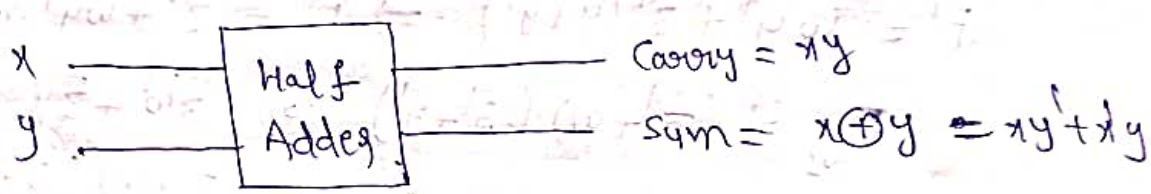
Unit-II:- Combinational Circuits.

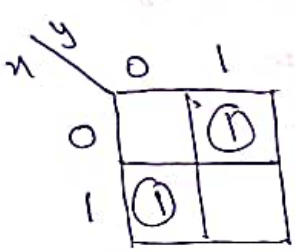
Combinational Logic Circuits: Adders & Subtractors, Multiplexers, Demultiplexers, Encoders, Decoders, Programmable Logic Devices.

Adders:-

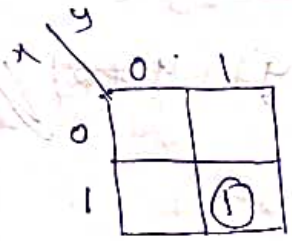
Half Adder:-

A combinational circuit that performs addition of two bits is called a half adder.





Sum = $xy' + x'y$

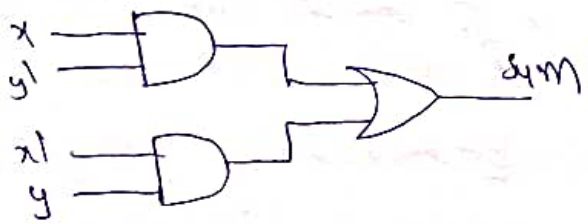


Carry = xy

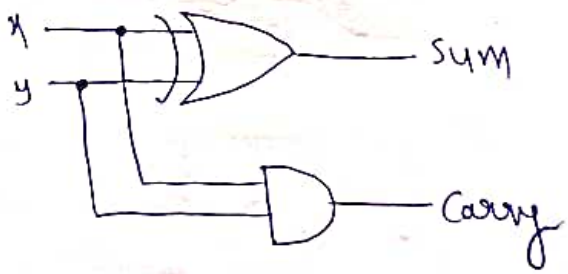
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

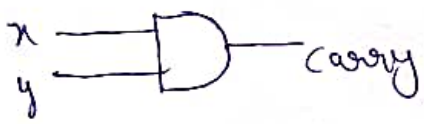
Implementation of half adder:-

$S = x \oplus y = xy' + x'y$ $\quad C = xy$



(or)

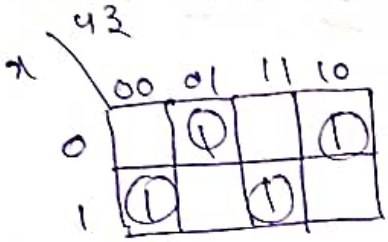




Full Adder:-

A combinational circuit that performs the addition of three bits (2-significant bits and previous carry) is called a full adder.

Two half adders can be employed to implement full adder.



	x	y	z	Carry	Sum
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

$Sum = x'y'z + x'yz' + xy'z' + xyz$

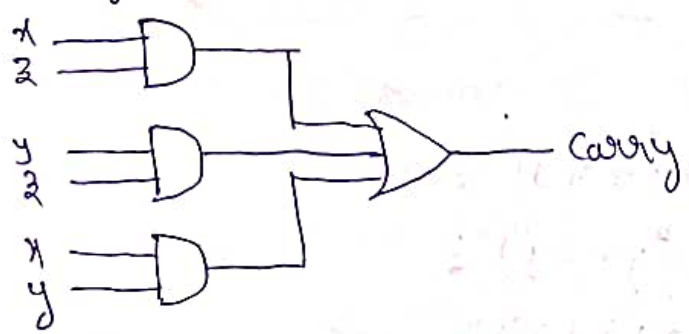
$Carry = xz + yz + xy$

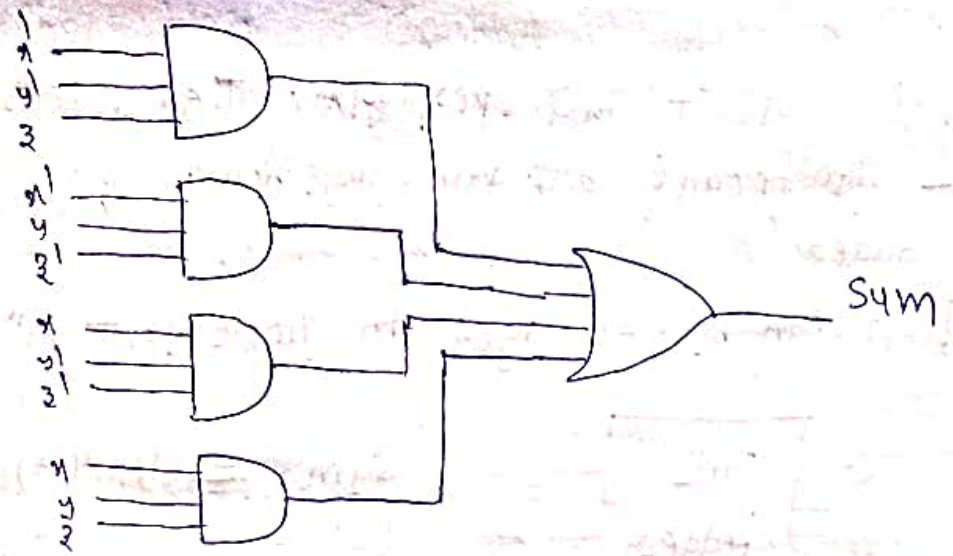
$\begin{array}{r} 101 \\ 111 \\ \hline xz \end{array}$ $\begin{array}{r} 111 \\ 011 \\ \hline yz \end{array}$ $\begin{array}{r} 111 \\ 110 \\ \hline xy \end{array}$

Implementation of full adder:-

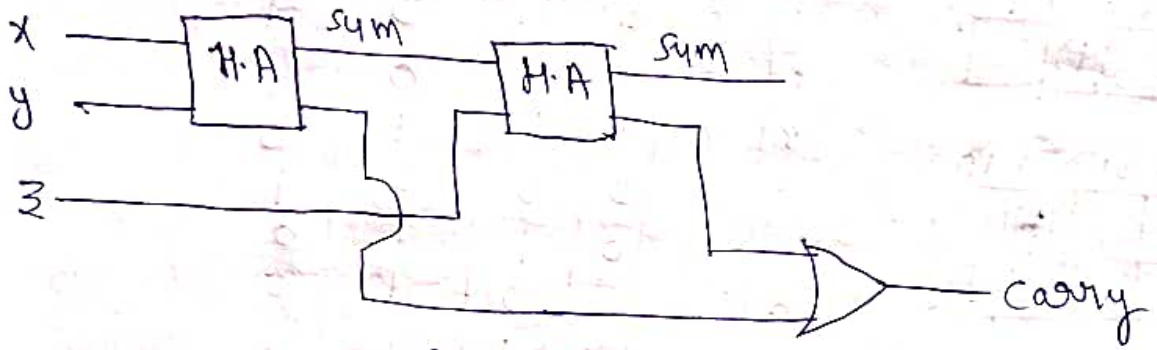
$Sum = x'y'z + x'yz' + xy'z' + xyz$

$Carry = xz + yz + xy$

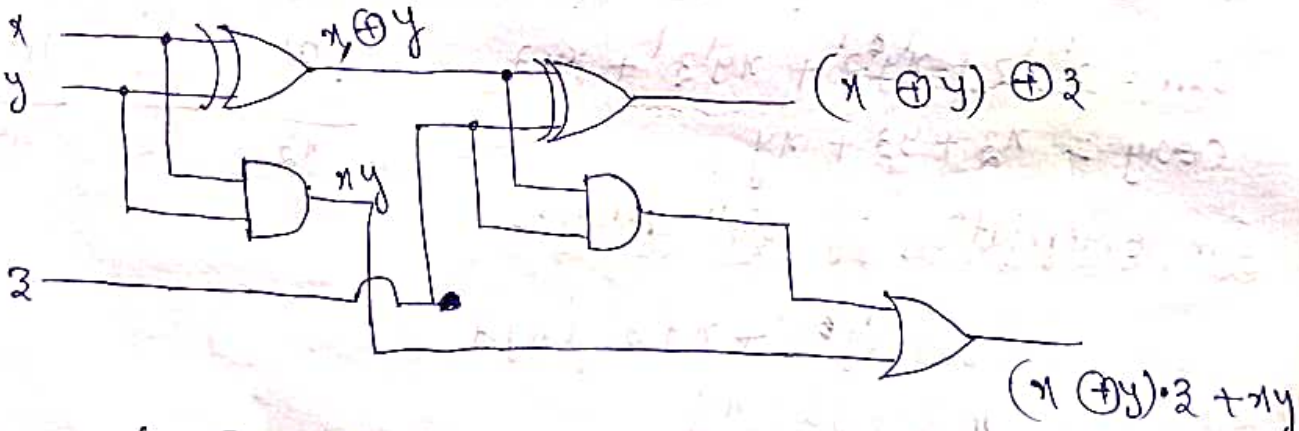




Implementation of full adder using Two half adders:-

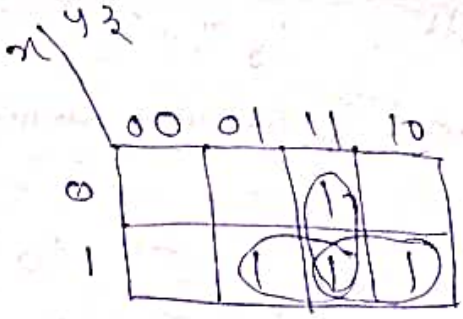


(or)

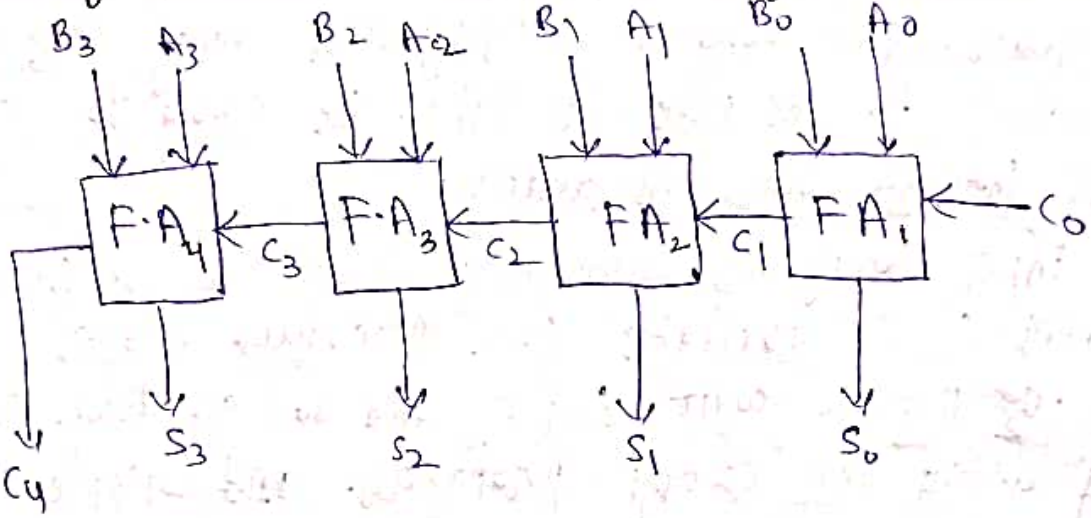


$$\text{Sum} = (x \oplus y) \oplus z$$
$$= (xy' + x'y) \oplus z$$
$$= (xy' + x'y)z' + (xy' + x'y)' \cdot z$$
$$= xy'z' + x'yz' + (x'y + xy')z$$
$$\therefore \text{Sum} = xy'z' + x'yz' + x'y'z + xyz$$

$$\text{Carry} = (x \oplus y)\cdot z + xy$$
$$= (xy' + x'y)z + xy$$
$$= xy'z + x'yz + xy$$
$$= xy'z + x'yz + xy(z + z')$$
$$= xy'z + x'yz + xyz + xyz'$$
$$\text{Carry} = xz + yz + xy$$



Binary Adder (or) Ripple Carry Adder (or) 4-bit adder:-



Subscript	3	2	1	0	
i/p carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
o/p carry	0	0	1	1	C_{i+1}

The above figure shows the inter connection of 4- full adder circuits to provide 4-bit binary ripple carry adder. The carries are connected in a chain through the full adders. The input carry

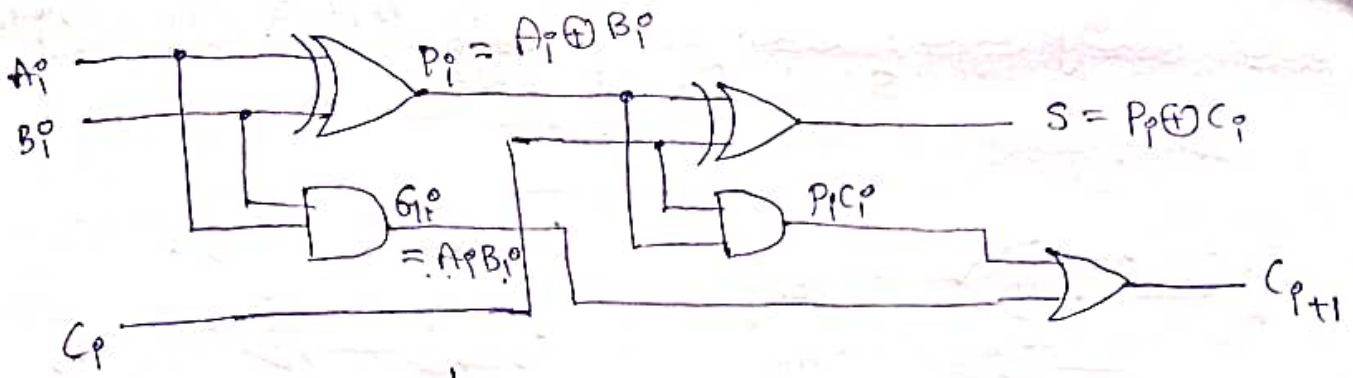
to the adder is C_0 and it ripples through the full adder to the O/p carry C_4.
The 'S' outputs generates the required sum bits. An n-bit adder requires n-full adders with each o/p carry connected to the input carry of next higher adder full adder.

Carry Propagation:-

The total propagation time equal to the propagation delay of typical gate times the number of gate levels in the circuit. The longest propagation delay time in an adder is the time it takes the carry to propagate through the full adder.
Let the input carry C_3 does not settle to its final value until C_2 is available from previous stage. Similarly, C_2 has to wait for C_1 and so on down to C_0. Thus, only after the carry propagates and ripples through all stages will the last output S_3 and carry C_4, settle to their final correct value.
So the carry propagation time is an important attribute of the adder because it limits the speed with which two numbers are added.

Carry look ahead logic / Generator:-

There are several techniques for reducing the carry propagation time in a parallel adder. The most widely used technique employs the principle of carry look ahead logic.



$P_i = A_i \oplus B_i$ | $S = P_i \oplus C_i$

$G_i = A_i B_i$ | $C_{i+1} = G_i + P_i C_i$

put $i=0$, $C_1 = G_0 + P_0 C_0$

put $i = 1$, $C_2 = G_1 + P_1 C_1$

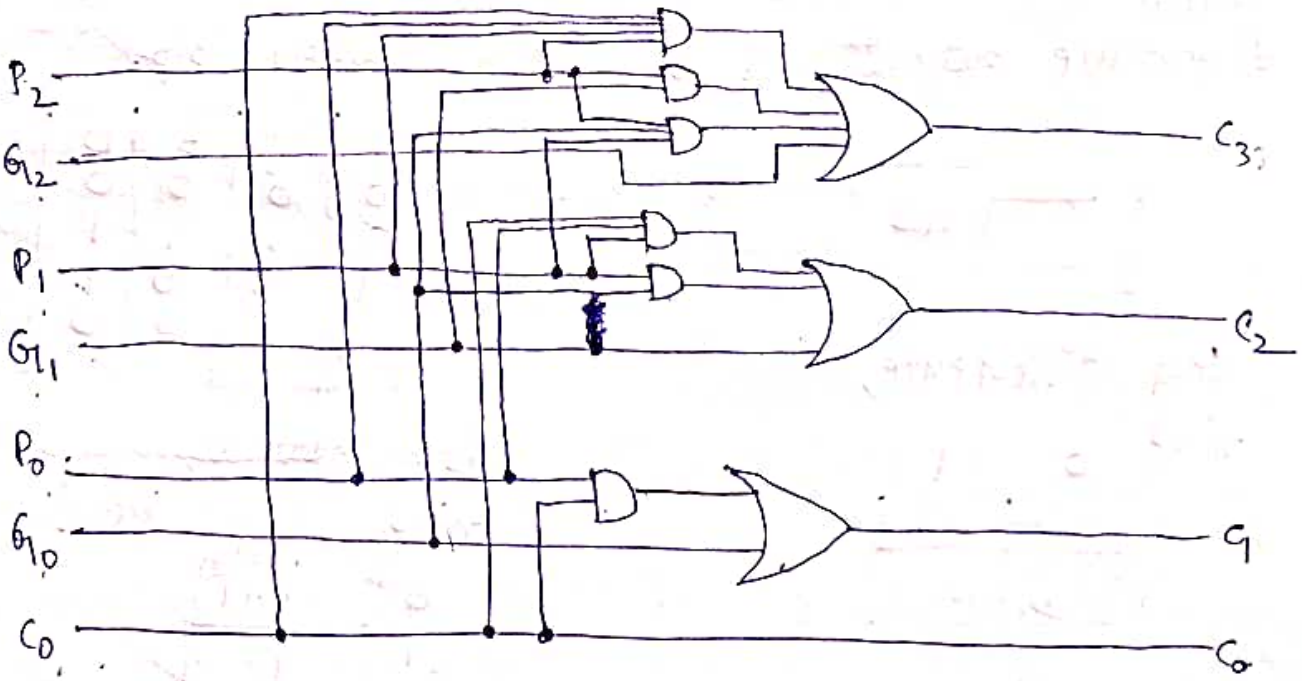
$= G_1 + P_1 (G_0 + P_0 C_0)$

$= G_1 + P_1 G_0 + P_1 P_0 C_0$

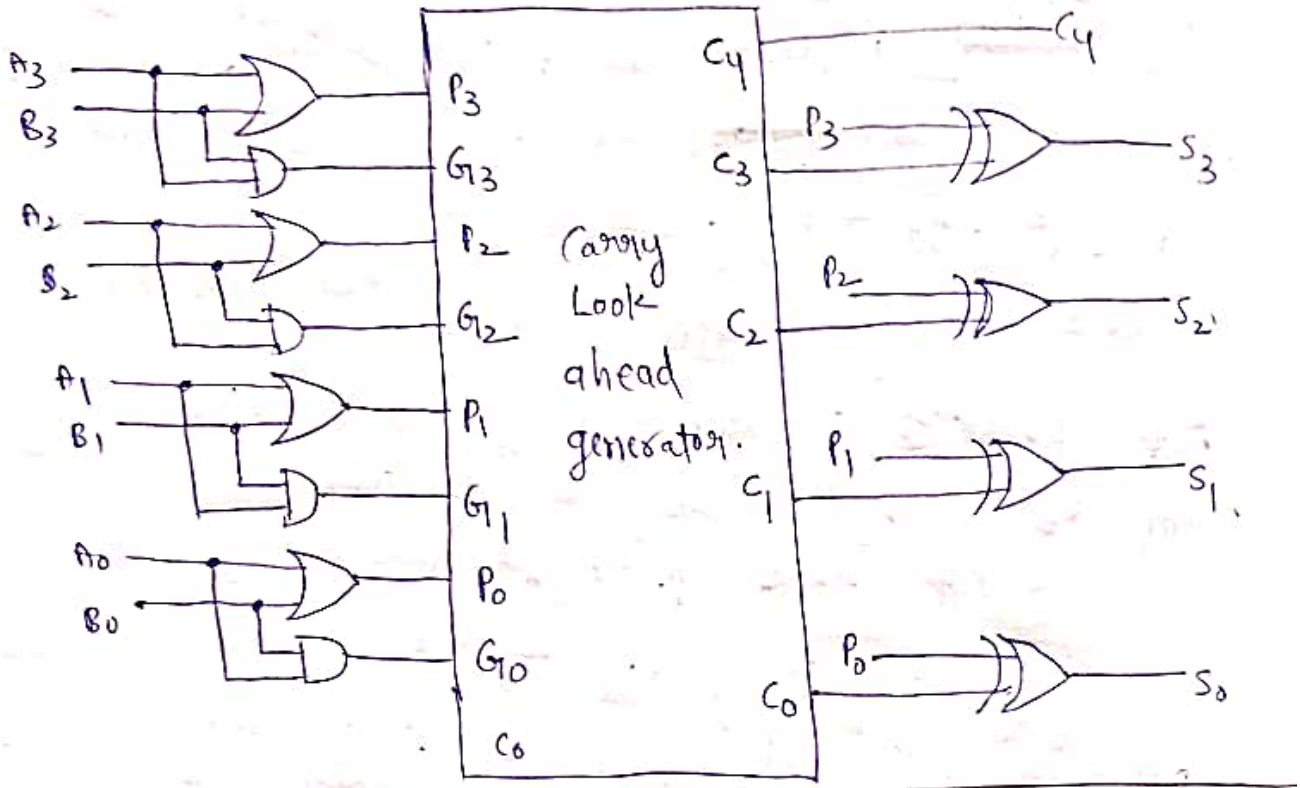
put $i=2$, $C_3 = G_2 + P_2 C_2$

$= G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_0)$

$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$



4-bit binary adder with carry look ahead generator (or) Speed Binary Adder:-

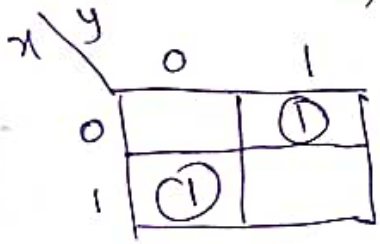


Half Subtractor:-

The two inputs for half subtractor are x and y from the minuend and the subtrahend, D is the difference output and B is the borrow o/p.

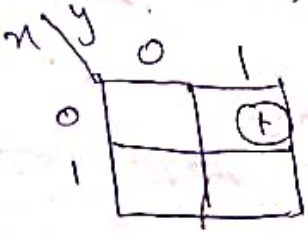
x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

for difference,

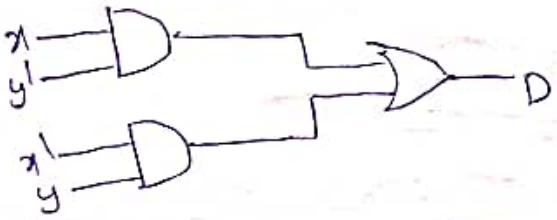


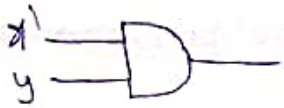
$D = xy' + x'y = x \oplus y$

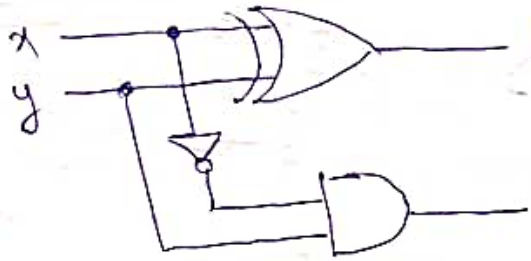
for borrow,



$B = x'y$



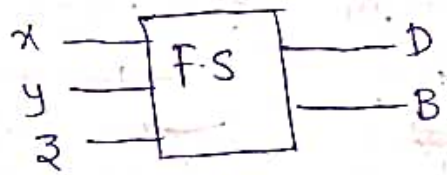




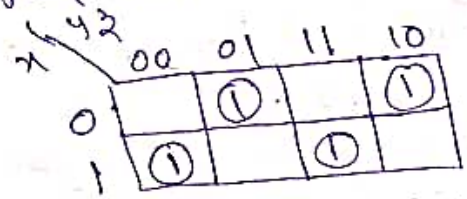
Full Subtractor:-

A full subtractor has three inputs and two outputs x, y and z are the inputs to be subtracted in which z represents barrow from the next stage. D and B are the outputs

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

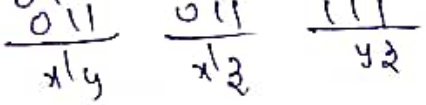


for difference

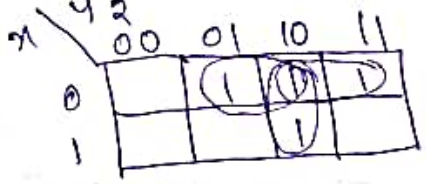


$D = xy'z' + x'yz' + x'y'z + xyz = x \oplus y \oplus z$

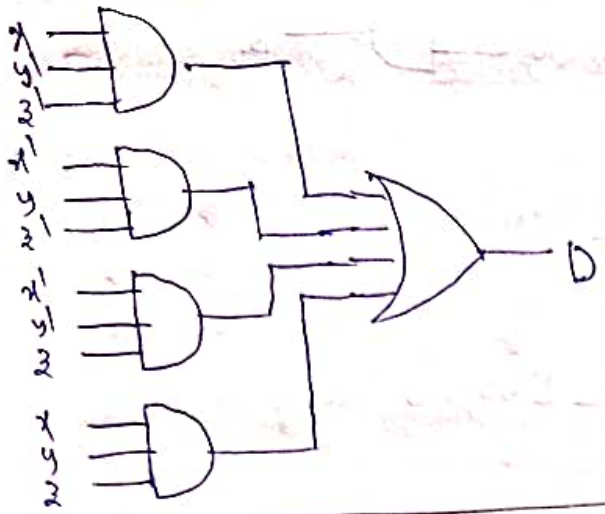
$D = \Sigma(1,2,4,7)$.

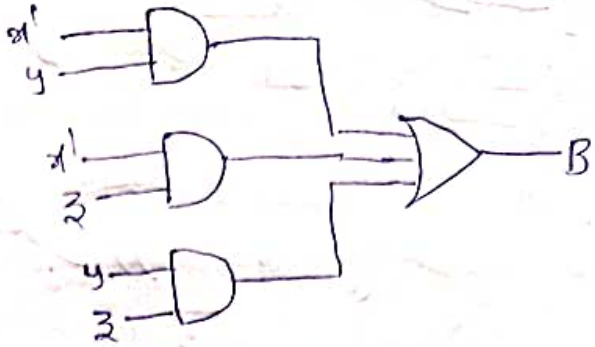


for barrow.



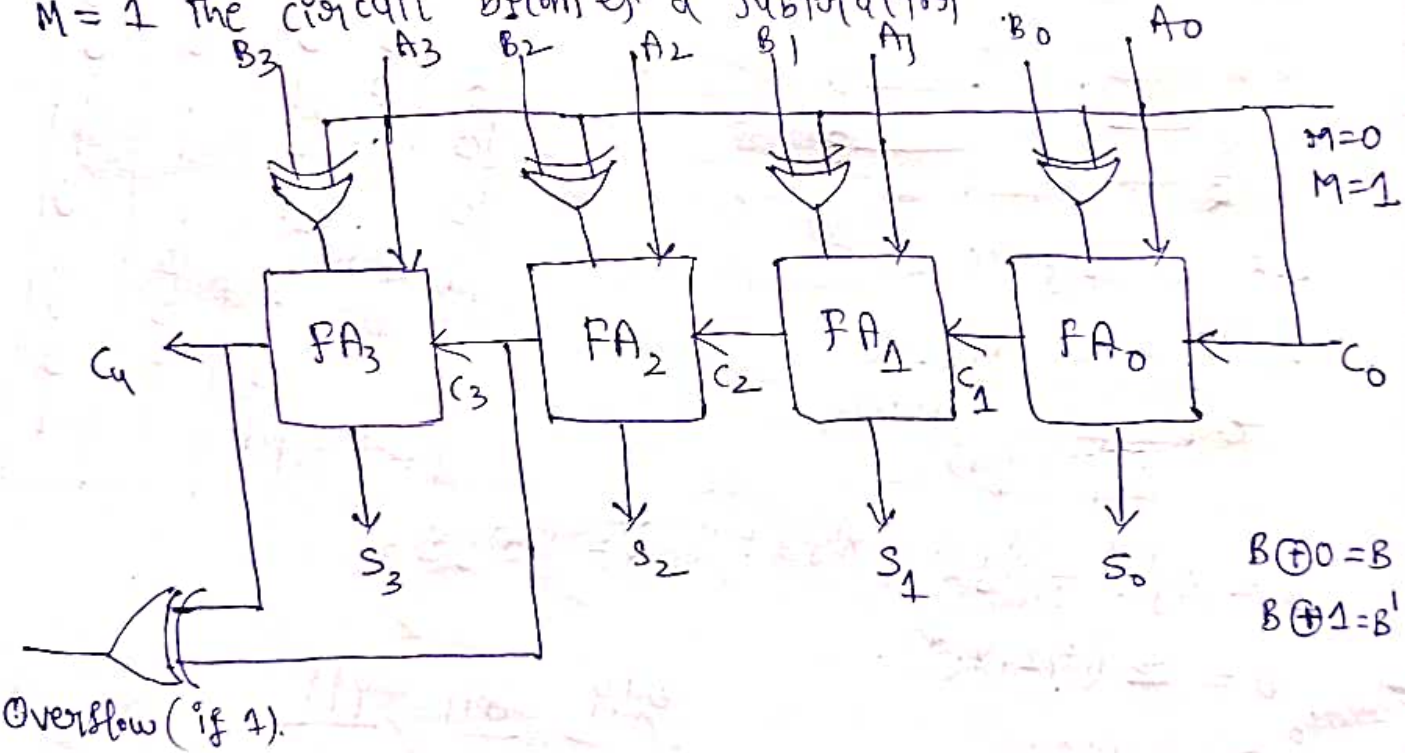
$B = x'y + x'z + yz$





Binary subtractor or Adder / subtractor Circuit:-

The addition and subtraction operations can be combined into one circuit. In this mode input 'M' controls the operation, when M=0 the circuit is an adder, when M=1 the circuit becomes a subtractor



Overflow :-

When two numbers with 'n' digits each are added sum is a number occupying 'n+1' digits we say that an overflow occured.

Overflow is a problem in digital computers because the number of bits that hold the number is finite and a result that contains 'n+1' bits can not be accomodated by an 'n' bit word, when overflow occurs a corresponding flip flop is set in the above binary adders subtractors that can then be checked by the user.

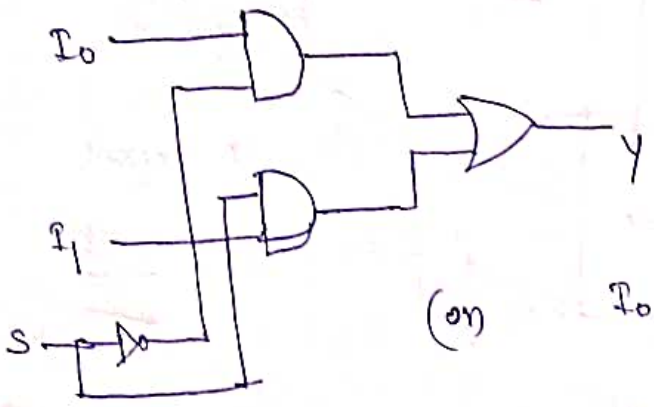
Multiplexer (MUX):-

(i) A multiplexer is a combinational circuit that selects binary information from one of many I/p lines and directs it to a single o/p line.

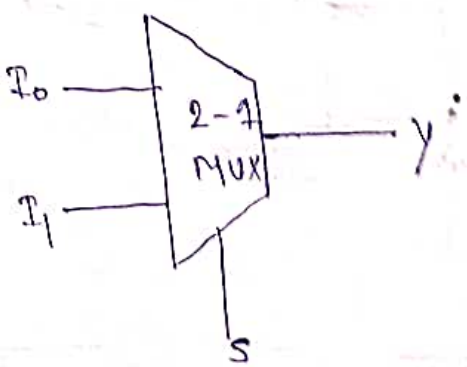
(ii) The selection of a particular I/p line is controlled by set of selection lines.

(iii) Normally there are 2^n input lines and 'n' selection lines whose bit combination determine which I/p is selected.

2-1 MUX:-



(or)



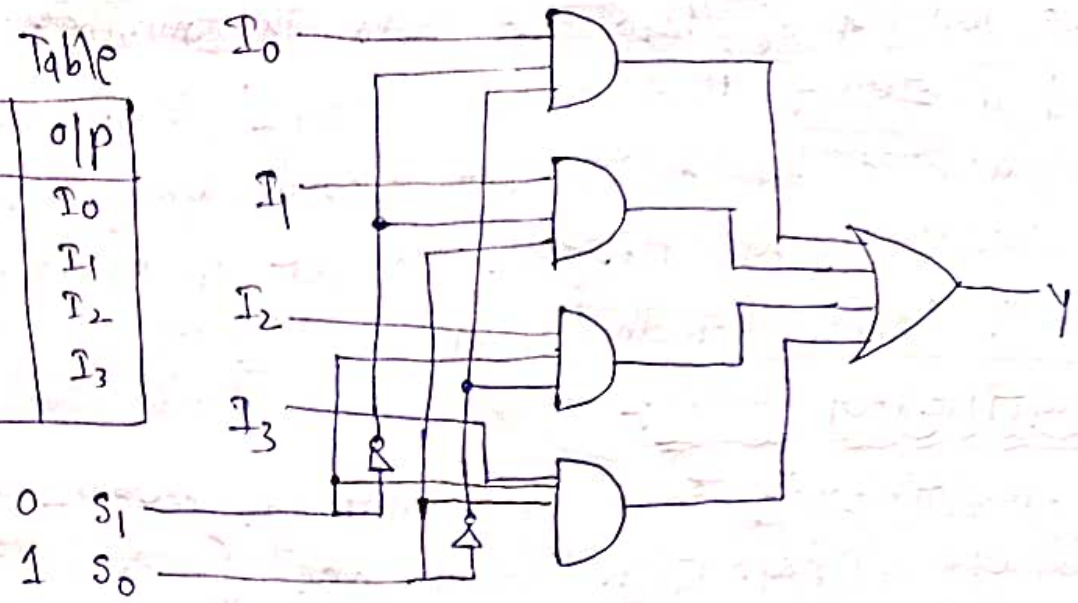
Function Table

S	Y
0	I_0
1	I_1

4-1 MUX :-

Function Table

S_1	S_0	o/p
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

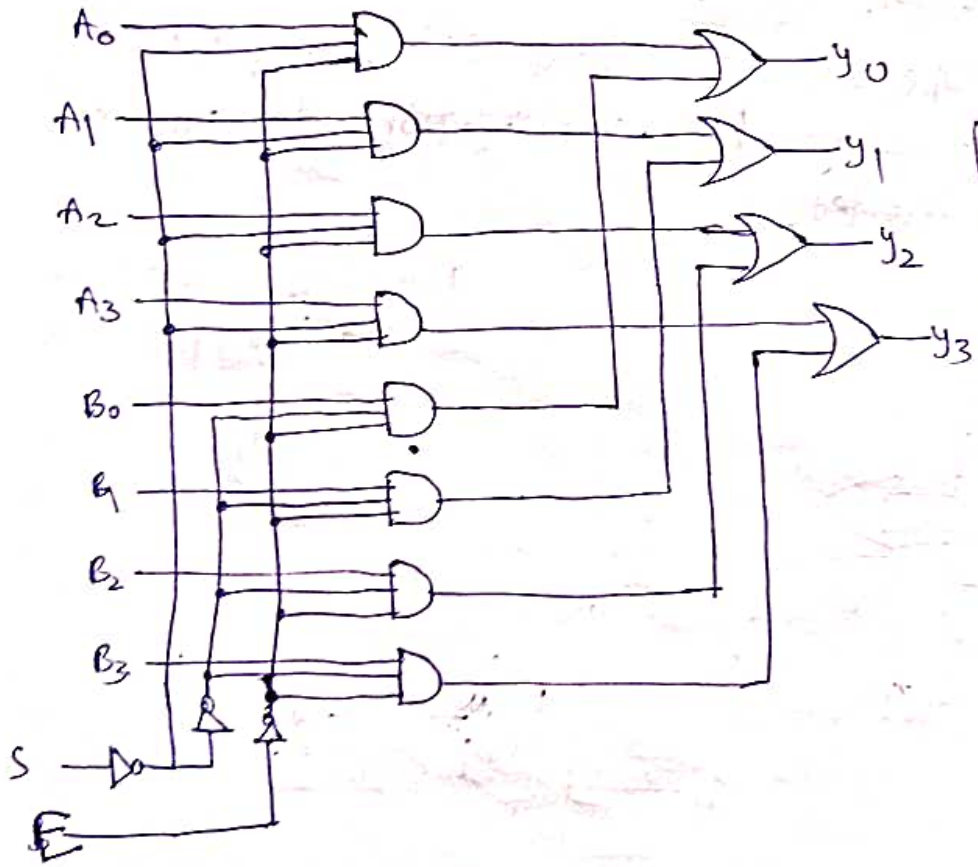


Quadraple 2-1 MUX :-

$2^n = 2^1$

$n = 1$

ϵ	S	o/p
1	X	0
0	0	A
0	1	B



Boolean function Implementation:-

Boolean function of 'n' variables with a multiplexer that has 'n' selection inputs and 2^n data inputs '1' for each min term. For implementing a boolean function of 'n' variables with a multiplexer that has 'n-1' selection inputs and 2^{n-1} data inputs.

The first 'n-1' variables of the function are connected to the inputs of the multiplexer. The remaining single variable of the function is used for the data inputs if the single variable is denoted by 'z', each data input of the multiplexer will be z, $\bar{z}$, 1 (or) 0.

① Implement the boolean function

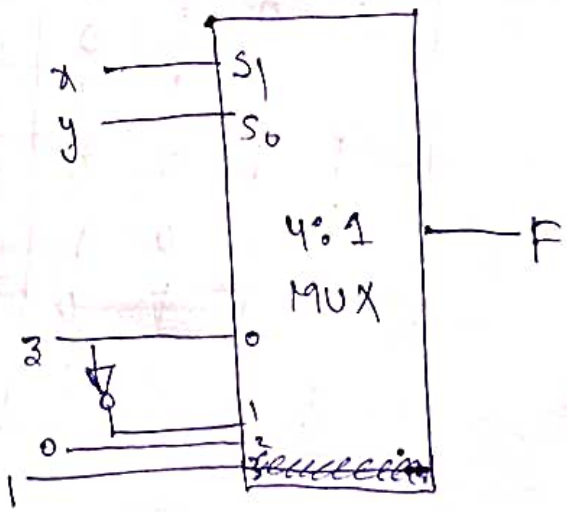
$F(x,y,z) = \Sigma(1,2,6,7)$ with a multiplexer.

Sol:- Here, $n=3$. = no. of variable (3 variables).

$2^{n-1} = 2^{3-1} = 2^2 = 4$ inputs.

$n-1 = 3-1 = 2$ selection lines.

S_1 x	S_0 y	z	F	
0	0	0	0	F=z
0	0	1	1	
0	1	0	1	F=z'
0	1	1	0	
1	0	0	0	F=0
1	0	1	0	
1	1	0	1	F=1
1	1	1	1	



② Implement the boolean function

$F(A,B,C,D) = \Sigma(1,3,4,11,12,13,14,15)$. with a multiplexer.

Sol^{n} Here, $n = 4$

$2^{n-1} = 2^{4-1} = 2^3 = 8$ inputs

$n-1 = 4-1 = 3$ selection lines.

S_2 A	S_1 B	S_0 C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

$F = D$

$F = D$

$F = D'$

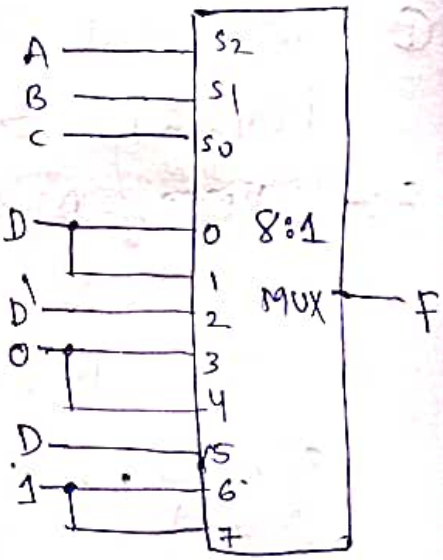
$F = 0$

$F = 0$

$F = D$

$F = 1$

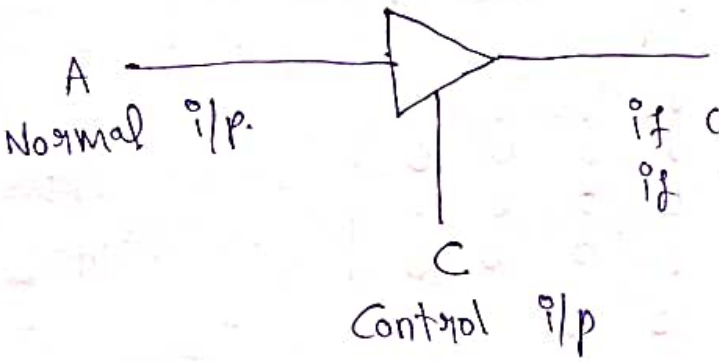
$F = 1$



Three State Gates :-

A multiplexer can be constructed with three state gates. Two of the states are signals equalent to logic-1 and logic-'0' in a conventional gate.

The third state is high impedence state.



if C = 1 then A
if C = 0 then high impedence.

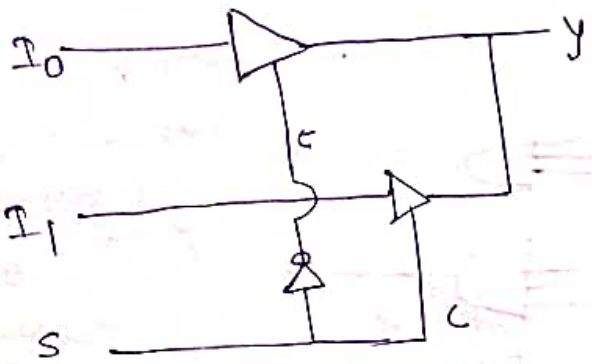
① Draw 2:1 multiplexer with 3-state gate.

Sol :- n = inputs = 2

$2^n = 2 = 2^1$

$n = 1$ selection line

Selection line S_0	o/p Y
0	I_0
1	I_1



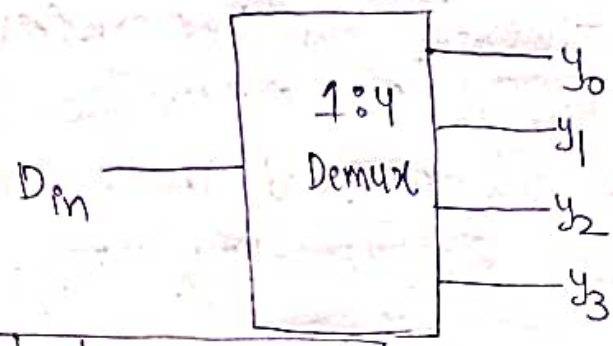
Demultiplexer :-

A demultiplexer is a circuit that receives information on single line and transmits this on one of the 2^n output lines. The selection of specific output lines is controlled by the value of 'n' selection lines.

① Draw the 1:4 de-multiplexer.

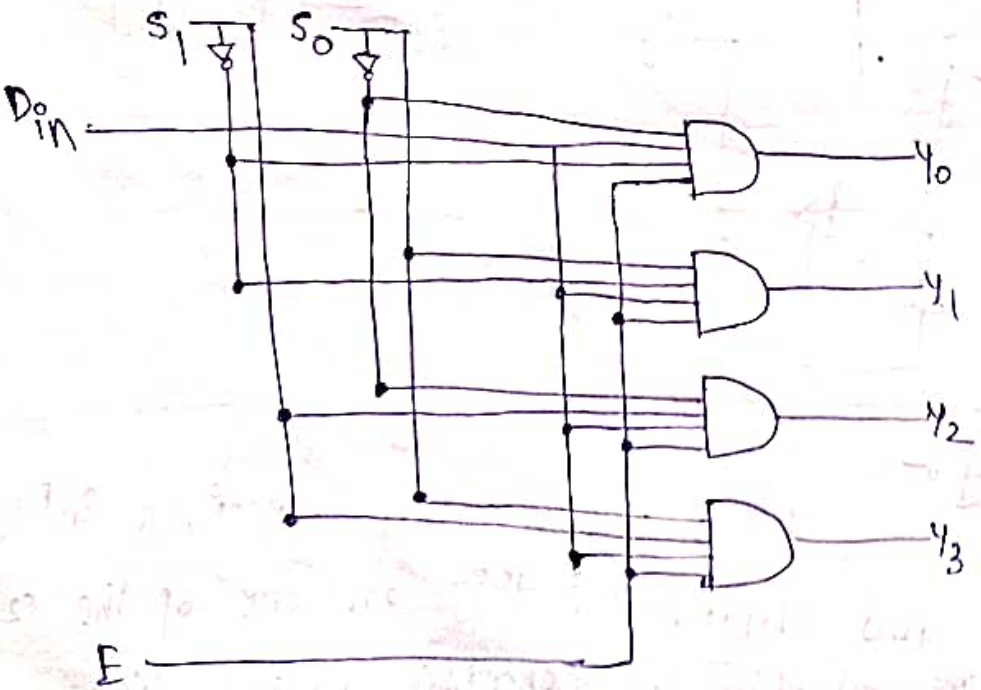
Sol:- $2^n = 4 = 2^2$

$n = 2$



Function Table

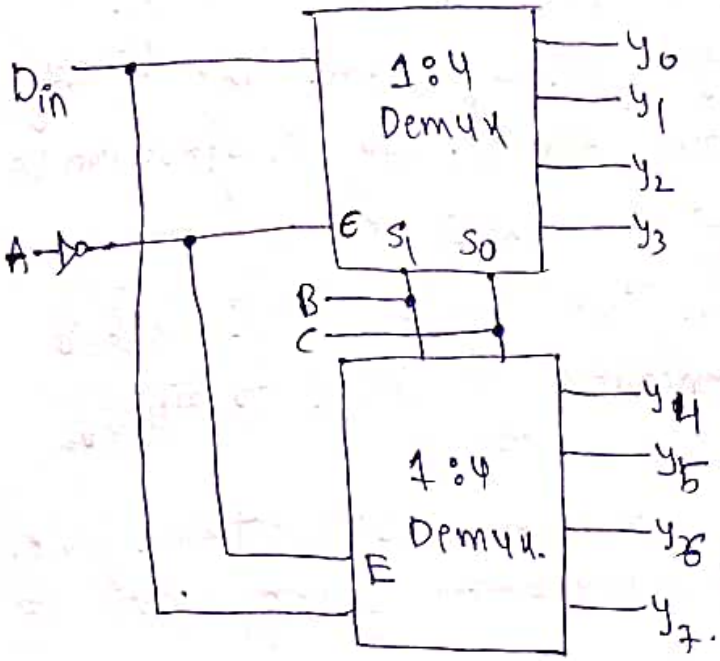
Enable	selection		i/p	o/p.			
E	S_1	S_0	D_{in}	y_0	y_1	y_2	y_3
0	X	X	X	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	1



② Design 1:8 demux using two 1:4 demux.

Sol:- $2^n = 8 = 2^3$

$n = 3$



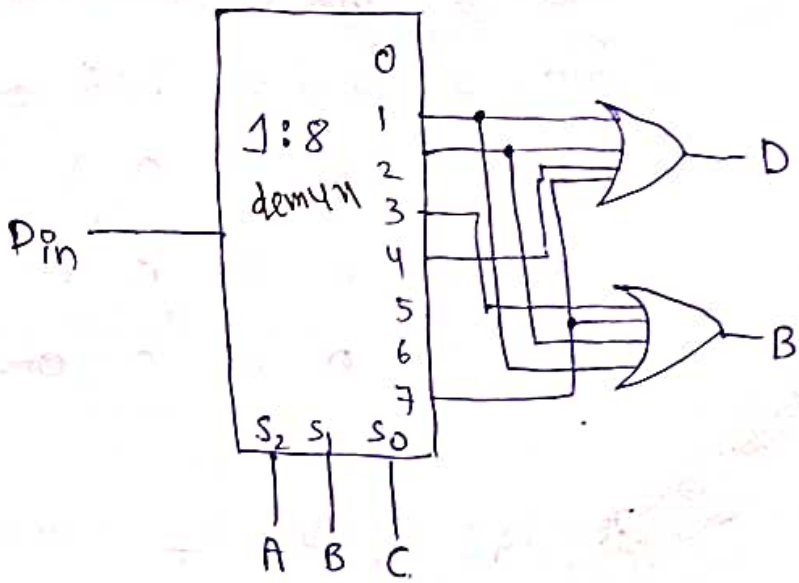
	S_1	S_0	D_{in}
A	B	C	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

③ Implement full subtractor using demux

Sol:- Difference = $D = \Sigma(1,2,4,7)$

Borrow = $B = \Sigma(1,2,3,7)$

A	B	C	Borrow	Diff
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



Encoders :-

An encoder is digital circuit that performs the inverse operation of a decoder. An encoder has 2^n input lines and 'n' output lines (fewer). The o/p lines are in aggregate generate the binary code corresponding to input lines. the encoder can be implemented with OR gates, whose inputs are directly determined from the truth table. One of the example for this encoder is Octal to Binary encoder.

① Draw Octal to Binary encoder with truth table.

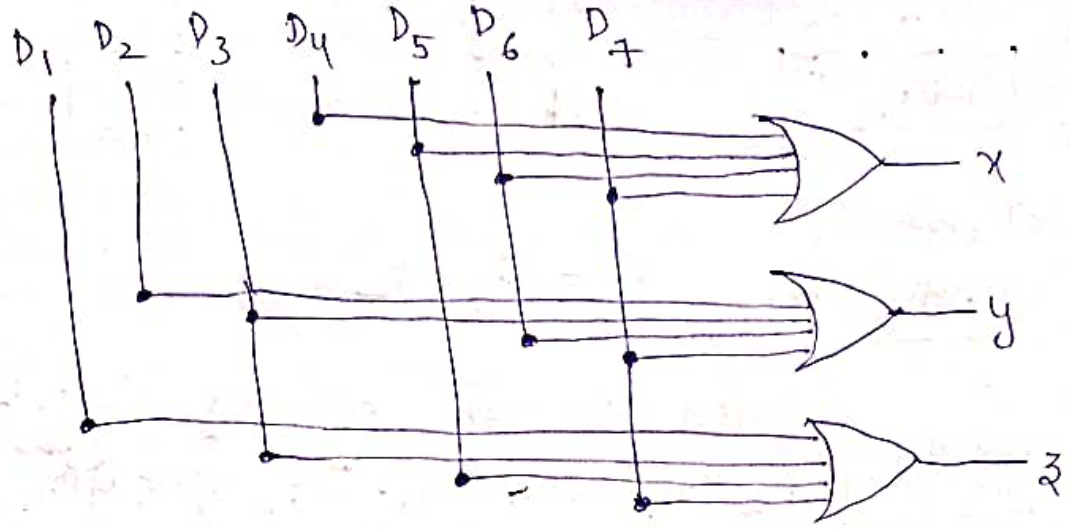
Sol:-

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$x = D_4 + D_5 + D_6 + D_7$

$y = D_2 + D_3 + D_6 + D_7$

$z = D_1 + D_3 + D_5 + D_7$.



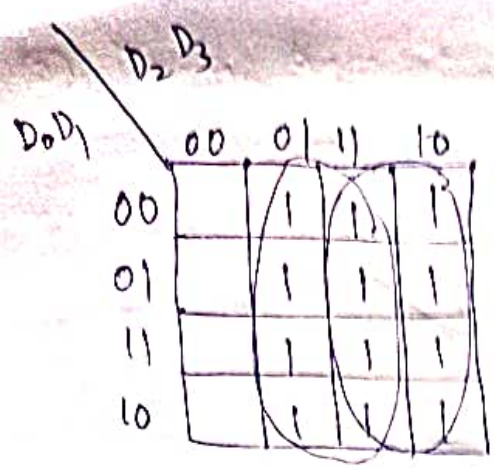
Ambiguities (or Limitations in Octal to Binary encoder:-

(i) In this, if two inputs are active (i.e, equal to 1). Simultaneously, the o/p produce an undefined combination.

(ii) If o/p with all zeros is generated when all the inputs are zero, but this output is seen as when D_0 is equal to 1. This can be resolved by one (or) more o/ps to indicate whether at least one input is equal to 1.
i.e, validate output.

Priority Encoder:-

D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

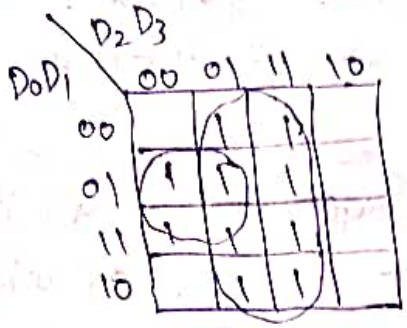


0010 — 2
0110 — 6
1010 — 10
1110 — 14

0001 — 1
0011 — 3
0101 — 5
0111 — 7
1001 — 9
1011 — 11
1101 — 13
1111 — 15

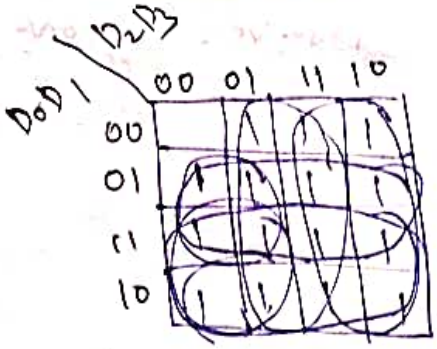
$x = D_3 + D_2$.

A priority encoder is an encoder circuit that includes priority function. The operation of the priority encoder is such that if two or more inputs equal to 1 at the same time the input having highest priority will take precedence.



0100 — 4
1100 — 12

$y = D_3 + D_1\bar{D}_2$

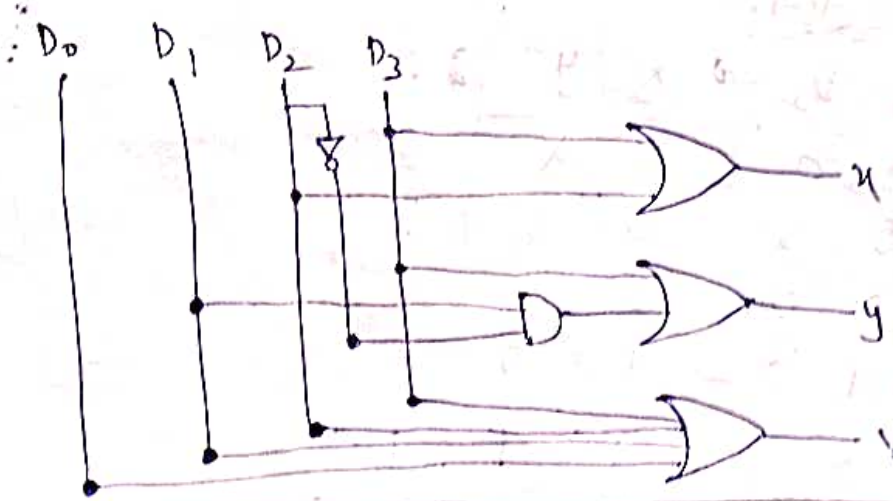


1000 — 8

$V = D_3 + D_2 + D_1\bar{D}_2 + D_0\bar{D}_2$

(or)

$V = D_0 + D_1 + D_2 + D_3$.



Decoder:-

A decoder is a combinational circuit that converts binary information from 'n' input lines to a maximum 2^n unique output lines.

If 'n' bit coded information has unused combinations the decoder may have fewer than 2^n o/ps.

① Draw 3:8 line decoder, circuit with truth table (or). Draw binary to octal converter.

Sol:- Binary to Octal Converter:-

i/p			o/p							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$D_0 = x'y'z'$

$D_1 = x'y'z$

$D_2 = x'yz'$

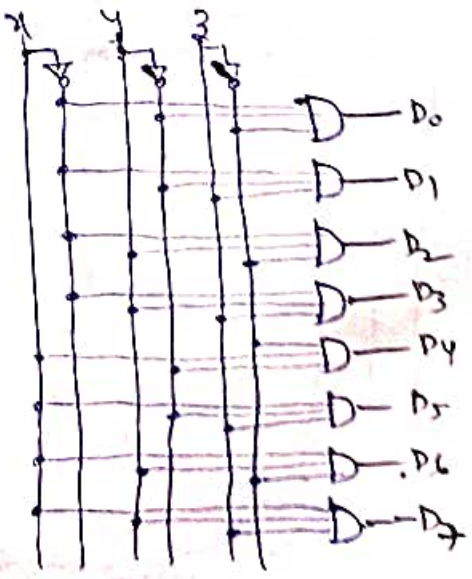
$D_3 = x'yz$

$D_4 = xy'z'$

$D_5 = xy'z$

$D_6 = xyz'$

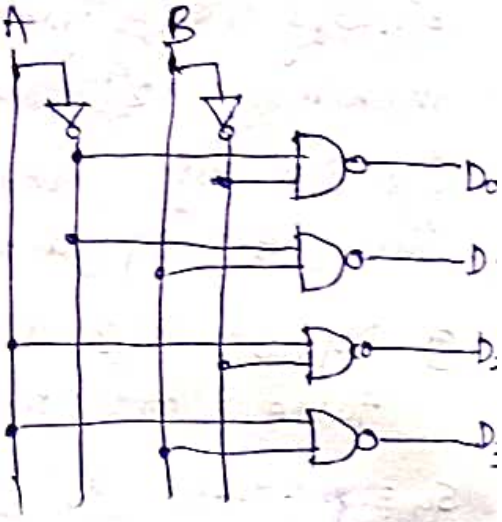
$D_7 = xyz$



② Draw 2:4 line decoder with enable i/p Construction with NAND gate.

Sol:-

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

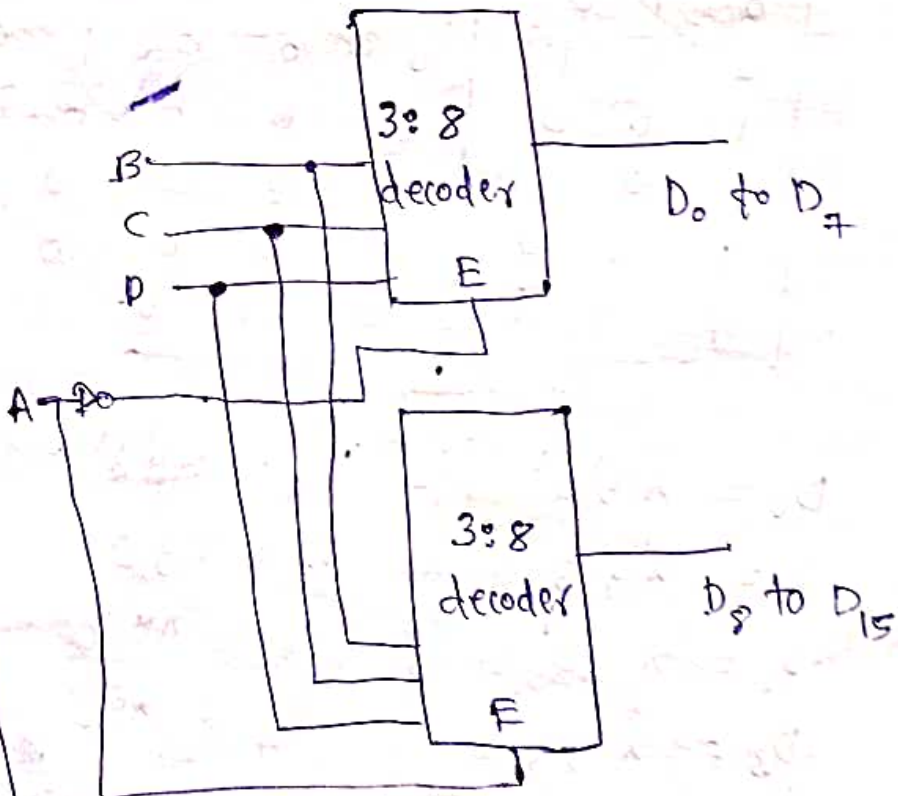


$D_0 = A'B'$, $D_1 = A'B$, $D_2 = AB'$, $D_3 = AB$

③ Construct 4:16 line decoder with two 3:8 line decoders and enable inputs.

Sol:-

A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1



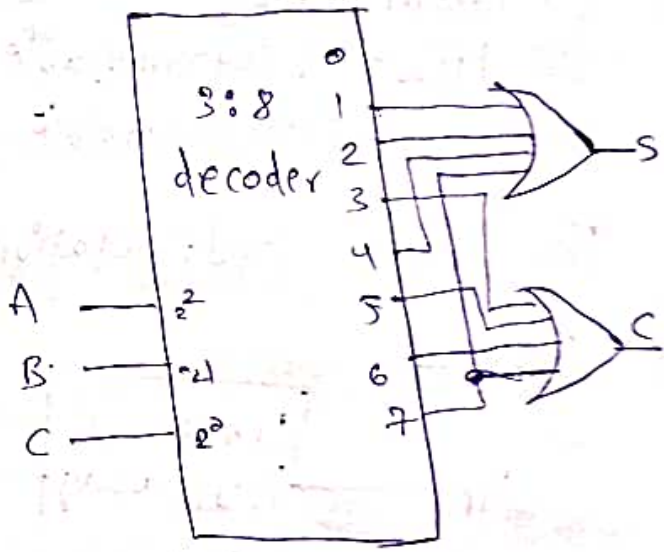
Combinational Logic Implementation:-

A decoder provides the 2^n min terms for 'n' input variables for implementing combinational circuit by means of a decoder and OR gates required.

① Implement a full adder with a decoder.

Sol:-

A	B	C	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

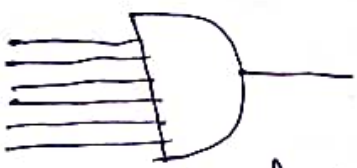


$Carry = \Sigma(3,5,6,7)$

$Sum = \Sigma(1,2,4,7)$

PLDs- Programmable Logic Devices:-

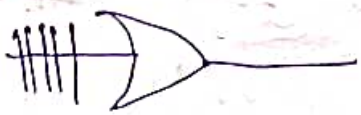
PLDs are that integrated circuits (ICs). They contain an array of AND gates and another array of OR gates. There are 3 types of PLDs. Based on the type of array. which has programmable feature.



Conventional symbol of AND gate

Array Logic symbol of AND gate

Conventional Symbol of OR Gate



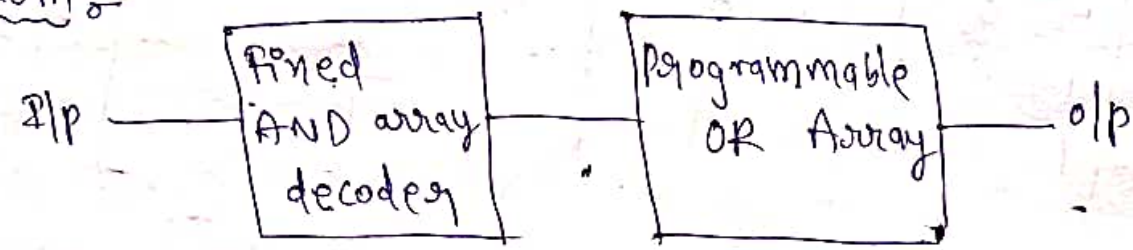
Array Logic Symbol of OR Gate.

The programming in here hardware programming but not software programming. The 3. types of PLDs are

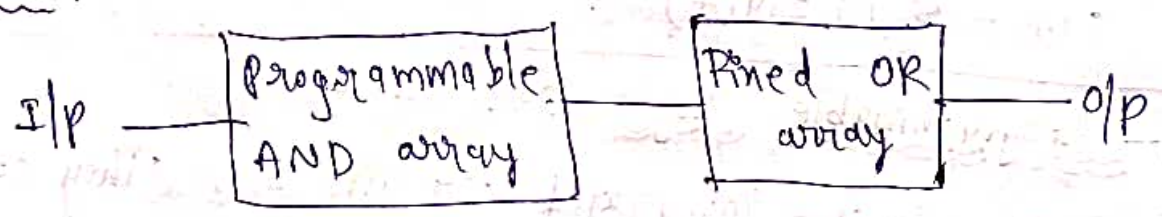
(i) PROM - Programmable Read Only Memory
(ii) PAL — Programmable Array Logic
(iii) PLA - Programmable Logic Array.

The basic configurations of 3 PLDs are

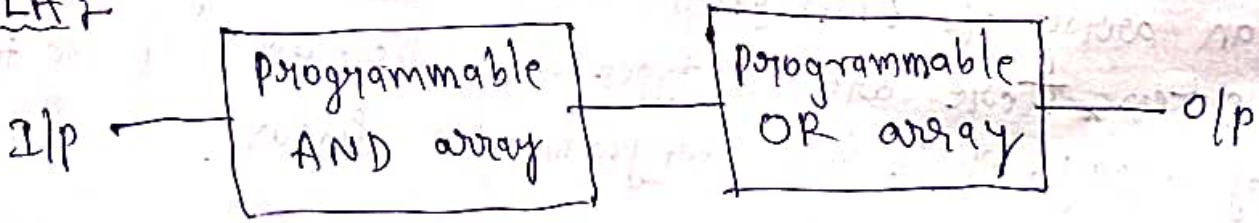
PROM:-



PAL:-

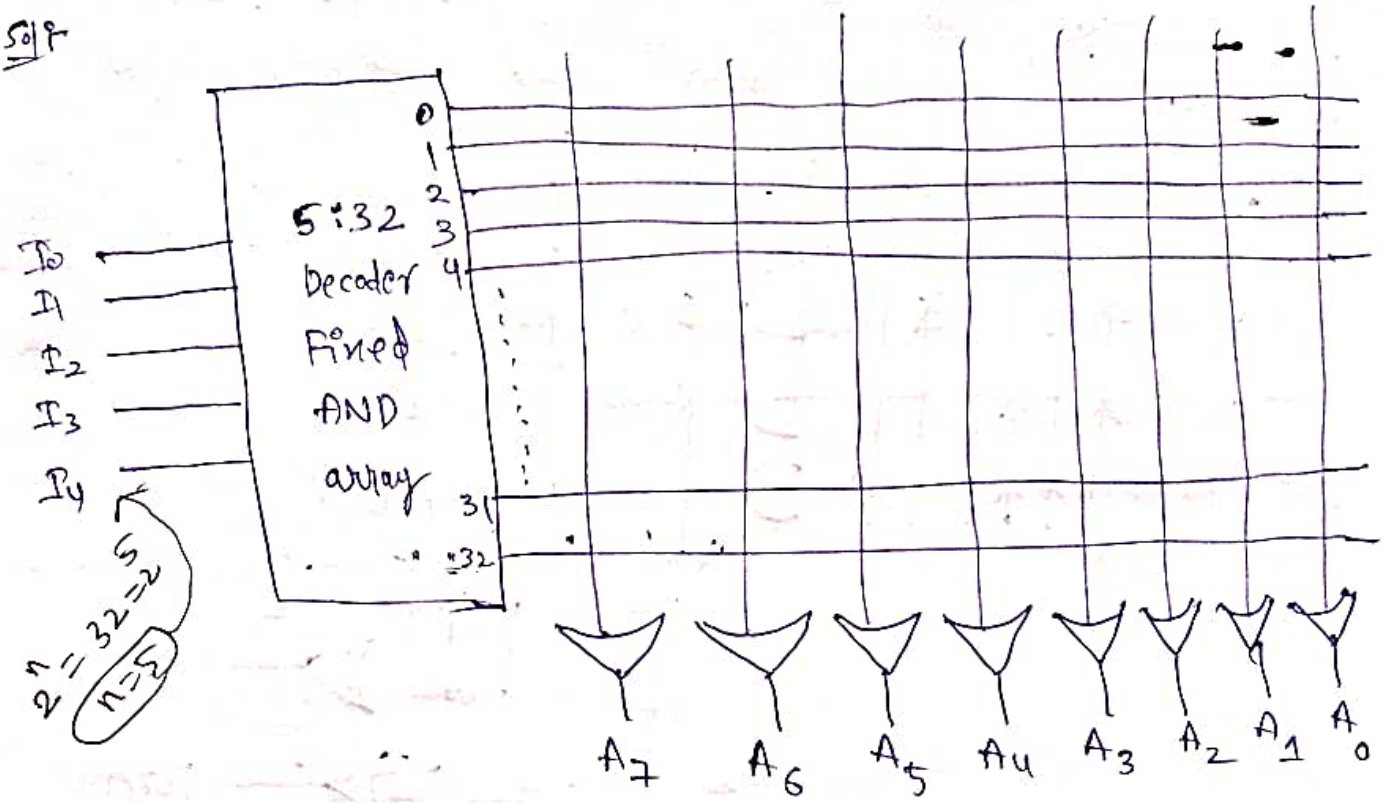


PLA:-



① Draw Internal logic of a 32x8 ROM.

Sol:



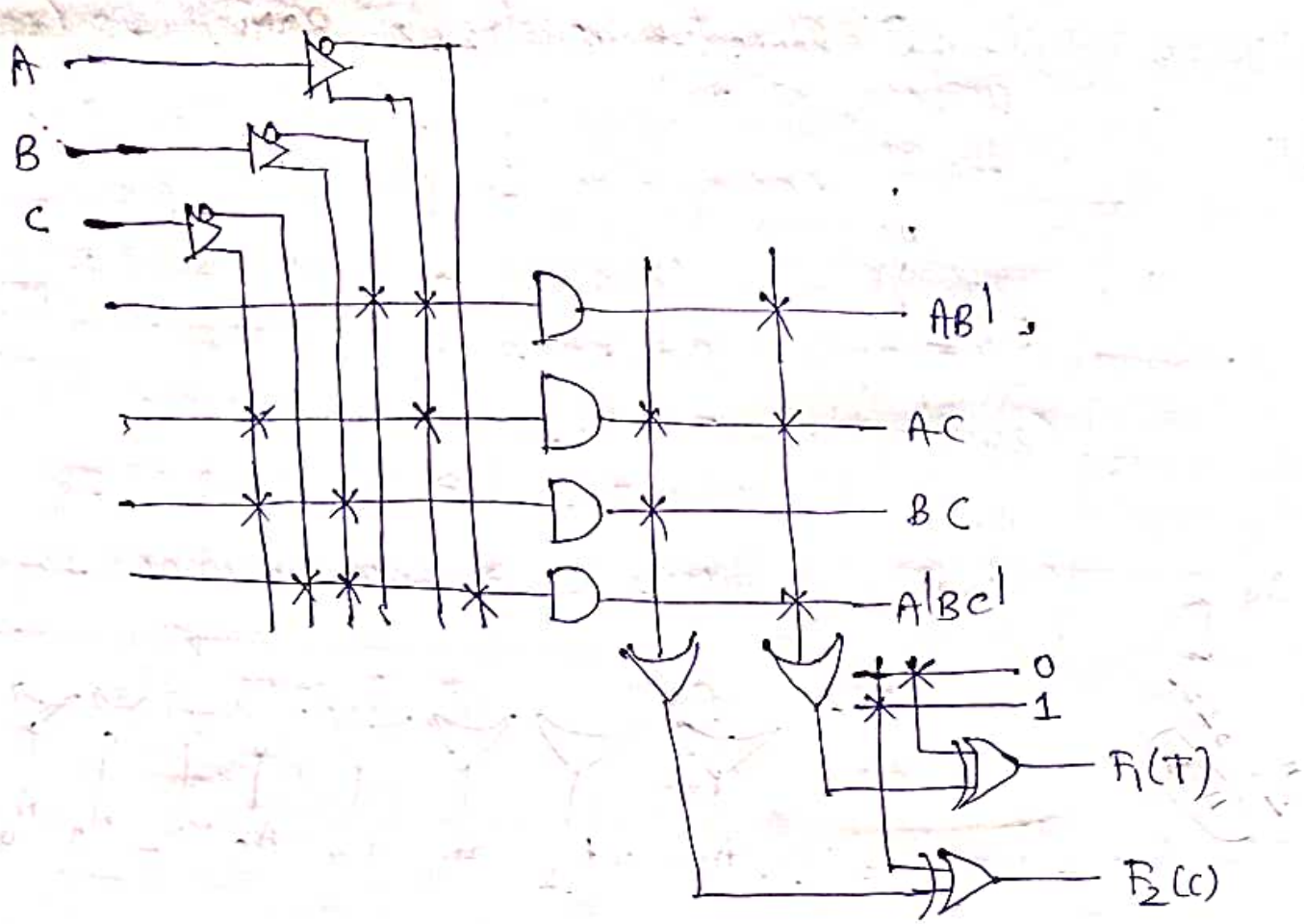
② Implement the following Boolean functions in the PLA.

$F_1 = AB' + AC + A'BC'$, $F_2 = (AC + BC)'$.

Sol:

	Product term	I/P A	I/P B	I/P C	O/P $F_1(T)$	O/P $F_2(C)$
AB'	1	1	0	–	1	–
AC	2	1	–	1	1	1
BC	3	–	1	1	–	1
$A'BC'$	4	0	1	0	1	–

Programming Table

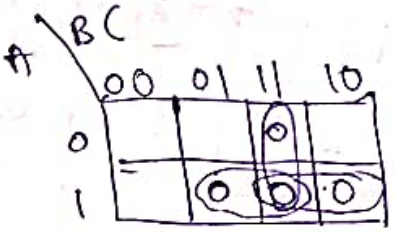


(3) Implement the following two boolean functions with a PLA.

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

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

Sol: given, $F_1(A,B,C) = \Sigma(0,1,2,4)$



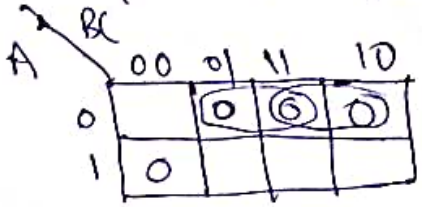
011 → BC

101 → AC

111, 110 → AB

$F_1'(A,B,C) = BC + AC + AB$

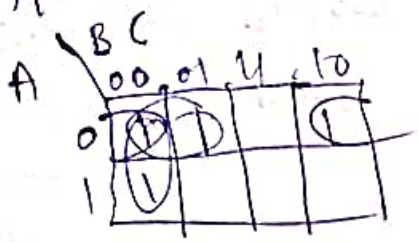
$F_1(A,B,C) = (BC + AC + AB)'$ ⟶ (1)



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

$F_2(A,B,C) = (A'C + A'B + AB'C')'$ ⟶ (2)

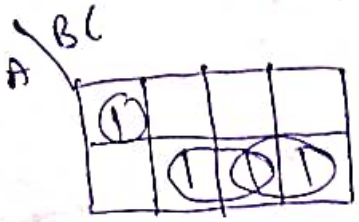
$F_1(A,B,C) = \Sigma(0,1,2,4)$



000, 100 → $B'C'$; 001, 000 → $A'B'$; 000, 010 → $A'C'$

$F_1(A,B,C) = B'C' + A'B' + A'C' \rightarrow (3)$

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



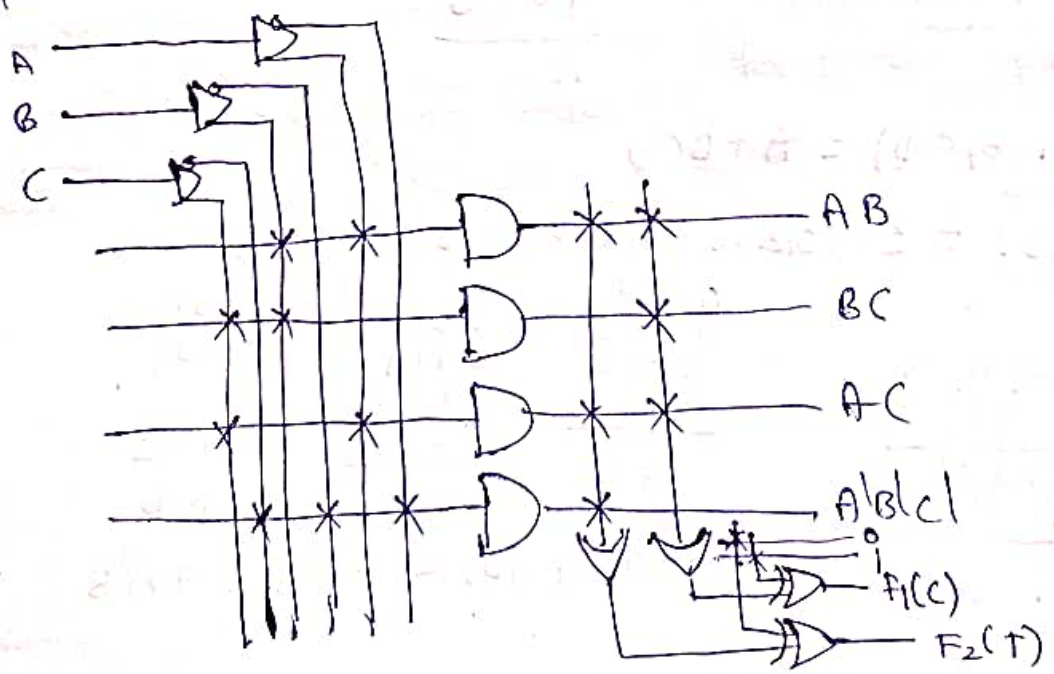
101, 111 → AC ; 110, 111 → AB ; 000 → $A'B'C'$

$F_2(A,B,C) = AC + AB + A'B'C' \rightarrow (4)$

Now, $F_1(A,B,C) = (AB + BC + AC)'$ ↓ from (1) & (4)

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

	Product term	i/p			o/p	
		A	B	C	$F_1(C)$	$F_2(T)$
AB	1	1	1	–	1	1
BC	2	–	1	1	1	–
AC	3	1	–	1	1	1
$A'B'C'$	4	0	0	0	–	1



④ By using PAL design a combinational circuit by considering the following boolean functions given in sum of min terms form

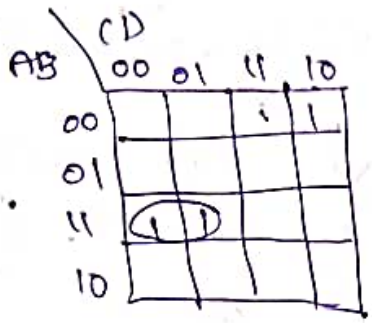
$W(A,B,C,D) = \Sigma(2,12,13)$

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

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

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

Sol:- $W(A,B,C,D) = \Sigma(2,12,13)$

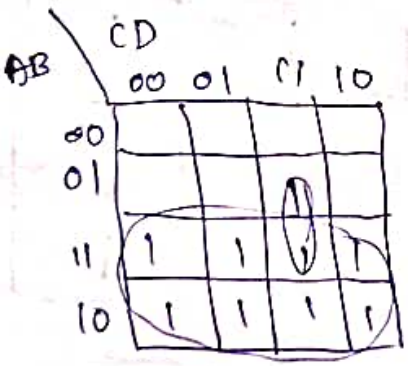


1100
1101
ABC'

0010
A'B'CD'

$\therefore W(A,B,C,D) = ABC' + A'B'CD'$

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

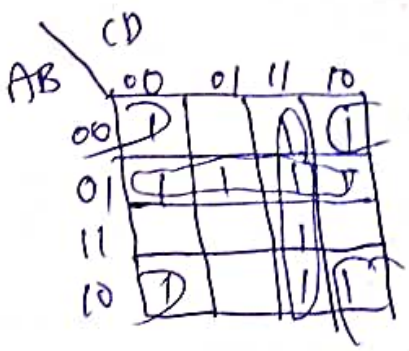


1100
1101
1111
1110
1000
1001
1011
1010
A

0111
1111
BCD

$\therefore X(A,B,C,D) = A + BCD$

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



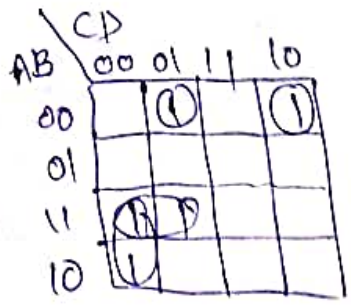
0000
1000
0010
1010
B'D'

0011
0111
1111
1011
CD

0100
0101
0111
0110
A'B

$\therefore Y(A,B,C,D) = B'D' + CD + A'B.$

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



0001 → $A'B'C'D$

0010 → $A'B'CD'$

1100, 1101 → ABC'

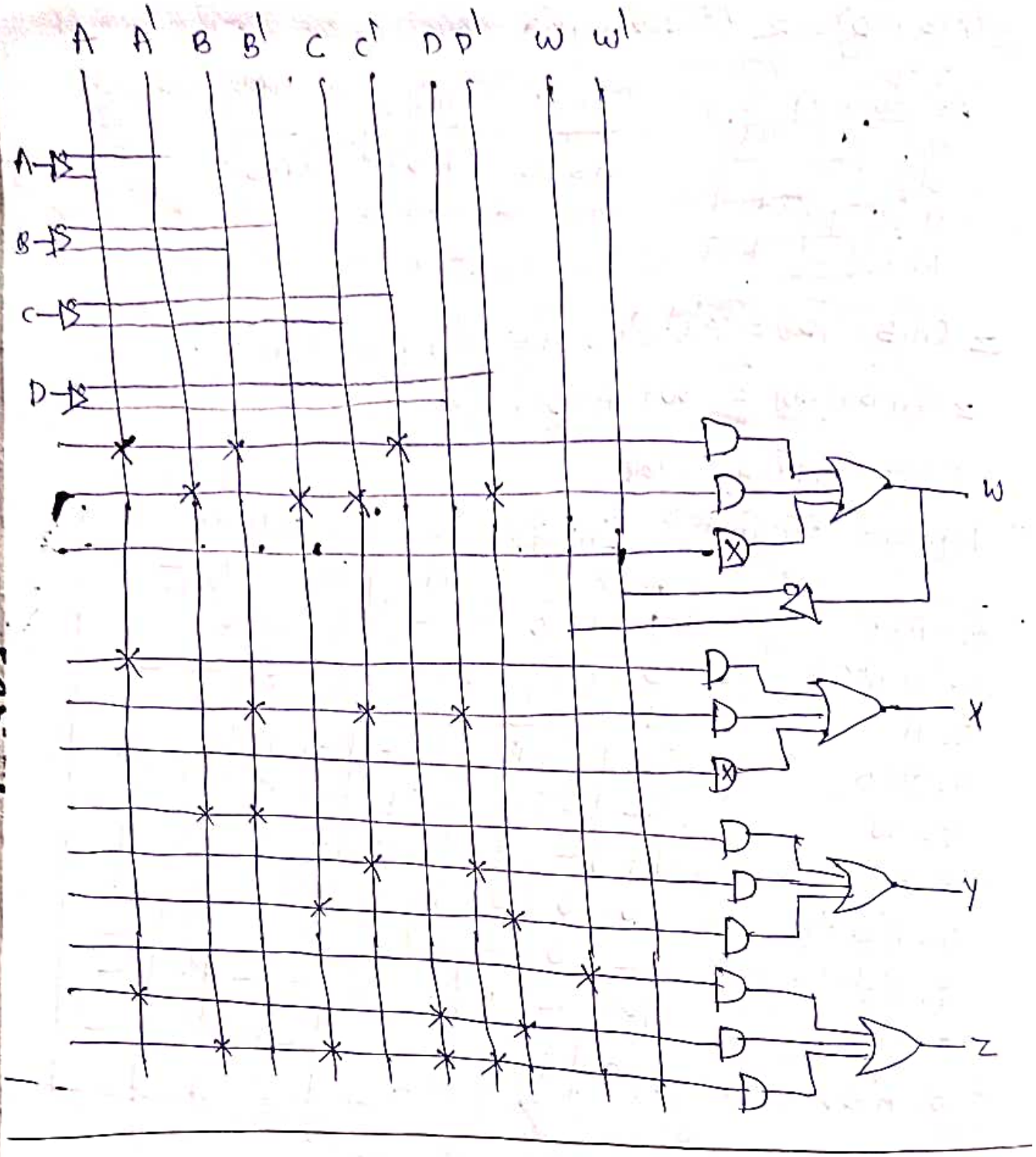
1000, 1100 → $AC'D'$

$Z(A,B,C,D) = A'B'C'D + A'B'CD' + ABC' + AC'D'$.

$Z(A,B,C,D) = W + AC'D' + A'B'C'D$.

Programming Table.

Product Term	Inputs					Outputs			
	A	B	C	D	W	W	X	Y	Z
1. ABC'	1	1	0	–	–	1	–	–	–
2. $A'B'CD'$	0	0	1	0	–	1	–	–	–
3. A	1	–	–	–	–	–	1	–	–
4. BCD	–	1	1	1	–	–	1	–	–
5. W	–	–	–	–	1	–	–	–	1
6. $AC'D'$	1	–	0	0	–	–	–	–	1
7. $A'B'C'D$	0	0	0	1	–	–	–	–	1
8. $B'D'$	–	0	–	0	–	–	–	1	–
9. CD	–	–	1	1	–	–	–	1	–
10. $A'B$	0	1	–	–	–	–	–	1	–


A A' B B' C C' D D' W W'
A
B
C
D
W
X
Y
Z