Unit-V

Memory And Programmable Logic Devices

Introduction:-

→ A memory unit is a collection of storage cells with associated circuits needed to transfer information in and out of the device.

→ A memory unit stores binary information in group of bits called words.

→ A word is a group of 1's and 0's and may represent a number, an instruction, one or more alphanumeric characters or any other binary-coded information.

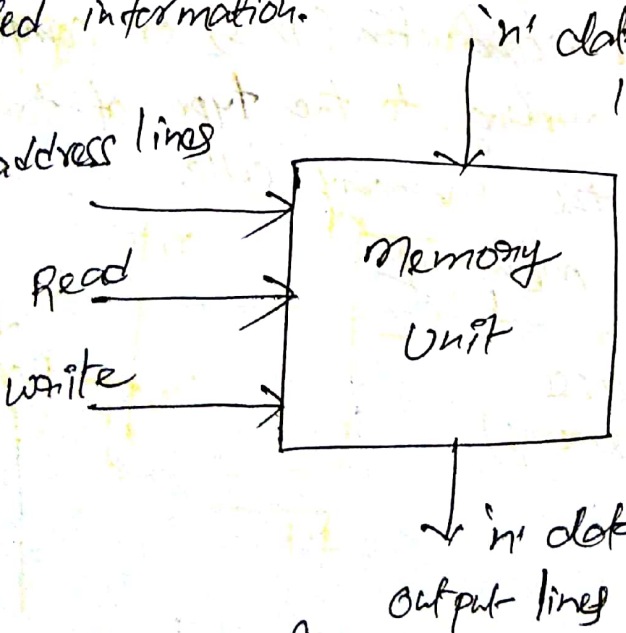


fig:- Block diagram of a memory unit

→ The communication between a memory and its environment is achieved through data i/p/o/p lines, address selection lines and control lines that specify the direction of transfer.

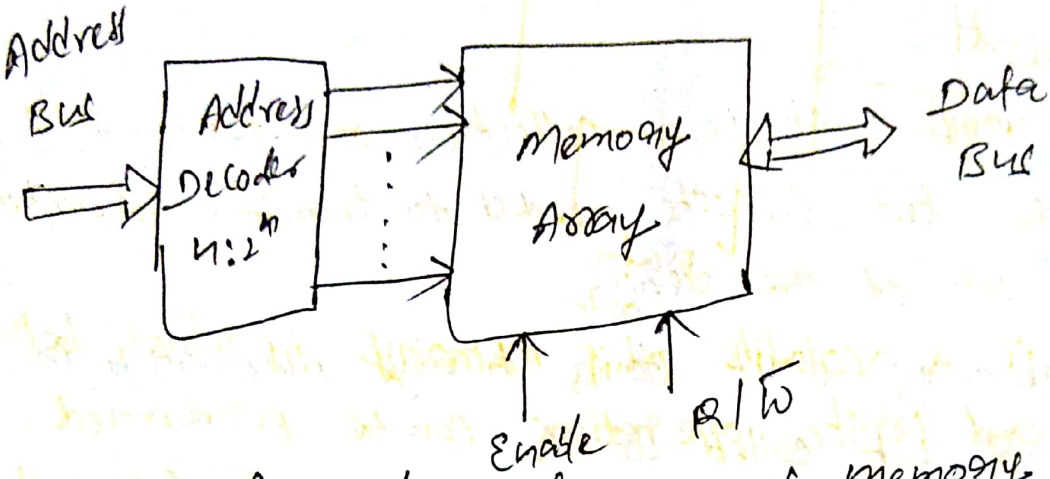
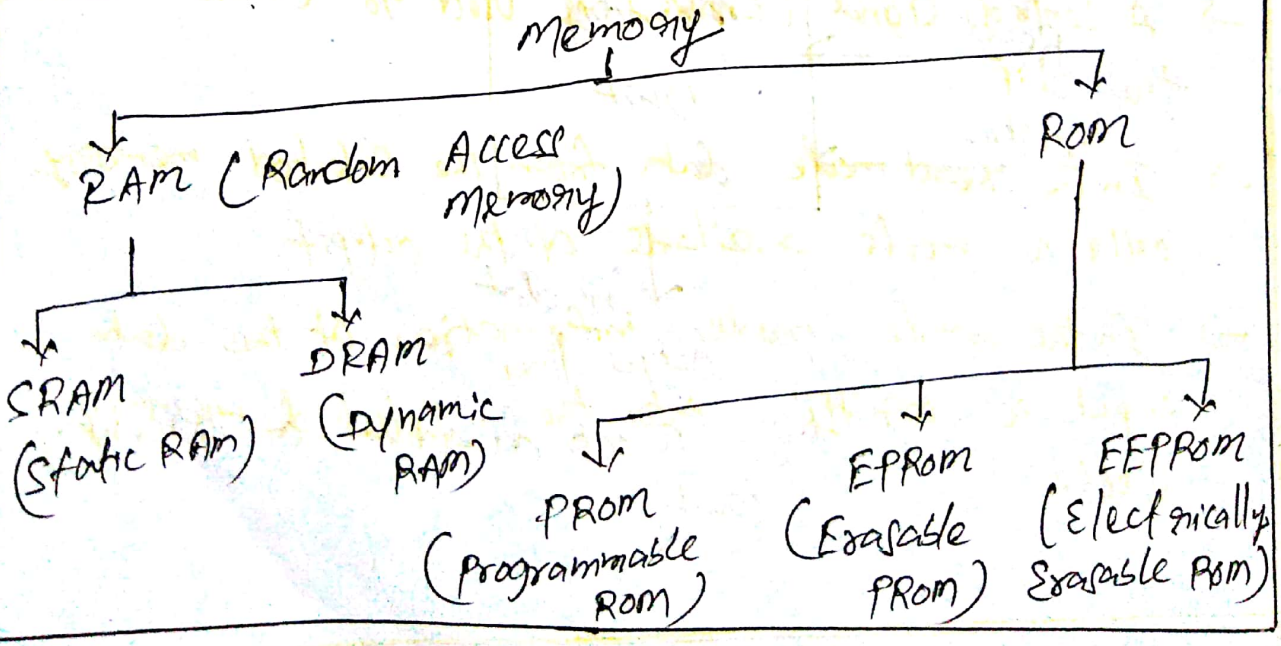


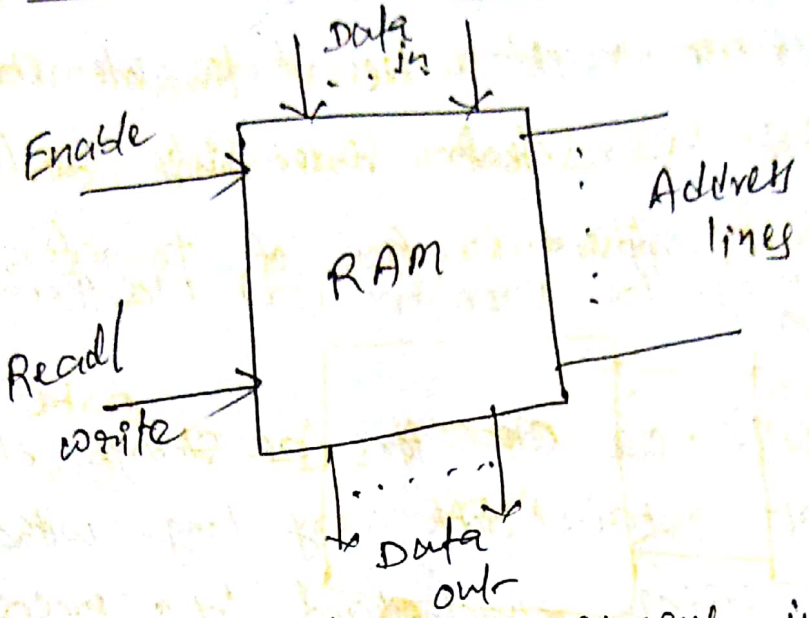
fig:- Block diagram of memory operation.

classification of Memories:-

→ Memories are usually classified as either bipolar or metal oxide semiconductor (MOS) or complementary MOS (CMOS) according to the type of transistor used to construct the memory cells.



Random Access Memory (RAM) :-



→ RAM is a volatile chip memory in which both read and write operations can be performed.

→ RAMs are volatile because the stored data will be lost once the d.c. power applied to the flip flops is removed.

→ Random access means a bit (0 or 1) can be written (stored) in any cell or read (detected) from any cell.

→ A control signal (Enable) is used to enable or disable the chip.

→ In the read mode, data from the selected memory cells is made available at the output.

→ In the write mode, information at the data input is written into the selected memory cells.

Static RAM:

→ Static RAMs use flipflops as storage elements and can therefore store data indefinitely as long as d.c. power is applied.

→ ~~Both~~ may be Bipolar (or) MOS Technology.

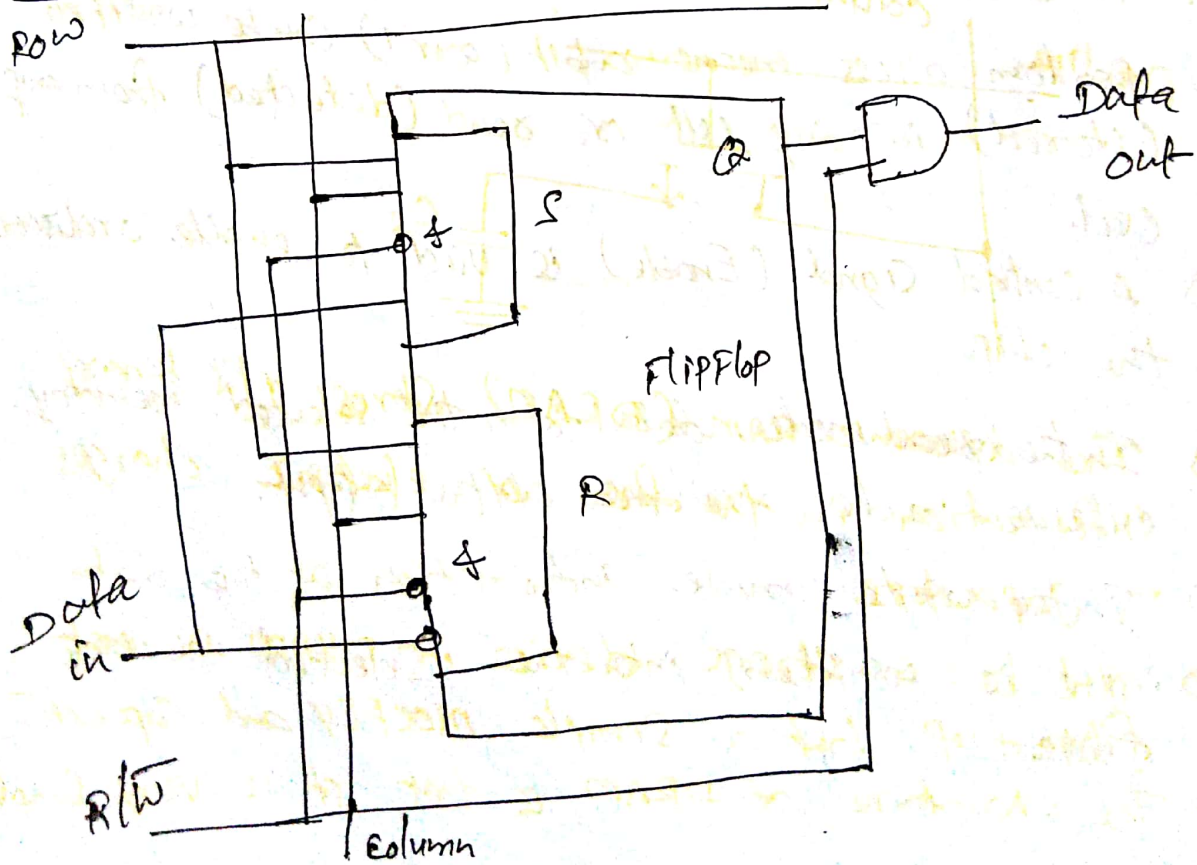
Dynamic RAM:

→ Dynamic RAM's use capacitors as storage elements and cannot retain data very long without the capacitors being recharged by a process called refreshing.

→ DRAM's can store much more data than SRAMs.

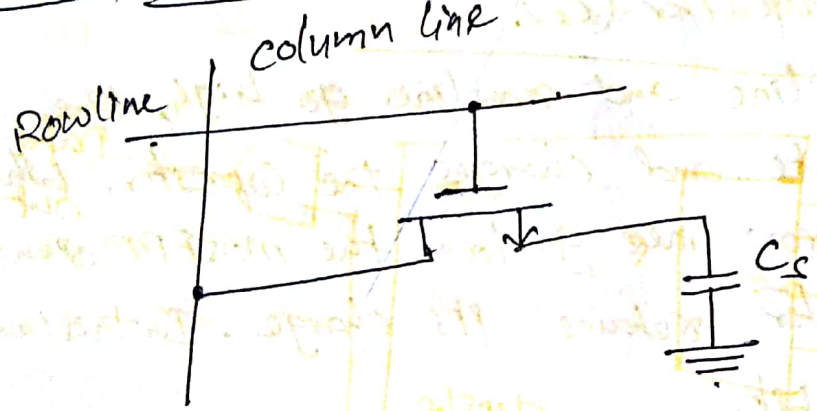
→ DRAM's are available in only MOS technology.

Static RAM Cell:-



→ The cell is a single storage element in the memory.
→ The cell is selected by high values on the row and column lines.
→ The input data bit (0 or 1) is written into cell by setting the flip-flop for a 1 and resetting the flip flop for a 0 when R/W is line is low.
→ When R/W line is high, the flip-flop is unaffected.
→ It means that the stored bit (data) is gated to the data out line.

Dynamic ~~Bipolar~~ RAM cell:-



→ The dynamic RAM (DRAM) stores bit's binary information in the form of electric charges on capacitors.
→ The basic storage device in DRAM is not a flip-flop but a simple MOSFET and capacitor.
→ The advantage of DRAM is that it is very simple.

thus allowing very large memory arrays to be constructed on a chip at a lower cost per bit.

→ The disadvantages of DRAM is that the storage capacitor cannot hold its charge over an extended period of time and will lose the stored data bit unless its charge is refreshed periodically.

→ In DRAM memory cell, a bit of data is stored as charge on storage capacitor, where the presence or absence of charge determines the value of stored bit 1 or 0.

→ The ~~MOSFET~~ DRAM cell includes a MOSFET and a storage capacitor (C_s).

→ when column line and row line go high, the MOSFET conducts and charges the capacitor. When column and row lines go low, the MOSFET opens and the capacitor retains its charge. In this way it stores 1 bit.

Rom:

⇒ Read only memory.

⇒ Rom is a memory device in which permanent binary information is stored.

→ It is a non-volatile memory.

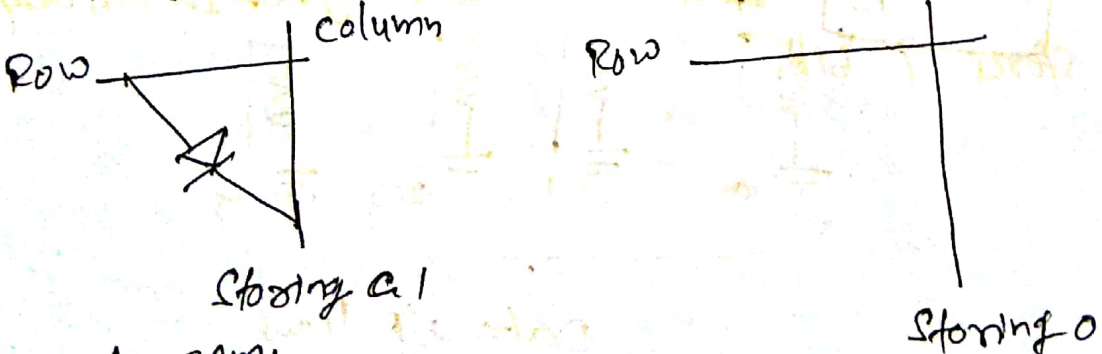
Types

① masked Rom or Rom
② Programmed Rom (PRom)
③ Erasable PROM (EPROM)
④ Electrically Erasable PROM (EEPROM)

Rom cell:

Diode Rom:

→ The presence of a connection from a row line to the anode of the diode represents a 1 at that location because when the row line is taken High, all the diodes are turn ON and connect the HIGH (1) to the associated column lines.

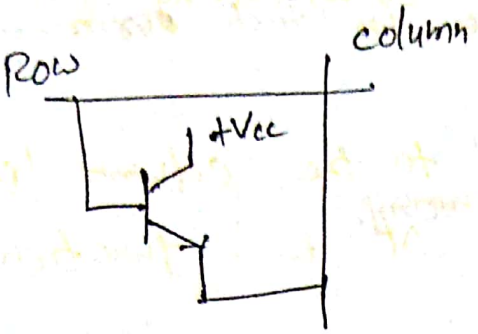


Bipolar RAM:

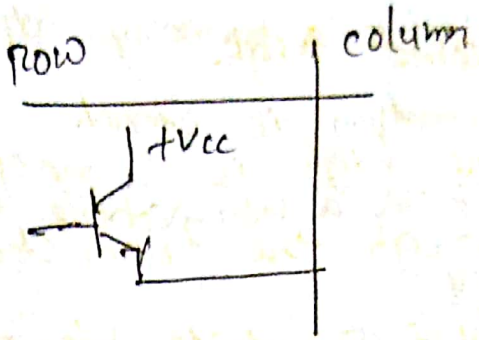
⇒ The presence of a connection from a row line to the base of a transistor represents a 1 at that location.

→ When the row line is taken HIGH, all transistors with base connection to that row line turn ON and connect the high (1) to the associated column lines.

→

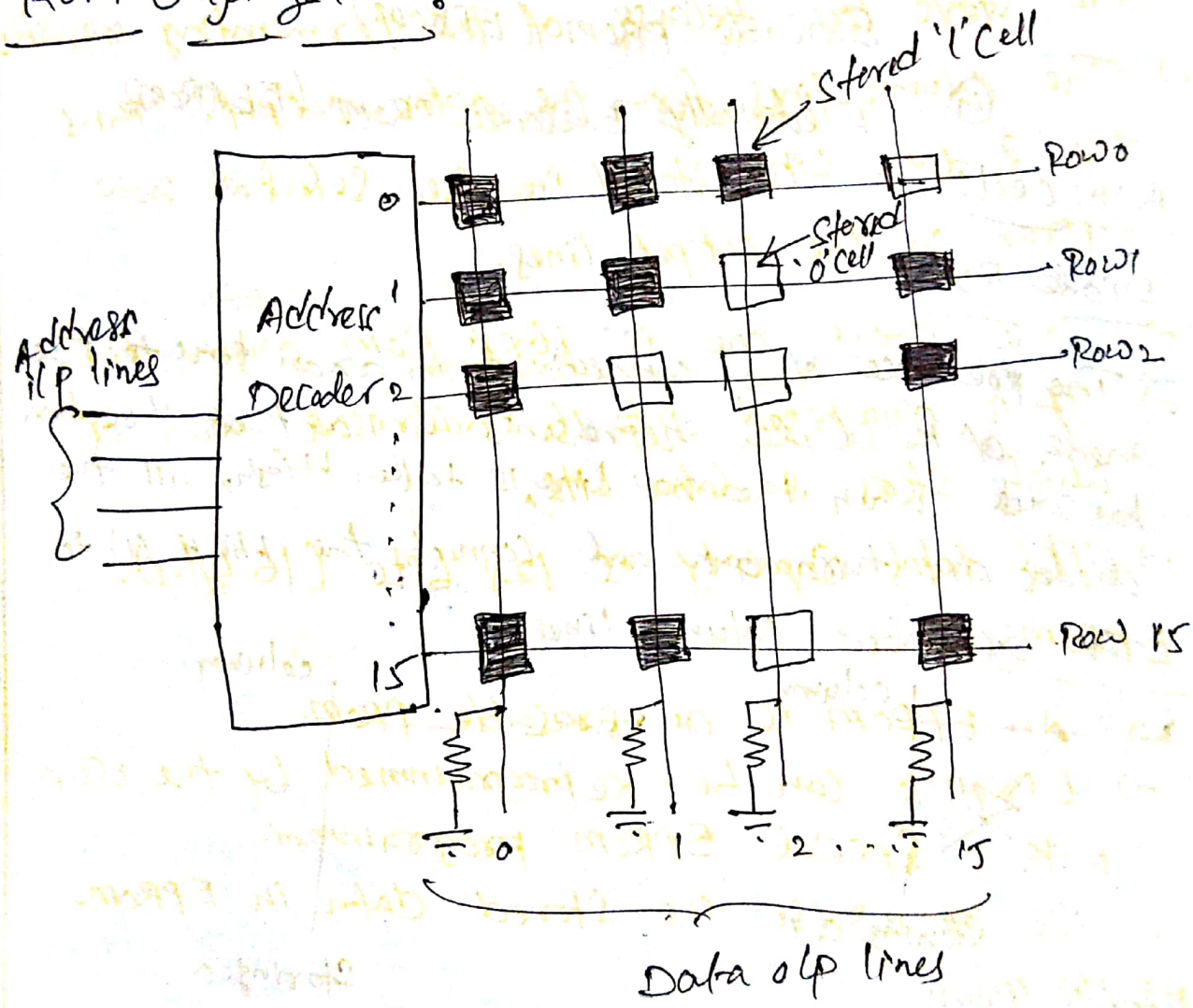


Sorting a 1



Sorting a 0

ROM organization :-



→ A simple ROM organization is shown above.

→ The dark squares represent ROM cells stored 1's and the light squares represent ROM

Cells stored 0's.

→ When a 4-bit binary address is applied to the address inputs, the corresponding row line become high.

→ The high is connected to the column lines through the transistors at each junction (cell) where a 1 is stored.

→ At each cell where a 0 is stored, the column line stays low because of the terminating resistor.

→ The column lines form the data output. Thus the 8 data bits stored in the selected row appear on the output lines.

→ The above one is 16x8 Rom organization.

→ ~~It~~ is organized into 16 addresses, each of which stores 8 data bits.

→ It's total capacity is 128 bits (16 bytes).

EPROM:

→ An EPROM is an erasable PROM.

→ EPROMs can be reprogrammed by the user with a special EPROM programmer.

→ We can erase the stored data in EPROM.

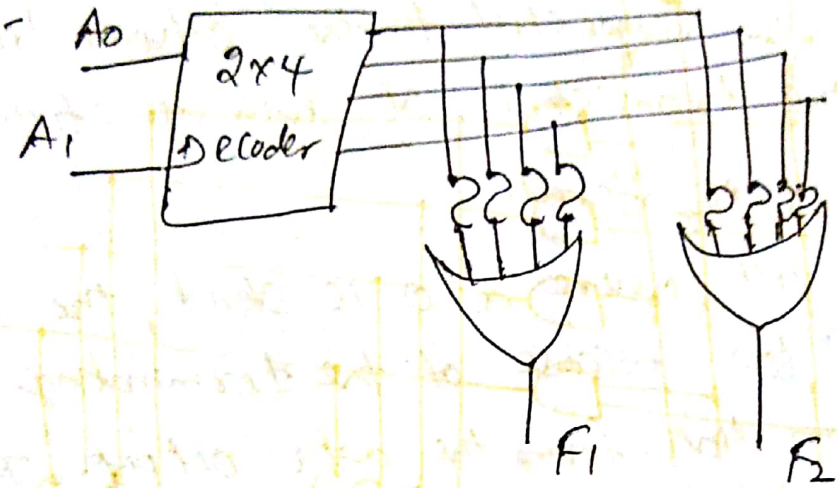
EEPROM:

→ Electrically Erasable PROM can be both erased and programmed by the application of controlled electric pulses to the IC in the circuit.

Implementation of Combinational Logic Circuits using ROM :-

Ex① Draw 4×2 ROM with AND-OR gates.

Sol:-

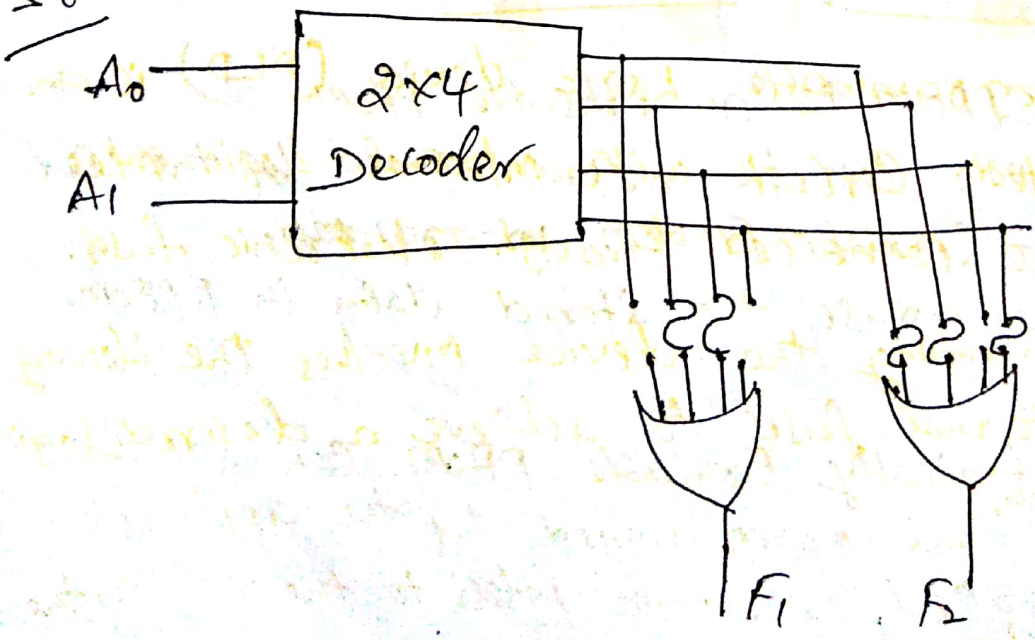


Ex② Implement the following Boolean functions using ROM.

Sol:- $F_1(A_1, A_0) = \sum(1, 2)$

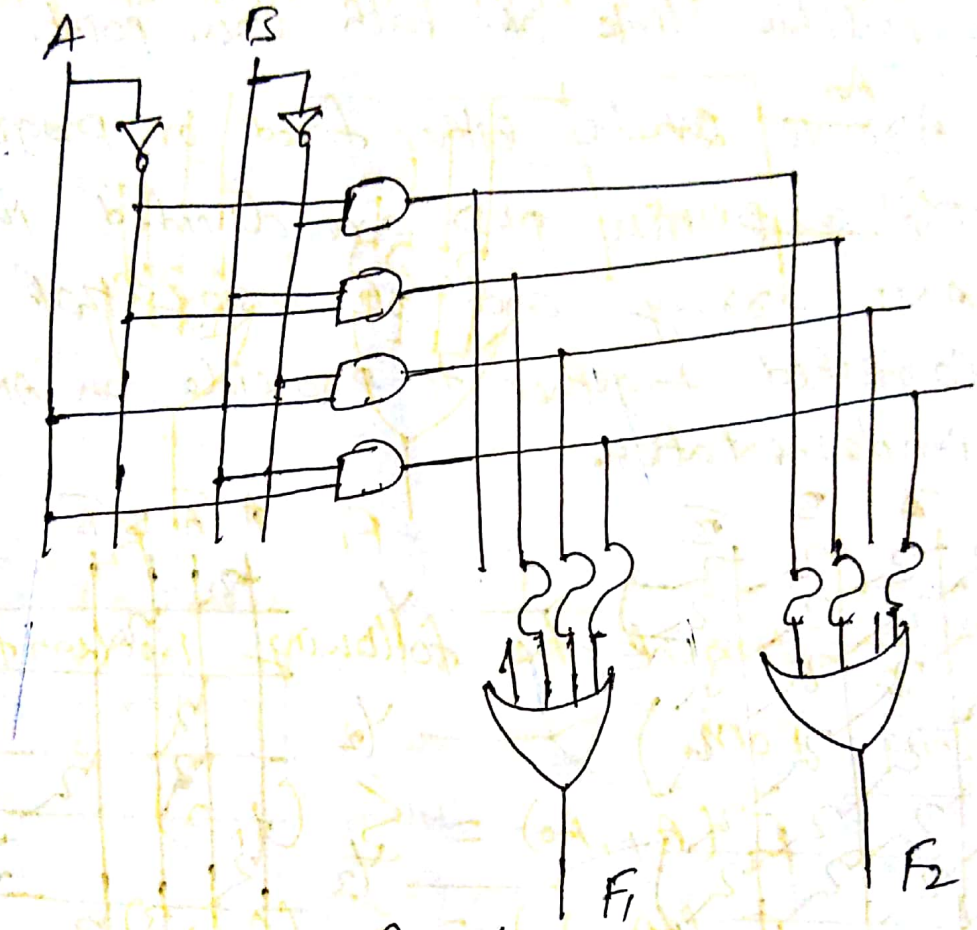
$F_2(A_1, A_0) = \sum(0, 1, 3)$.

Sol:-



Ex⑧ Implement the following Boolean function using ROM (PROM). $F_1 = \sum m(1,2,3)$, $F_2 = \sum m(0,1,3)$

Sol:-



PROM Circuit

Programmable Logic Devices:

→ A programmable logic device (PLD) is an integrated circuit with internal logic gates that are connected through electronic fuses.

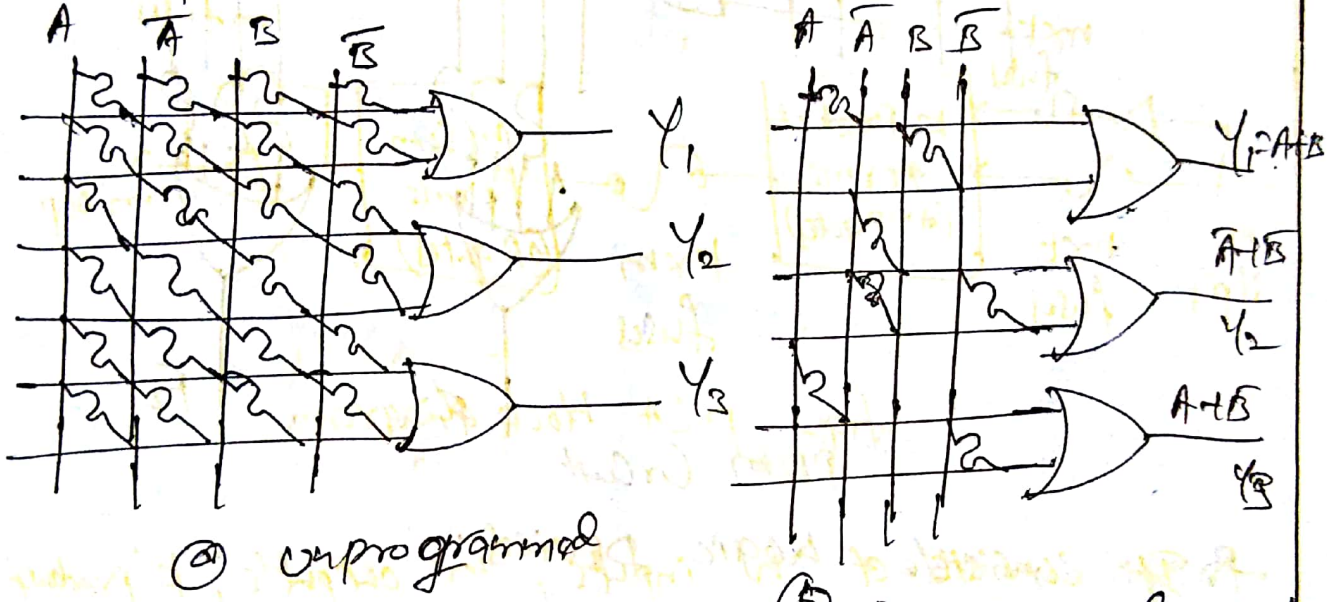
→ Programming the device involves the blowing of internal fuses to achieve a desired logic function.

→ All PLD consist of programmable arrays.

→ A programmable array is essentially a grid of conductors that form rows and columns with a fusible link at each cross point.

→ Arrays can be either fixed or programmable.

→ The gates in a PLD are divided into an AND array and OR array that are connected together to provide an AND-OR implementation.



(a) unprogrammed (b) programmed.

Classification of PLDs :-

→ The classification of PLDs are as follows.

1. programmable ROM
2. Programmable Logic Array (PLA)
3. Programmable Array Logic (PAL)
4. Generic Array Logic (GAL)
5. Field programmable Gate Array
6. Complex programmable Logic devices

Programmable Logic Array:-

→ PLA is used in logic design where the number of don't cares conditions is excessive, since it is more economical.

→ In the PLA the decoder (in ROM) is replaced by a group of AND gates, each of which can be programmed to generate a product term of the input variables.

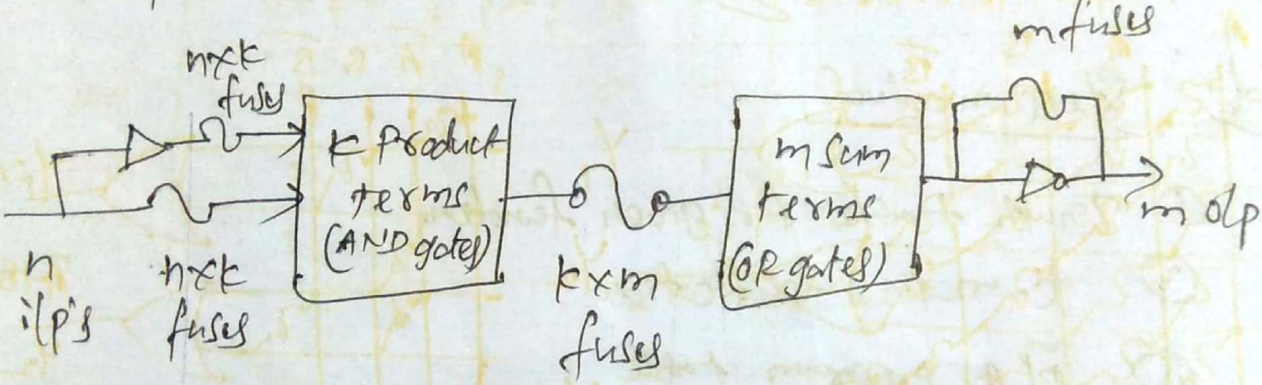


fig- PLA block diagram.

→ 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 constitutes a group of 'm' OR gates.

→ fuses are inserted between to all 'n' inputs and their complements values to each of the AND gates.

→ Fuses are also provide between the outputs of the AND gates and the inputs of OR gates.

→ Another set of fuses in the output inverters allow the output function to be generated either in the AND-OR form or in the AND-OR-INVERT form.

Ex① Implement the combinational circuit for the functions.

$$F_1 = \sum m(4,5,7), \quad F_2 = \sum m(3,5,7).$$

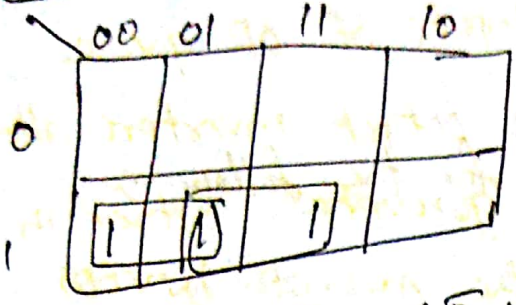
Sol: Steps involved

① Truth table for given function
② K-map simplification
③ PLA program table
④ PLA diagram.

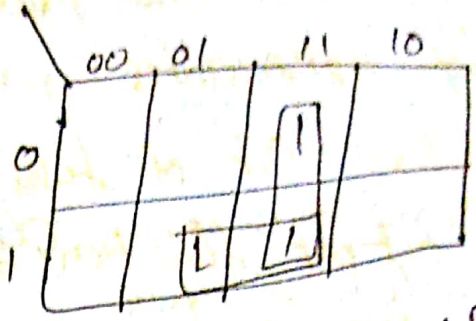
① Truth Table

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

K-map:



$F_1 = A\bar{B} + AC$

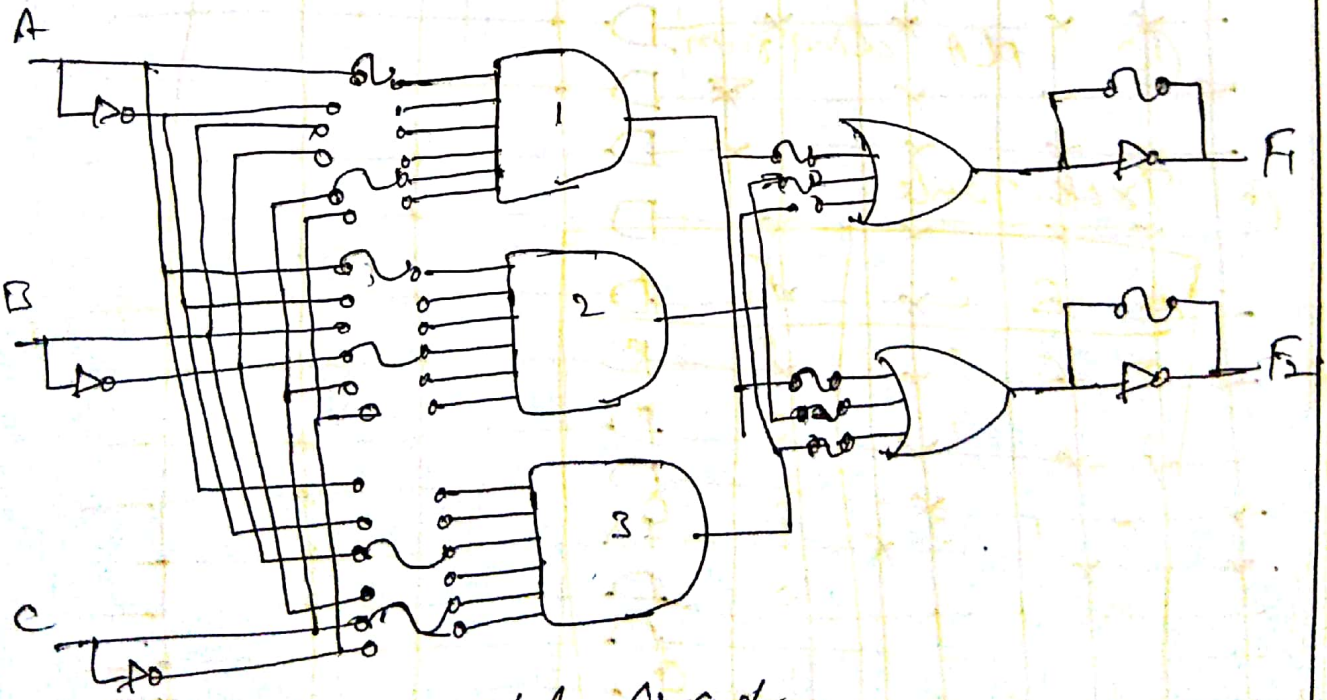


$F_2 = AC + BC$

(8) PLA Program Table:

	Product Term	Inputs A	B	C	Outputs F_1	F_2
$A\bar{B}$	1	1	0	–	1	–
AC	2	1	–	1	1	1
BC	3	–	1	1	–	1
					T	T

T/C



PLA Circuit

Implementation of Combinational Logic Circuit using PAL :-

① Draw the PAL diagram for the following Boolean function:

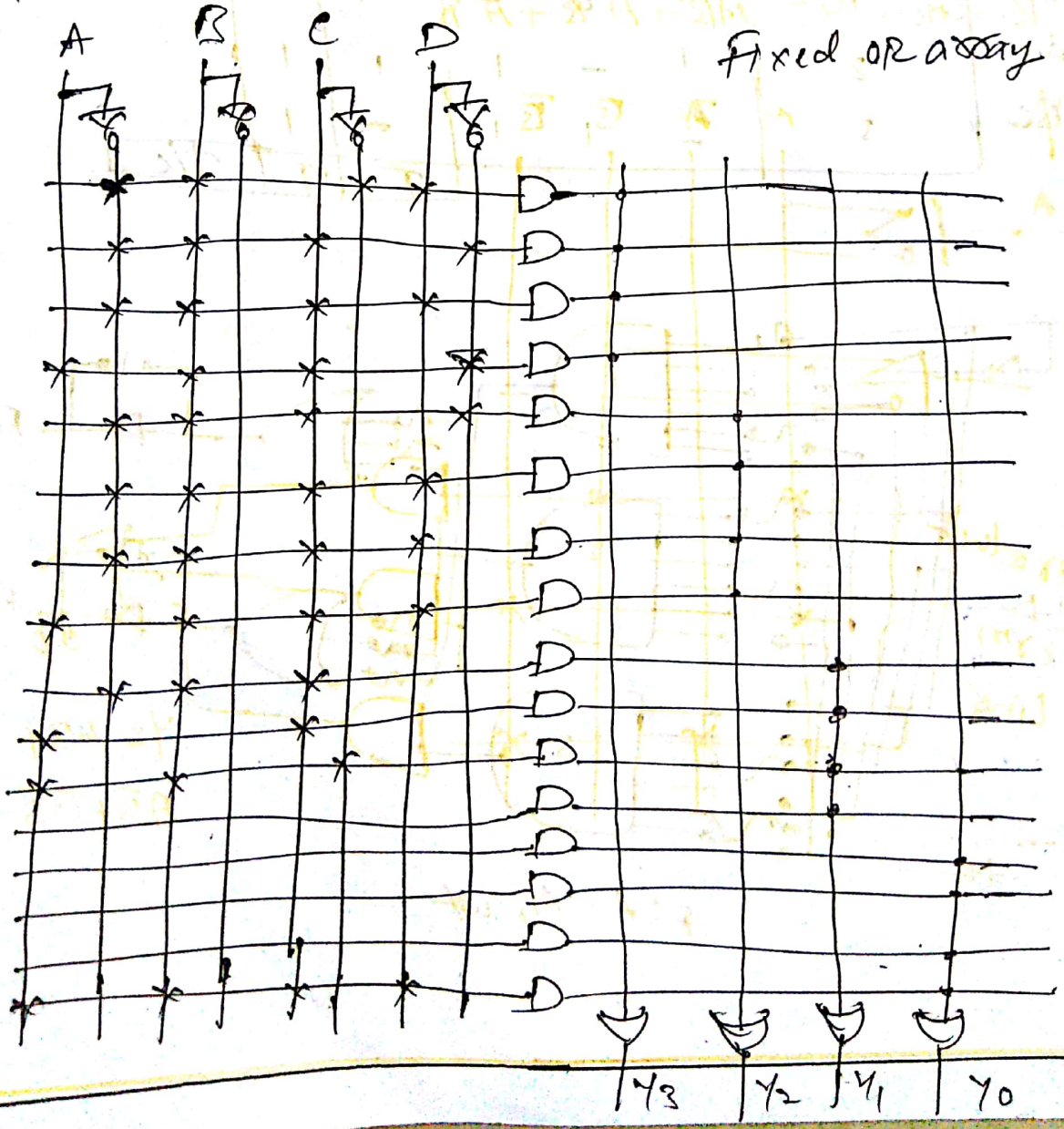
$Y_3 = \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}BCD + AB\bar{C}\bar{D}$

$Y_2 = \bar{A}B\bar{C}\bar{D} + \bar{A}BCD + ABCD$

$Y_1 = \bar{A}B\bar{C} + \bar{A}BC + AC + AB\bar{C}$

$Y_0 = ABCD$

Sol:-

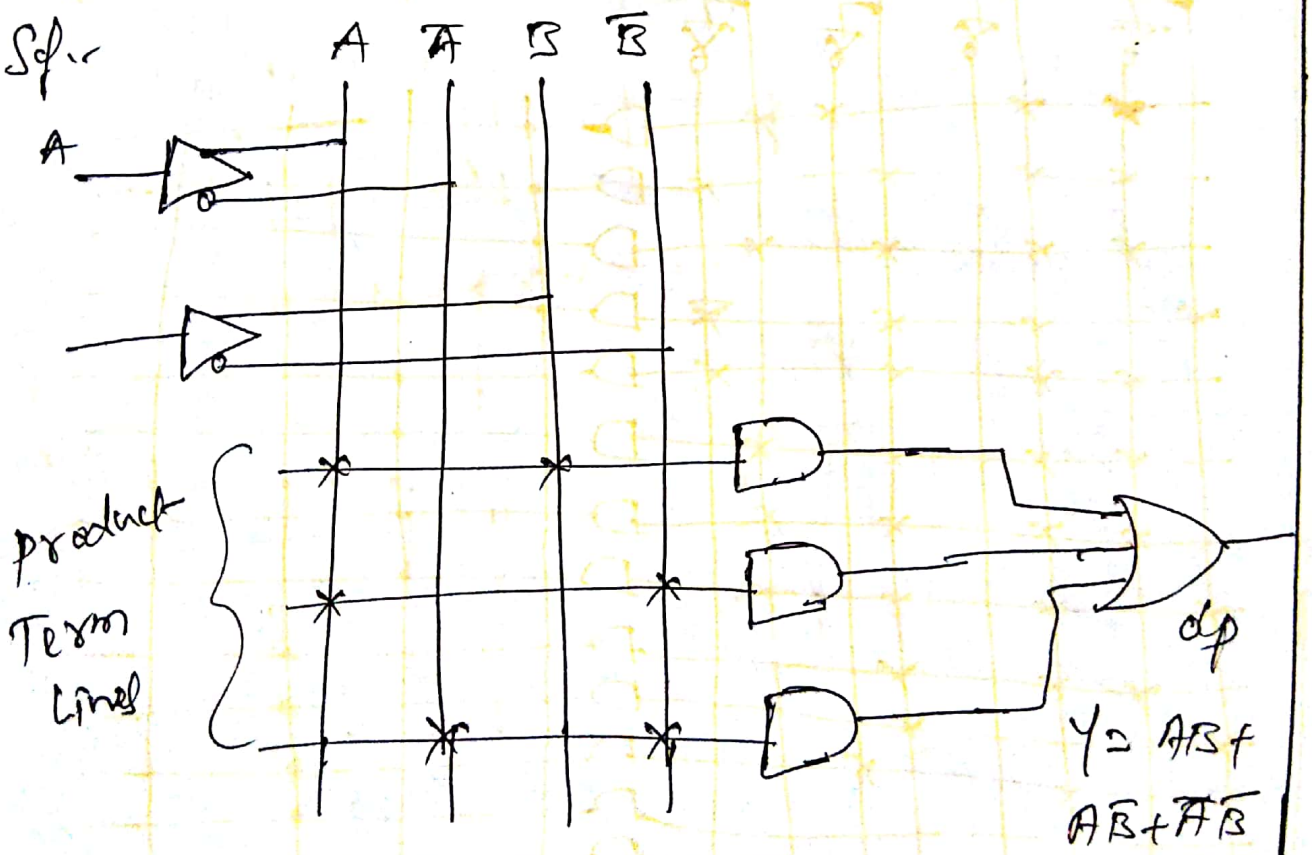


Programmable Array Logic (PAL):

⇒ The PAL consists of a programmable array of AND gates that connects to a fixed array of OR gates.

⇒ This structure allows any Sum-of-products (SOP) logic expressions with a defined number of variables to be implemented.

Ex① Implement the circuit with PAL for the function $Y = AB + A\bar{B} + \bar{A}\bar{B}$.



Ex② Implement the given function using PAL.

$A = \sum(0, 2, 6, 7, 8, 9, 12, 13)$

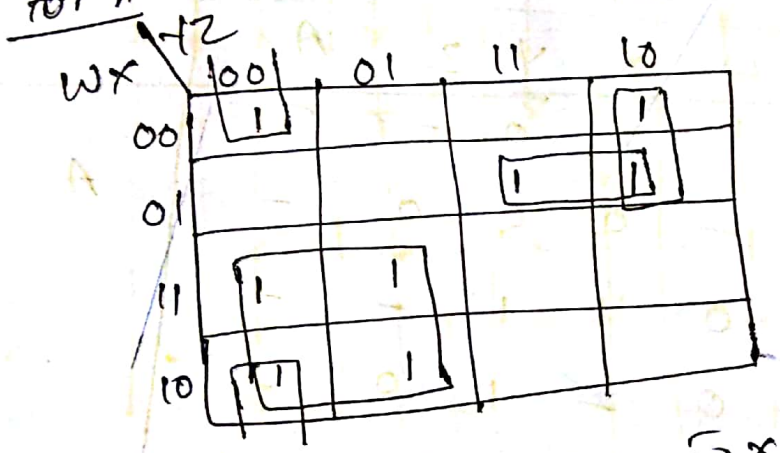
$B = \sum(0, 2, 6, 7, 8, 9, 12, 13, 14)$

$C = \sum(1, 3, 4, 6, 10, 12, 13)$

$D = \sum(1, 3, 4, 6, 9, 12, 14)$.

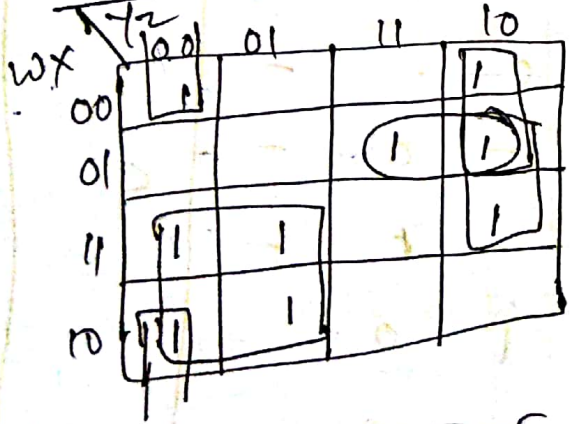
Sol: K-map Simplification:

for A



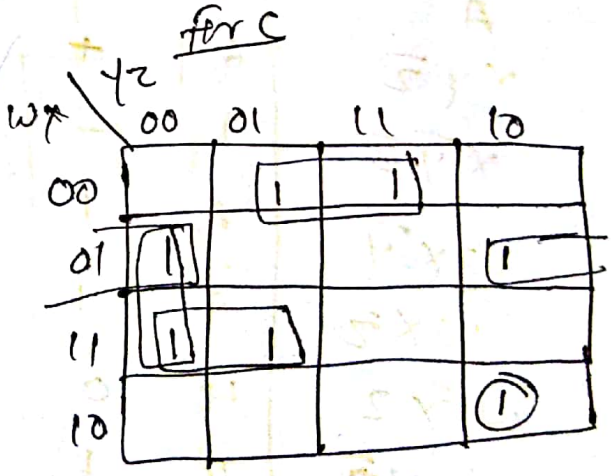
$A = W\bar{Y} + \bar{X}\bar{Y}\bar{Z} + \bar{W}XY + \bar{W}Y\bar{Z}$

for B:



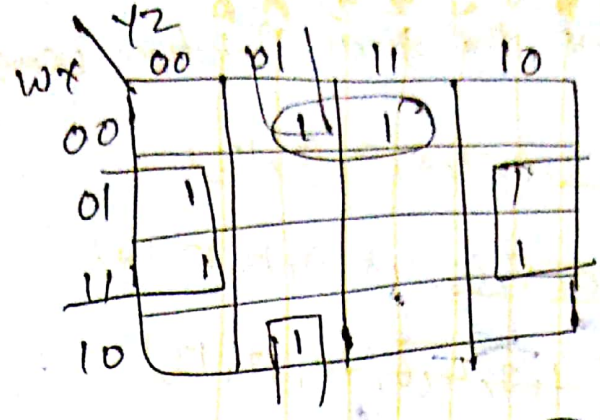
$B = W\bar{Y} + \bar{X}\bar{Y}\bar{Z} + \bar{W}XY + \bar{W}Y\bar{Z} + XY\bar{Z} = A + XY\bar{Z}$

for C



$C = \bar{X}Y\bar{Z} + WX\bar{Y} + \bar{W}X\bar{Z} + \bar{W}\bar{X}Z + W\bar{X}Y\bar{Z}$

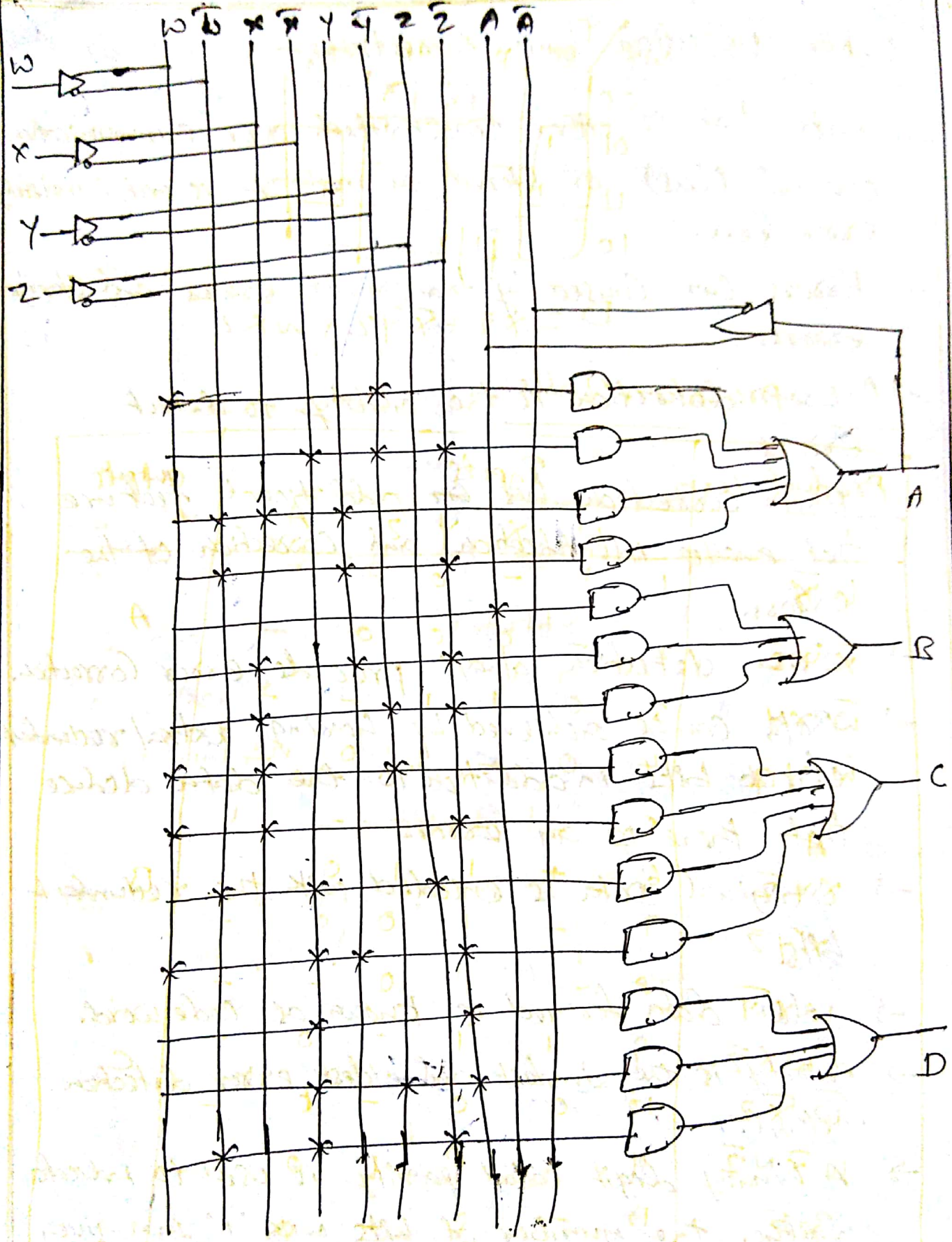
For D:



$$D = X\bar{Z} + \bar{X}\bar{Y}Z + \bar{W}\bar{X}Z$$

PAL program Table:

Product Term	Inputs						Outputs
		W	X	Y	Z	A	
$W\bar{X}$	1	1	–	0	–	–	A
$\bar{X}\bar{Y}\bar{Z}$	2	–	0	0	0	–	
$\bar{W}XY$	3	0	1	1	–	–	
$\bar{W}Y\bar{Z}$	4	0	–	1	0	–	
A	5	–	–	–	–	1	
$XY\bar{Z}$	6	1	1	1	0	–	B
$X\bar{Y}Z$	7	–	1	0	1	–	
$WX\bar{Y}$	8	1	1	0	–	–	
WXZ	9	1	1	–	1	–	C
$\bar{W}\bar{X}Z$	10	0	0	–	1	–	
$W\bar{X}Y\bar{Z}$	11	1	0	1	0	–	
$\bar{X}\bar{Z}$	12	–	0	–	0	–	D
$\bar{X}\bar{Y}\bar{Z}$	13	–	0	0	0	–	
$\bar{W}\bar{X}Z$	14	0	0	–	1	–	



Error Detection and Correction:-

→ Data that is either transmitted over communication channel (bus) or stored in memory is not completely error free.

→ Errors can caused by transmission errors and storage errors.

→ Error detection is the ability to detect errors.

→ Error correction has an additional feature that enables identification and correction of the errors.

→ Error detection always precede error correction.

→ Both can be achieved by having extra/redundant check bits in addition to ~~the~~ data deduce that there is an error.

→ Original Data is encoded with the redundant bits.

→ New data formed is known as code word.

→ Parity is the simplest and oldest error detection method.

→ A binary digit called 'parity' is used to indicate whether the number of bits with '1' in a given set of bits is even or odd.

→ The parity bit is then appended to original data.

→ Sender adds the parity bit to existing data bits before transmission.

→ Receiver checks for the expected parity.

Hamming Code :-

→ Hamming code is used for error detection and error correction.

→ This code uses a number of parity bits (dependent on the number of information bits), located at certain positions in the code group.

→ If the number of information bit is designated 'm', then the number of parity bits 'p' is determinated as,

$$2^p \geq m+p+1.$$

→ for example, if we have 4 information bits, then p is found by trial and error method.

let $p=2$, then $2^p = 2^2 = 4$ and $m+p+1 = 4+2+1 = 7$

Equation not satisfied.

let $p=3$, then $2^p = 2^3 = 8$ and $m+p+1 = 4+3+1 = 8$

Equation satisfied.

3- parity bits (P_1, P_2, P_3) are located in the positions that are numbered corresponding to ascending powers of two (1, 2, 4, 8....)

bit 1, bit 2, bit 3, bit 4, bit 5, bit 6, bit 7

P_1 P_2 M_1 P_3 M_2 M_3 M_4

Ex①: Determine the single-error correcting code for the BCD number 1001 (information bits) using even parity.

Sol:- **Step 1:** Find the number of parity bits required.

Let $P = 3$, then $2^P = 2^3 = 8$

information bits, $M = 4$

$M + P + 1 = 4 + 3 + 1 = 8$

The equation $2^P \geq M + P + 1$ is satisfied.

Total code bits $= M + P + 1 = 4 + 3 = 7$.

Step 2:- Construct a bit position table, and enter the information bits,

Bit designation	P_1	P_2	M_1	P_3	M_2	M_3	M_4
Bit position	1	2	3	4	5	6	7
Binary position number	001	010	011	100	101	110	111
Information bits			1		0	0	1
Parity bits							

Step 3:- parity bits are determined as follows.

For P_1: Bit P_1, checks the bit positions 3, 5 and 7. They have two 1's and therefore to have an even parity P_1 must be 0.

For P_2: Bit P_2 checks the bit positions 3, 6 and 7. They have two 1's and therefore P_2 is '0'

for P_3: Bit P_3 checks the bit positions 5, 6 and 7. They have one 1's and P_3 is '1'

Step 4: These parity bits are entered in the table, and the resulting combined code is 0011001.

Bit designation	P_1	P_2	M_1	P_3	M_2	M_3	M_4
Bit position	1	2	3	4	5	6	7
Binary position number	001	010	011	100	101	110	111
Information bits			1		0	0	1
Parity bits	0	0		1			

Result: 0 0 1 1 0 0 1

Detecting and Correcting the error.

Ex① Assume that the code word (0011001) is transmitted and that 0010001 is received.

Sol:

Bit Designation	P_1	P_2	m_1	P_3	m_2	m_3	m_4
Bit position	1	2	3	4	5	6	7
Binary position number	001	010	011	100	101	110	111
Received code	0	0	1	0	0	0	1

For P_1: Bit P_1 checks positions 1, 3, 5 and 7.
There are two 1's in this group.
Parity check is good. ⟶ 0 (LSB)

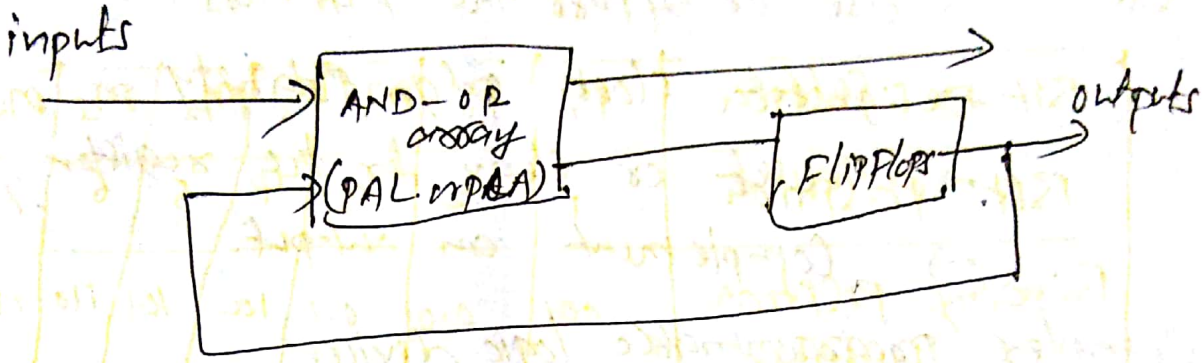
For P_2: P_2 checks positions 2, 3, 6 and 7
There are two 1's in this group
Parity check is good ⟶ 0

For P_3: Bit P_3 checks positions 4, 5, 6 and 7.
There are one 1 in this group.
parity check is bad. ⟶ 1 (MSB)

The error position code = 100 i.e. 4
This says that the bit in position 4 is in error.
It is a '0' and should be a '1'.
correct code is 0011001.

Sequential programmable Devices:-

→ The device can be programmed to perform a variety of sequential circuit functions.



Types:

1. Sequential (or Simple) programmable logic device (SPLD)
2. Complex programmable logic device (CPLD)
3. Field programmable gate array (FPGA)

① Sequential programmable logic devices:

→ SPLD includes flip-flops and AND-OR array.

→ flip-flops outputs could be included in product terms of AND array.

→ Field programmable logic sequencer (FPLS).

→ combinational PAL together with D flipflops are mostly used.

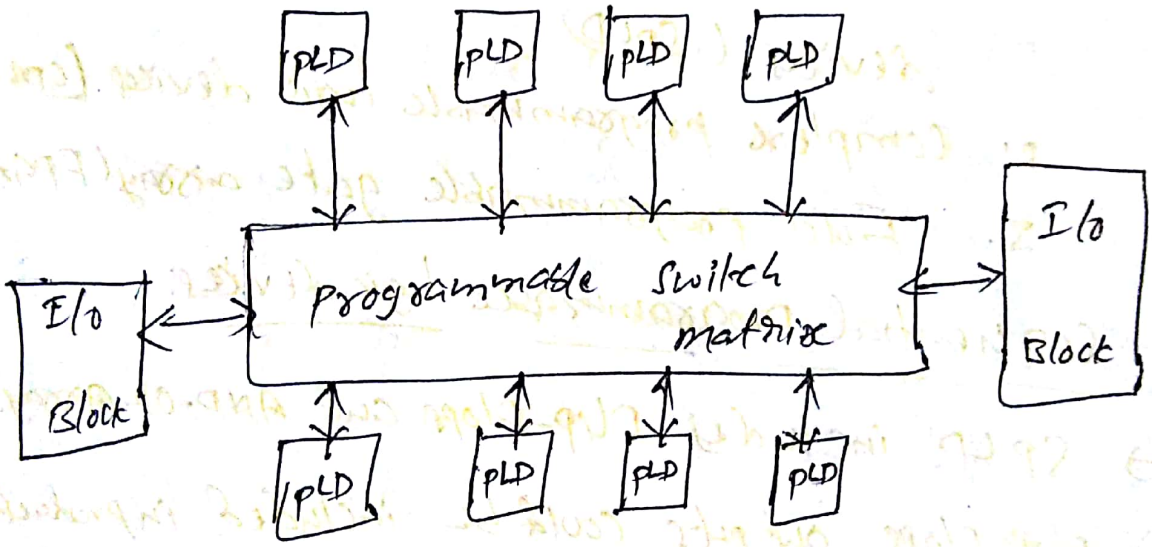
→ Each section of an SPLD is called macrocell, which consists of SOP and an optional flipflop.

→ A typical SPLD contains 8-10 macrocells with one IC package.

features:

- → programming AND array
- → use or bypass the FLIPFLOPS
- → select clock edge polarity
- → preset or clear for the register
- → complement an output.

Complex programmable logic devices:



→ CPLD is a collection of PLD's to be connected to each other through a programmable switch matrix

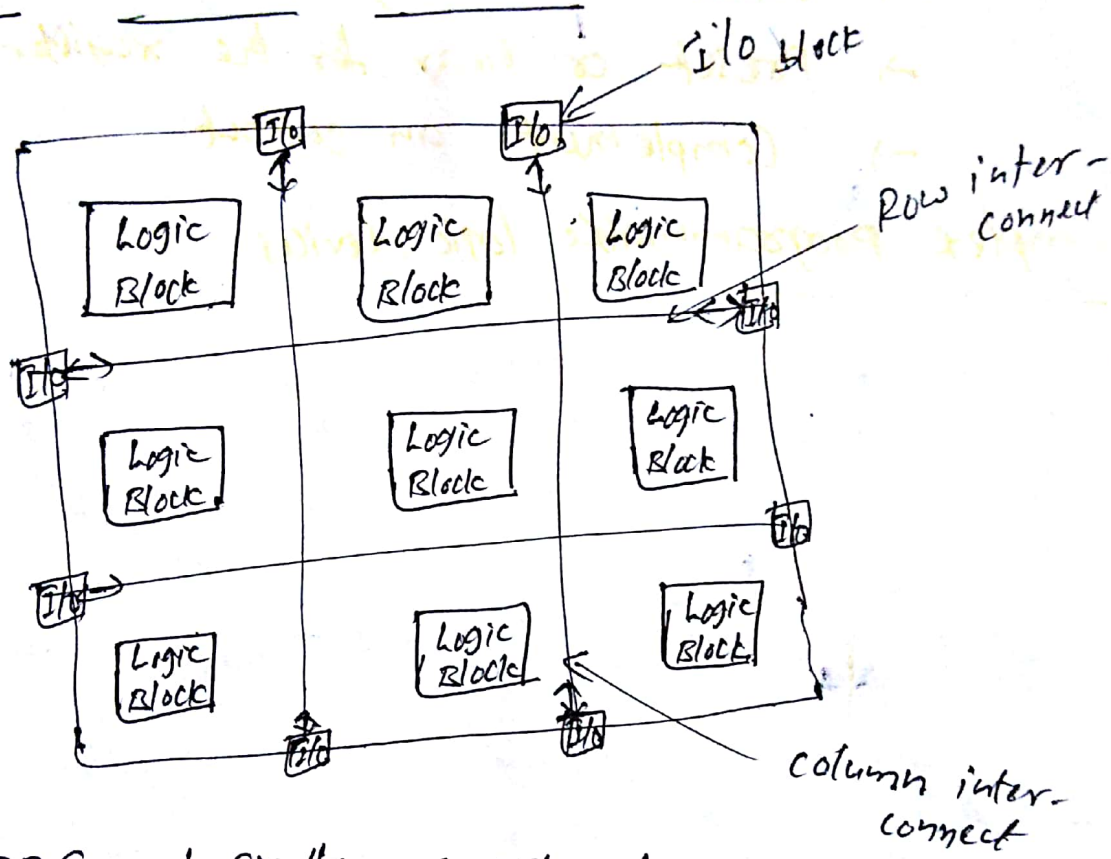
→ Input/output blocks provide connections to IC pins

→ Each I/o pin is driven by a three-state buffer and can be programmed to act as input or output.

→ Switch matrix receives inputs from I/O block and directs it to individual macrocells.

→ Each PLD typically contains from 8 to 16 macrocells.

Field programmable Gate Array (FPGA):



→ FPGA basically consists of an array of logic blocks with programmable row and column interconnecting channels surrounded by programmable I/o blocks.

→ many FPGA architectures are based on a type of memory called LUT (Look-up Table) rather than on AND/OR arrays.

→ Another approach is the use of multiplexers to generate logic functions.

→ FPGA is a Single VLSI (Very large Scale Integrated) Circuit constructed on a Single piece of Silicon.

prepared By

Verified By

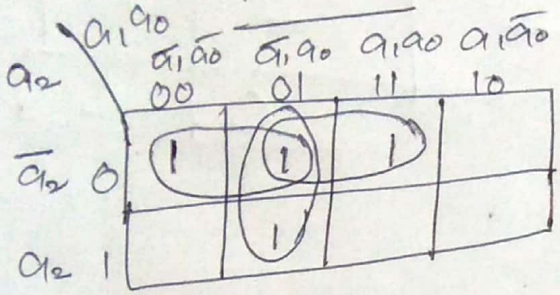
Ex① Implement the following multi boolean function using 3x4x2 PLA PLD.

$$f_1(a_2, a_1, a_0) = \sum m(0,1,3,5)$$

$$f_2(a_2, a_1, a_0) = \sum m(3,5,7)$$

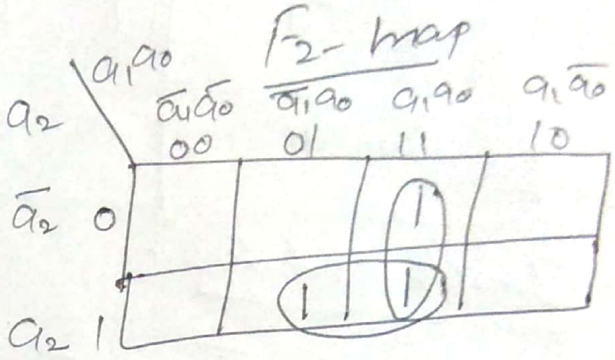
Sol:-

F_1 - map



$F_1 = \bar{a}_2\bar{a}_1 + \bar{a}_2 a_0 + \bar{a}_1 a_0$

F_2 - map

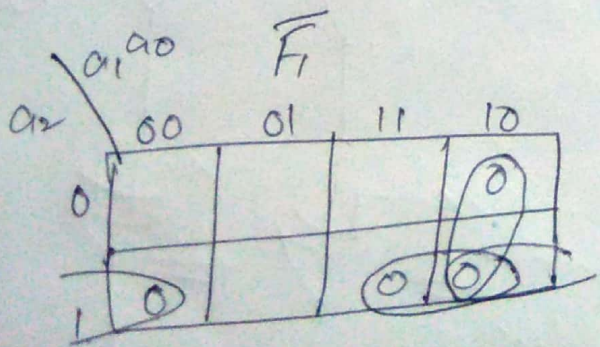


$F_2 = a_2 a_0 + a_1 a_0$

→ To implement functions f_1 and F_2 we require 3x5x2 PLA and we have to implement them using 3x4x2 PLA.

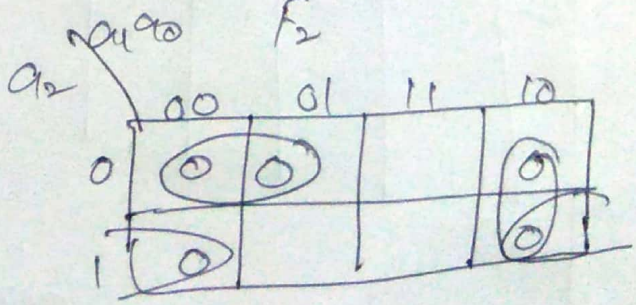
→ Therefore, we have examine product terms by grouping 0's instead of 1's. That is product terms for complement of a function.

$\bar{F}_1$



$\bar{F}_1 = a_2\bar{a}_0 + a_1\bar{a}_0 + a_2 a_1$

$\bar{F}_2$



$\bar{F}_2 = a_2\bar{a}_1 + a_1\bar{a}_0 + a_2\bar{a}_0$

PLA Table :

Product terms	inputs			outputs	
	a_2	a_1	a_0	f_1	F_2
$a_2\,\bar{a}_0$	1	–	0	1	1
$a_1\,\bar{a}_0$	–	1	0	1	1
$a_2\,a_1$	1	1	–	1	–
$\bar{a}_2\,\bar{a}_1$	0	0	–	–	1
				C	C

Implementation :