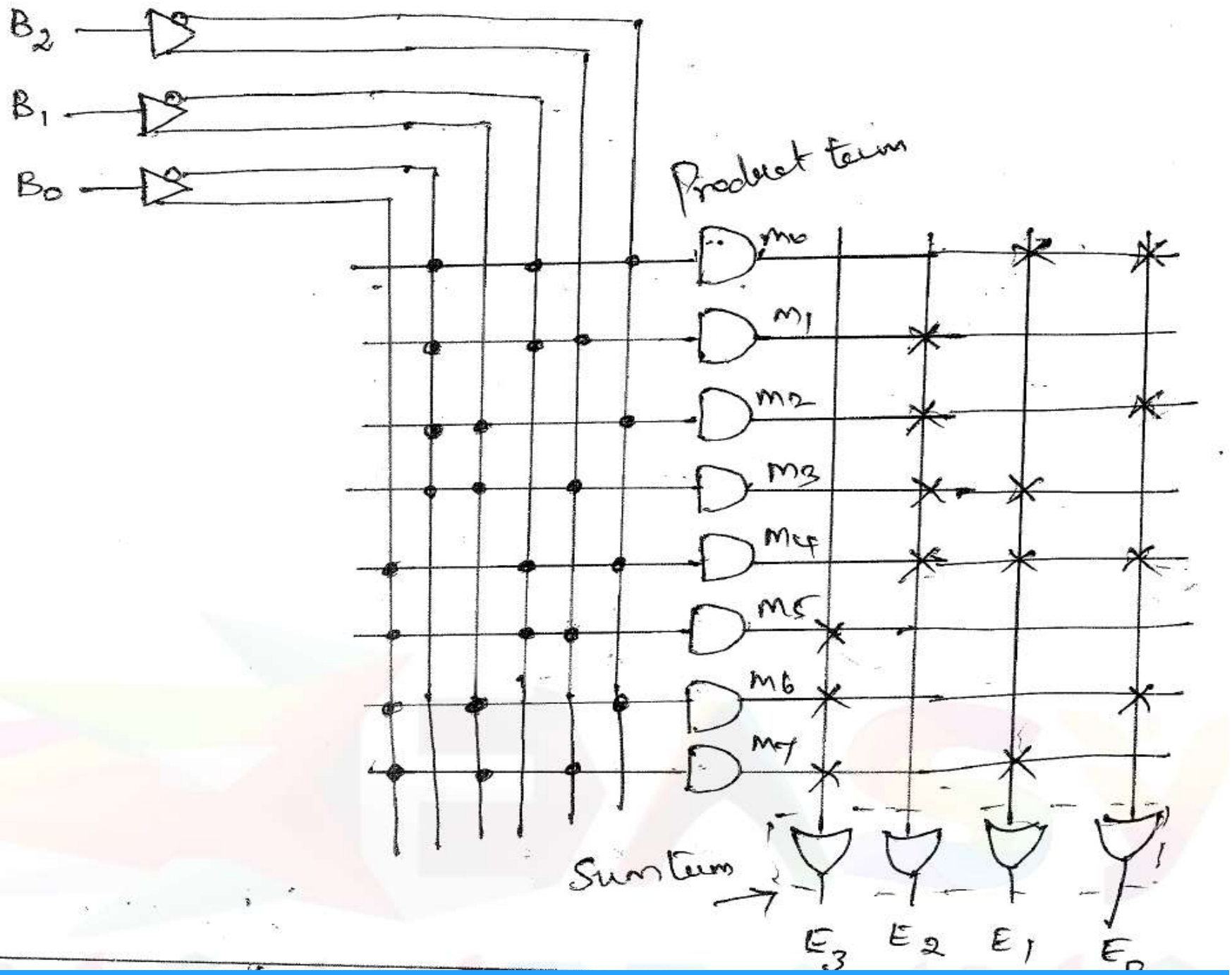
Truth table.

A	B	C	f_1	F_2
0	0	0	1	0
0	0	1	1	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	0


A
B
C
Product terms
M0
M1
M2
M3
M4
M5
M6
M7
Sum terms
F2
F1

Design a combinational circuit using PROM. The circuit accepts 3-bit binary and generates its equivalent excess-3 code.

B_2 B_1 B_0	E_3 E_2 E_1 E_0
0 0 0	0 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	0 1 1 1
1 0 1	1 0 0 0
1 1 0	1 0 0 1
1 1 1	1 0 1 0


B2
B1
B0
Product term
m0
m1
m2
m3
m4
m5
m6
m7
Sum term
E3
E2
E1
E0

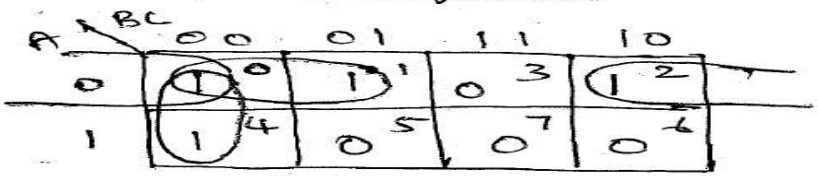
1. Implement the Combinational circuit with a PLA having 3 inputs, 4 product terms and 2 outputs for the functions. $f_1(A,B,C) = \Sigma m(0,1,2,4)$

$f_2(A,B,C) = \Sigma m(0,5,6,7)$.

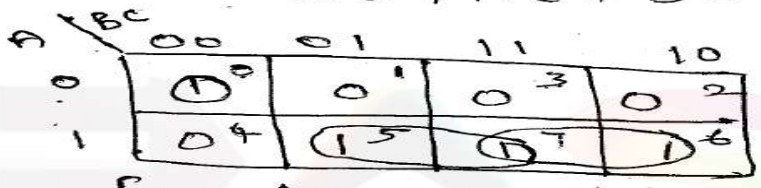
Truth table

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

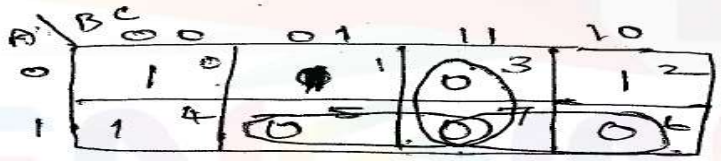
K-map simplification



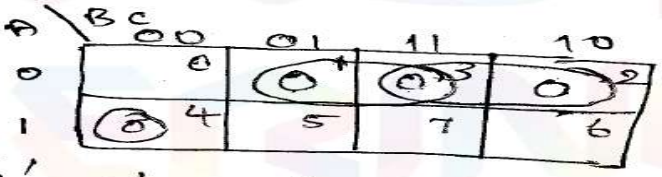
$F_1 = A'B' + A'C' + B'C'$.



$f_2 = AC + AB + A'B'C'$.



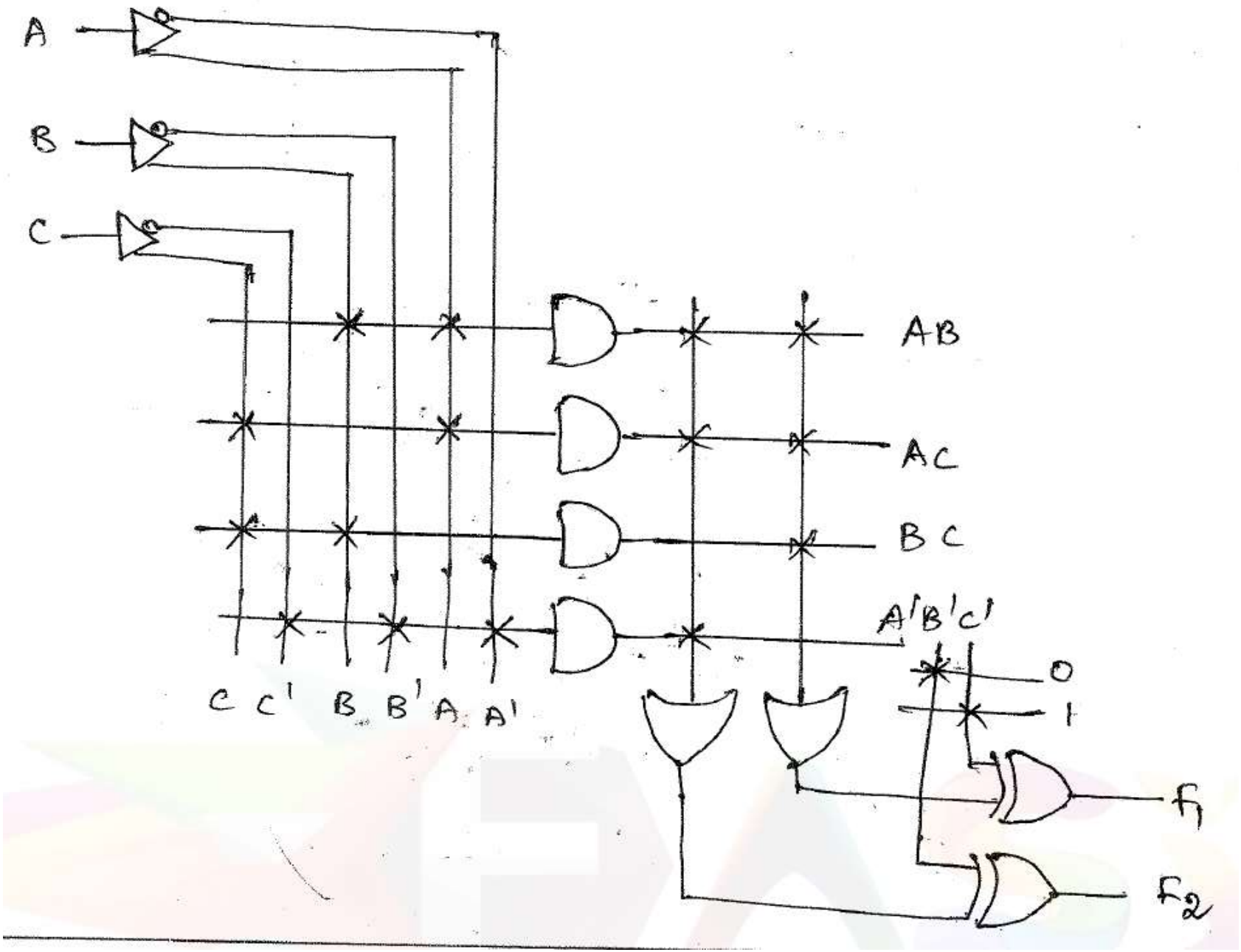
$F_1' = AC + BC + AB$.



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

Program table

	Product term	Inputs			Output	
		A	B	C	$F_1(C)$	$F_2(T)$
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


A
B
C
AB
AC
BC
A'B'C'
C C' B B' A A'
0
1
F1
F2

Implement the following function using PLA

$f_1(A,B,C) = \Sigma m(1,2,4,6)$

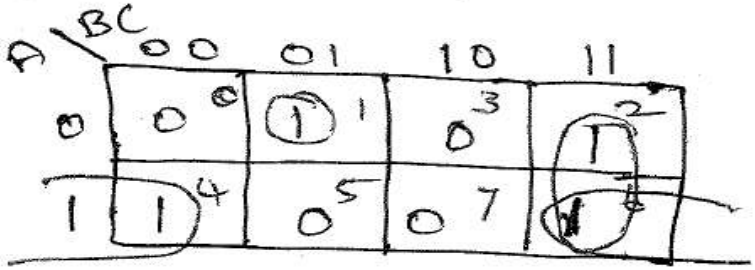
$F_2(A,B,C) = \Sigma m(0,1,6,7)$

$F_3(A,B,C) = \Sigma m(2,6)$

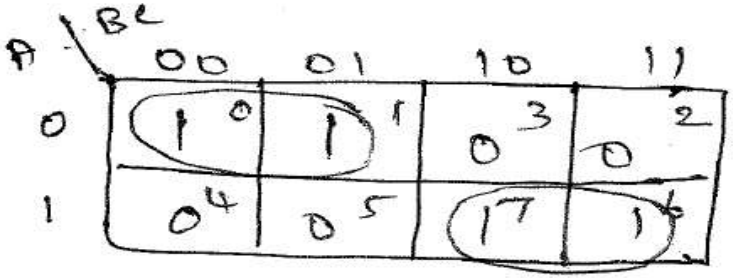
Truth table

A	B	C	f_1	f_2	f_3
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	1	0

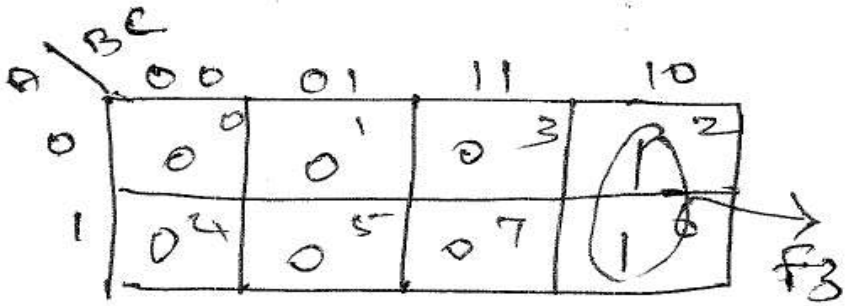
K map simplification



$f_1 = A'B'C + AC' + BC'$



$F_2 = A'B' + AB$



$f_3 = BC'$

Implement the following function using PLA

$f_1(A,B,C) = \sum m(1,2,4,6)$

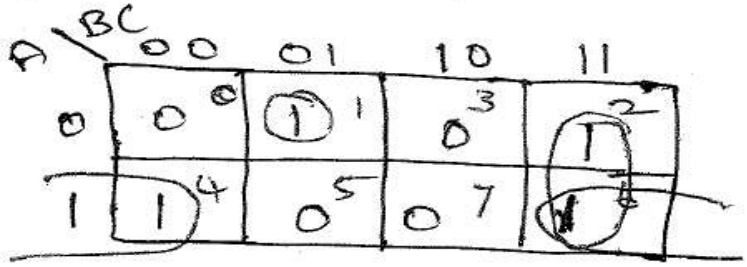
$F_2(A,B,C) = \sum m(0,1,6,7)$

$f_3(A,B,C) = \sum m(2,6)$

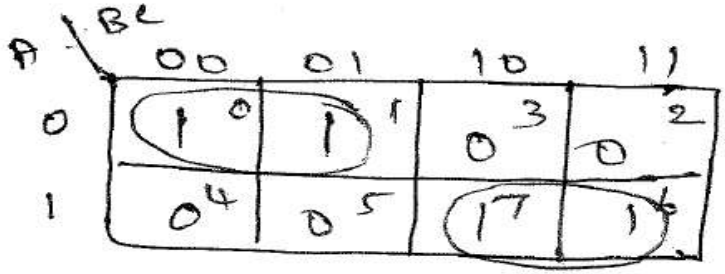
Truth table

A	B	C	f_1	f_2	f_3
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	1	0

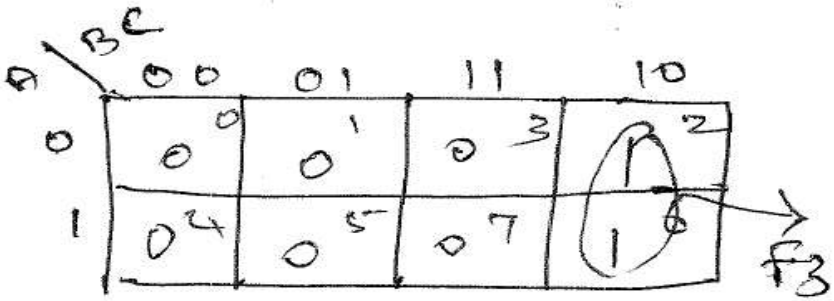
K map simplification



$f_1 = A'B'C + AC' + BC'$



$f_2 = A'B' + AB$

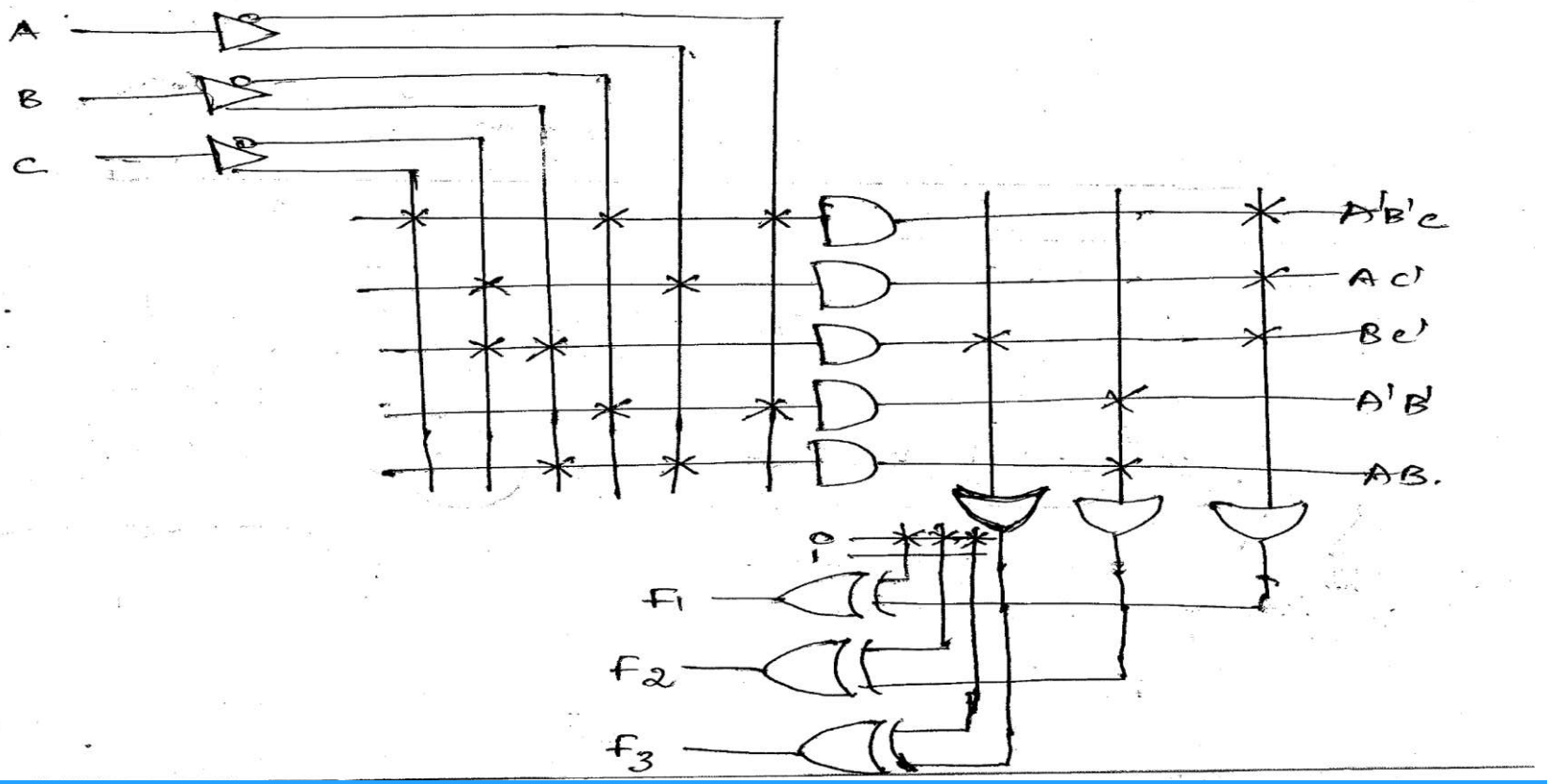


$f_3 = BC'$

Program table.

	Product term	Inputs			Outputs		
		A	B	C	$F_1(T)$	$F_2(T)$	$F_3(T)$
$A'B'C$	1	0	0	1	1	—	—
AC'	2	1	—	0	1	—	—
BC'	3	—	1	0	1	—	1
$A'B'$	4	0	0	—	—	1	—
AB	5	1	1	—	—	1	—

PLA diagram



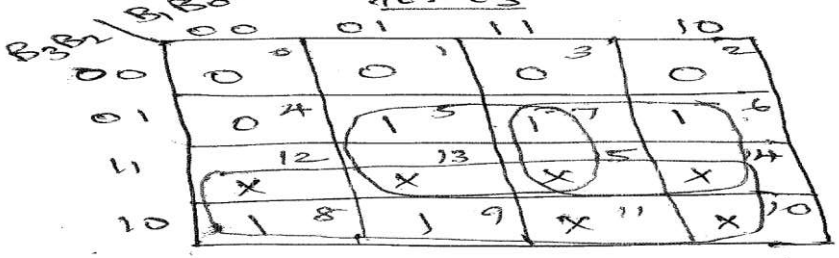
Design a BCD to Excess-3 code convertor and implement using suitable PLA.

Truth table.

BCD code.				Excess-3 code.			
B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

K-map Simplification

For E_3

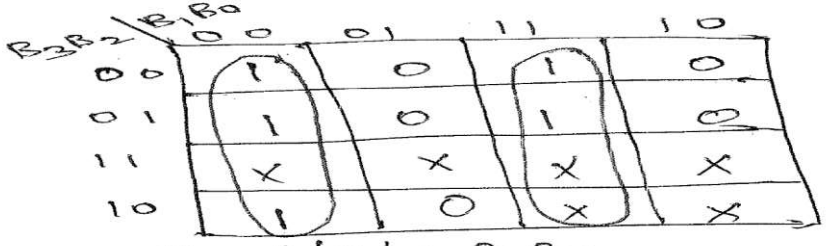


$E_3 = B_3 + B_2 B_0 + B_2 B_1$

For E_2

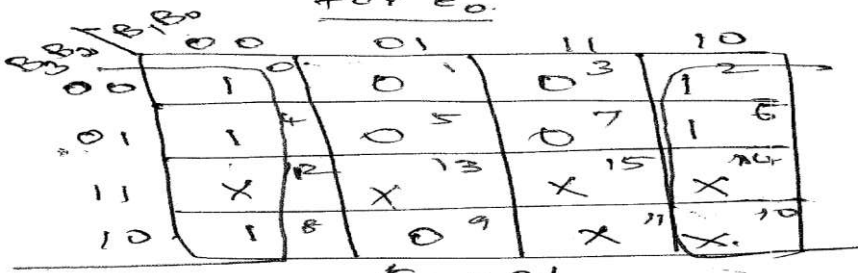
$E_2 = B_2 B_1' B_0' + B_2' B_0 + B_2' B_1$

For E_1



$E_1 = B_1' B_0' + B_1 B_0$

For E_0

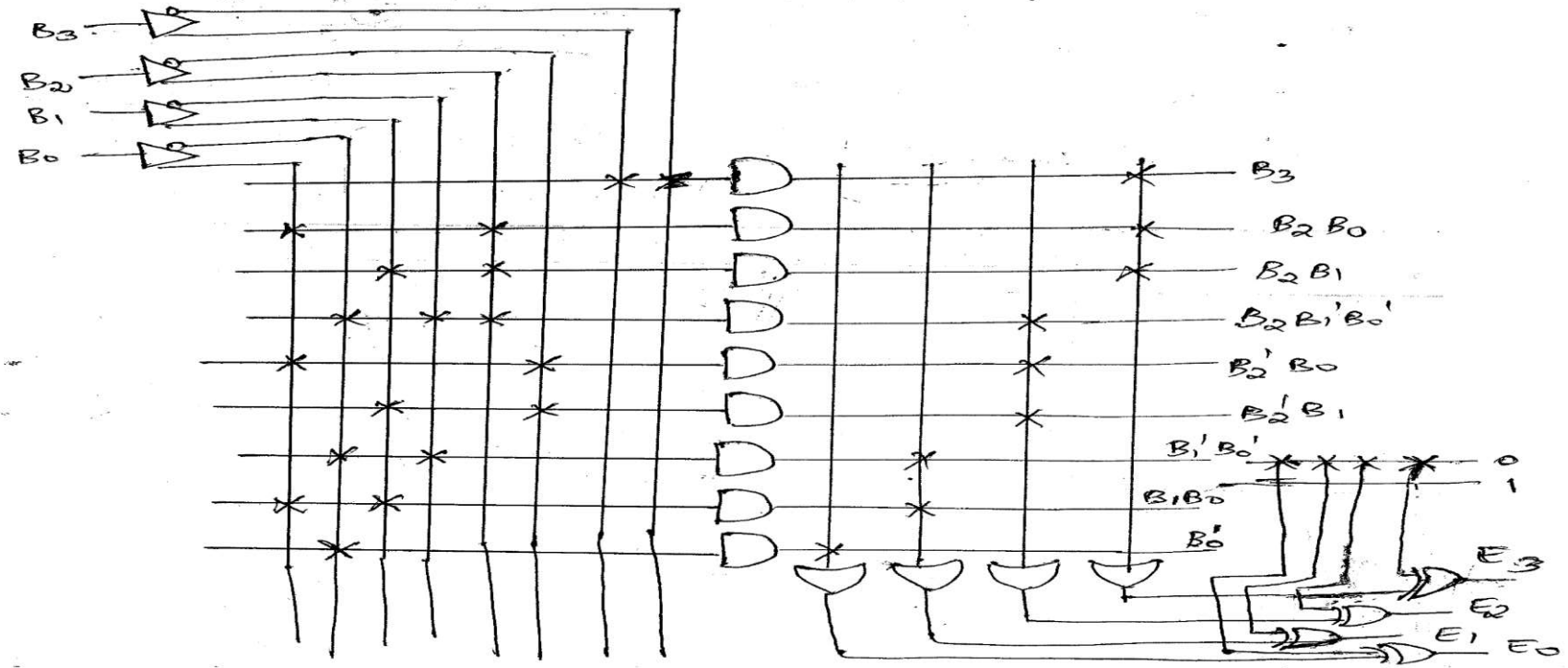


$E_0 = B_0'$

Program table :

	Product term	Inputs				Outputs			
		B_3	B_2	B_1	B_0	$E_3(T)$	$E_2(T)$	$E_1(T)$	$E_0(T)$
B_3	1	1	–	–	–	1	–	–	–
$B_2 B_0$	2	–	1	–	1	1	–	–	–
$B_2 B_1$	3	–	1	1	–	1	–	–	–
$B_2 B_1' B_0'$	4	–	1	0	0	–	1	–	–
$B_2' B_0$	5	–	0	–	1	–	1	–	–
$B_2' B_1$	6	–	0	1	–	–	1	–	–
$B_1' B_0'$	7	–	–	0	0	–	–	1	–
$B_1 B_0$	8	–	–	1	1	–	–	1	–
B_0'	9	–	–	–	0	–	–	–	1

PLA diagram :



Programmable array logic (PAL).

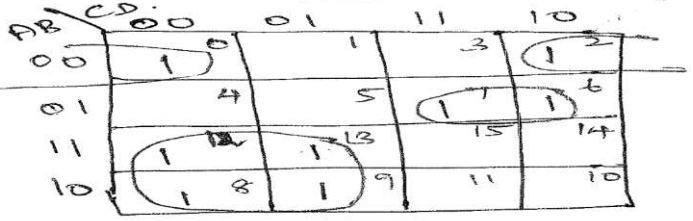
$W(A,B,C,D) = \Sigma(0,2,6,7,8,9,12,13)$

$X(A,B,C,D) = \Sigma(0,2,6,7,8,9,12,13,14)$

$Y(A,B,C,D) = \Sigma(2,3,8,9,10,12,13)$

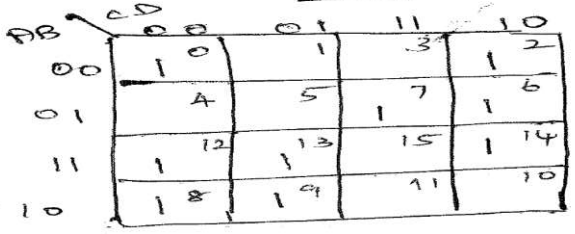
$Z(A,B,C,D) = \Sigma(1,3,4,6,9,12,14)$

For W



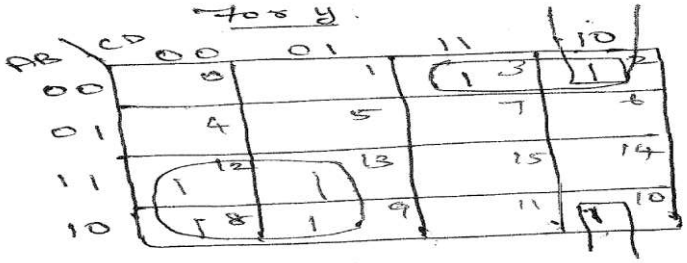
$W = \bar{A}\bar{B}\bar{D} + \bar{A}BC + A\bar{C}$

For X



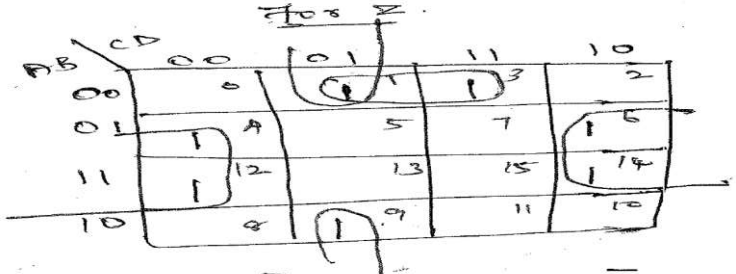
$X = \bar{A}\bar{B}\bar{D} + \bar{A}BC + A\bar{C} + BC\bar{D}$

For Y



$Y = \bar{A}\bar{B}C + \bar{B}C\bar{D} + A\bar{C}$

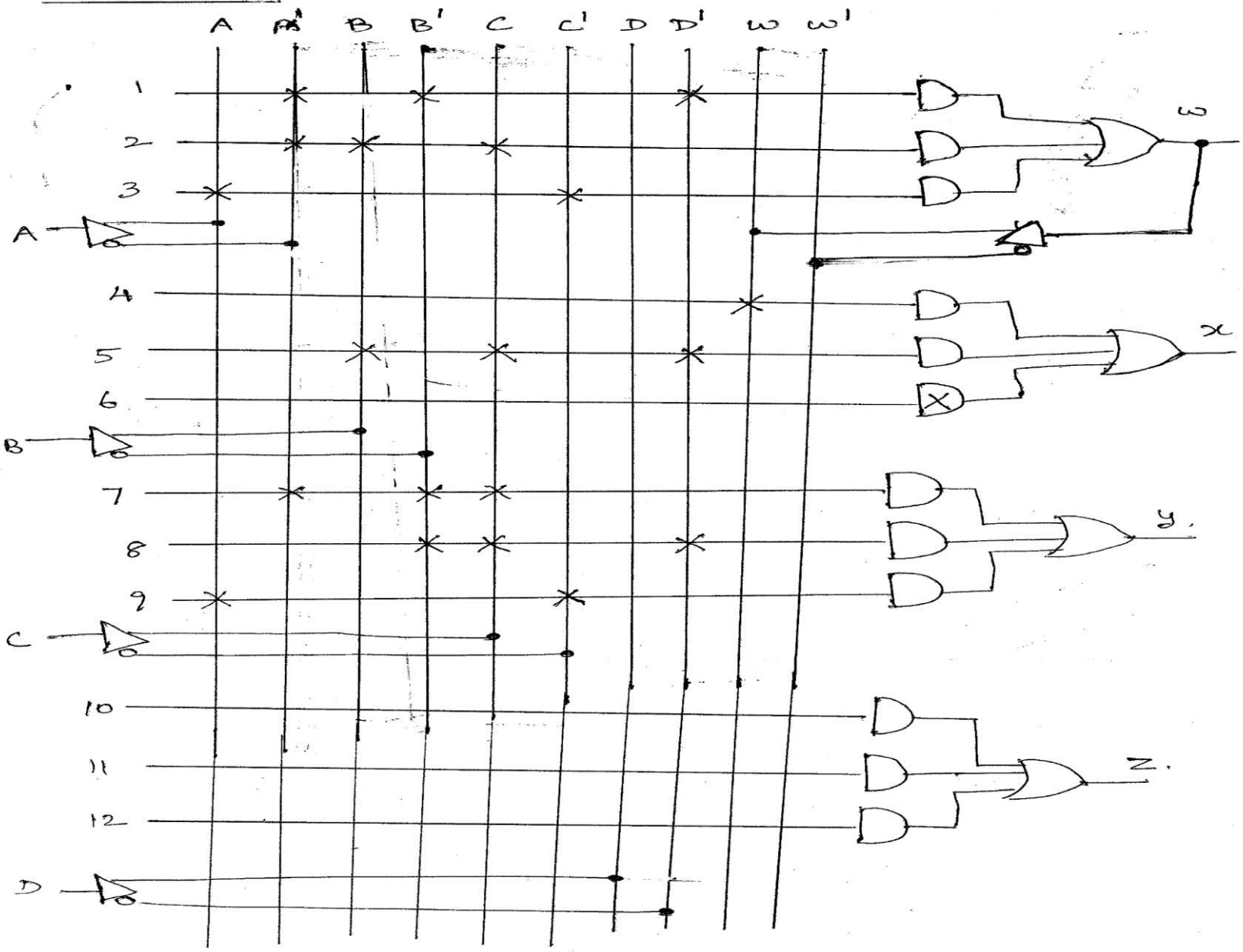
For Z



$Z = \bar{A}\bar{B}D + \bar{B}\bar{C}D + B\bar{D}$

Product term	AND inputs A	B	C	D	W	Outputs
1. $\bar{A}\bar{B}\bar{D}$	0	0	—	0	—	$W = \bar{A}\bar{B}\bar{D} + \bar{A}BC + A\bar{C}$
2. $\bar{A}BC$	0	1	1	—	—	
3. $A\bar{C}$	1	—	0	—	—	
4. W	—	—	—	—	1	$X = W + BC\bar{D}$
5. $BC\bar{D}$	—	1	1	0	—	
6. —	—	—	—	—	—	
7. $\bar{A}\bar{B}C$	0	0	1	—	—	$Y = \bar{A}\bar{B}C + \bar{B}C\bar{D} + A\bar{C}$
8. $\bar{B}C\bar{D}$	—	0	1	0	—	
9. $A\bar{C}$	1	—	0	—	—	
10. $\bar{B}\bar{D}D$	0	0	—	1	—	$Z = \bar{A}\bar{B}D + \bar{B}\bar{C}D + B\bar{D}$.
11. $\bar{B}\bar{C}D$	—	0	0	1	—	
12. $B\bar{D}$	—	1	—	0	—	

PAL diagram.



End
of
Unit-4