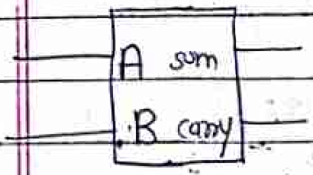
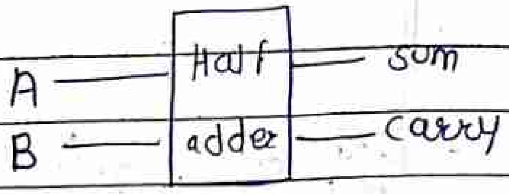
Unit - 3

* Arithmetic Circuits :

1) Half adder.

A logic ckt for addition of two one bit number is reffer to as Half adder





Truth table :

Inputs		Outputs	
A	B	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

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

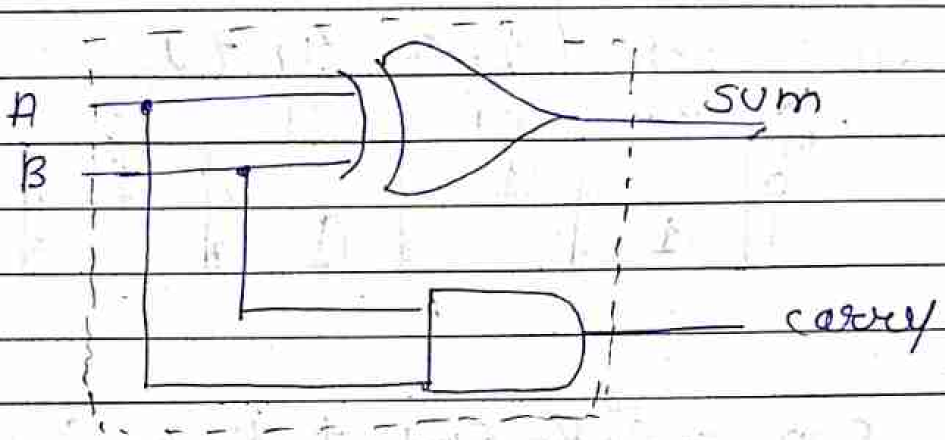
B \ A	0	1
0	$_0$	1_2
1	1_1	$_3$

$$Sum = \overline{A}B + A\overline{B}$$
$$= A \oplus B$$

Carry = $\Sigma m(s)$.

B \ A	0	1
0	0	2
1	1	1 3

carry = AB.

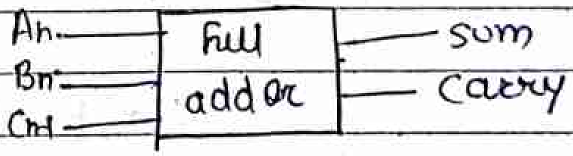


* full Adder.

- defination:

A logic ckt for the addition of two inputs (An & Bn) and carry bit (Cn-1) is reffered to as full adder.

- Truth table.



	Inputs			outputs.	
	An	Bn	Cn-1	Sum	carry
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0

3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	01	1	0	0	1
7	1	1	1	1	1

- $\text{Sum} = \sum m(1, 2, 4, 7)$

C_{n-1} \ A_nB_n	00	01	11	10
0	$_{0}$	1 $_{2}$	$_{6}$	1 $_{4}$
1	1 $_{1}$	$_{3}$	1 $_{7}$	$_{5}$

$$\text{Sum} = \overline{A_n}\,\overline{B_n}\,C_{n-1} + \overline{A_n}\,B_n\,\overline{C_{n-1}} + A_nB_nC_{n-1} + A_n\overline{B_n}\,\overline{C_{n-1}}$$

$$= \overline{A_n}\left(\overline{B_n}\,C_{n-1} + B_n\overline{C_{n-1}}\right)$$
$$A_n\left(\overline{B_n}\,C_{n-1} + \overline{B_n}\,\overline{C_{n-1}}\right)$$

$$= \overline{A_n}\left(B_n \oplus C_{n-1}\right) + A_n\left(B_n \odot C_{n-1}\right) \quad \left(\begin{array}{l} A\oplus B = A\overline{B} + \overline{A}B \\ A\odot B = AB + \overline{A}\,\overline{B} \end{array}\right)$$

But:

$$A \odot B = \overline{A + B}$$

$$= \overline{A_n}\left(B_n \oplus C_{n-1}\right) + A_n\left(\overline{B_n \oplus C_{n-1}}\right)$$

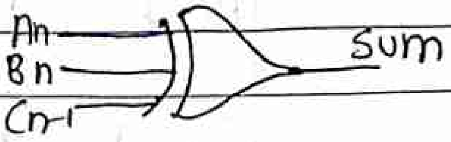
Assume $D = B_n \oplus C_n$.

$$= \overline{A_n}D + A_n\overline{D}$$

$$= A_n \oplus D$$

put the value of D.

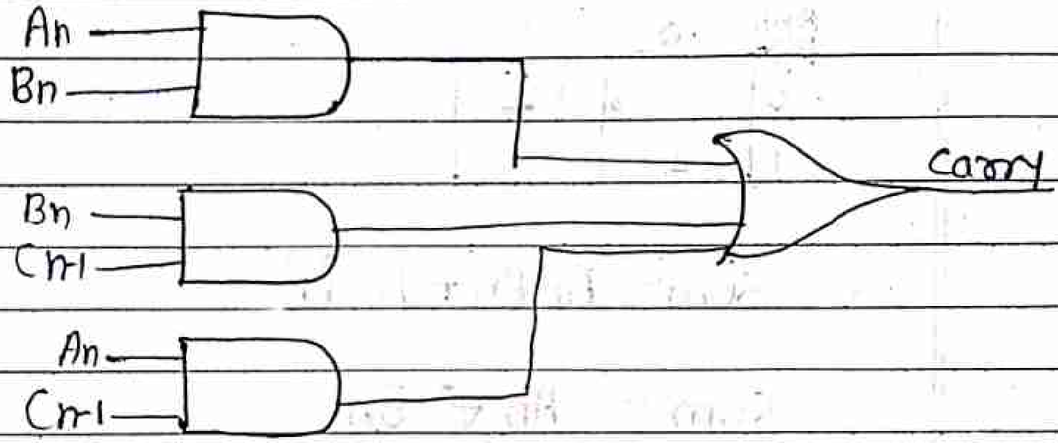
$$\boxed{Sum = A_n \oplus B_n \oplus C_{n-1}}$$



carry = $\Sigma m\,(3, 5, 6, 7)$

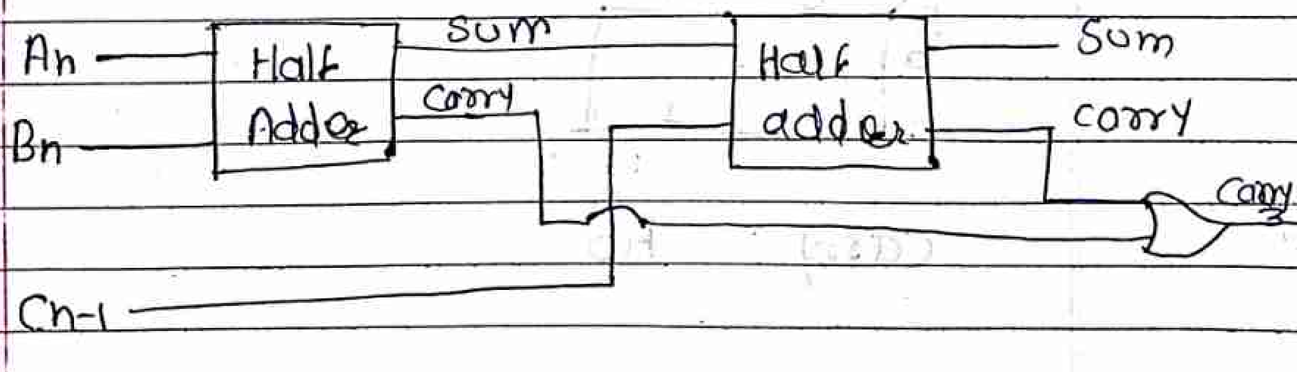
C_{n-1} \ A_nB_n	00	01	11	10
0	0	2	1 6	4
1	1	1 3	1 7	1 5

carry = $A_nB_n + B_nC_{n-1} + A_nC_{n-1}$



Imp

Que: Design full adder ckt using two half adder and one or gate.



Half adder.

Truth table

An	Bn	Sum	Carry.
0	0	0	0
0	1	~~0~~1	0
1	0	1	0
1	1	0	1

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

Bn \ An	0	1
0	$_0$	1 $_2$
1	1 $_1$	$_3$

∴ Sum = $\overline{An}\,Bn + An\,\overline{Bn}$

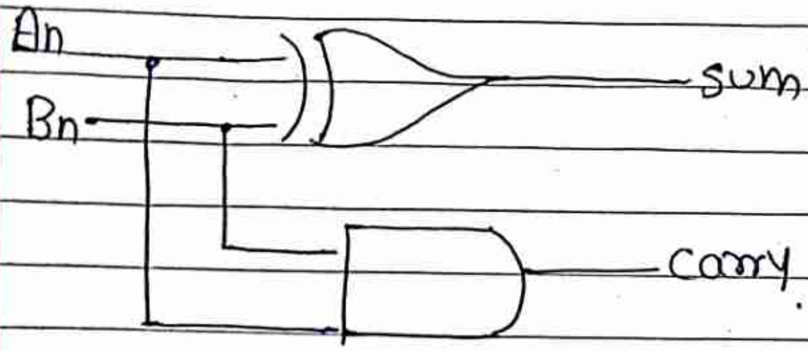
Sum = $An \oplus Bn$

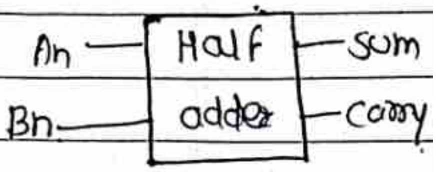
Carry = $\Sigma m (3)$

Bn \ An	0	1
0	$_0$	$_2$
1	$_1$	1 $_3$

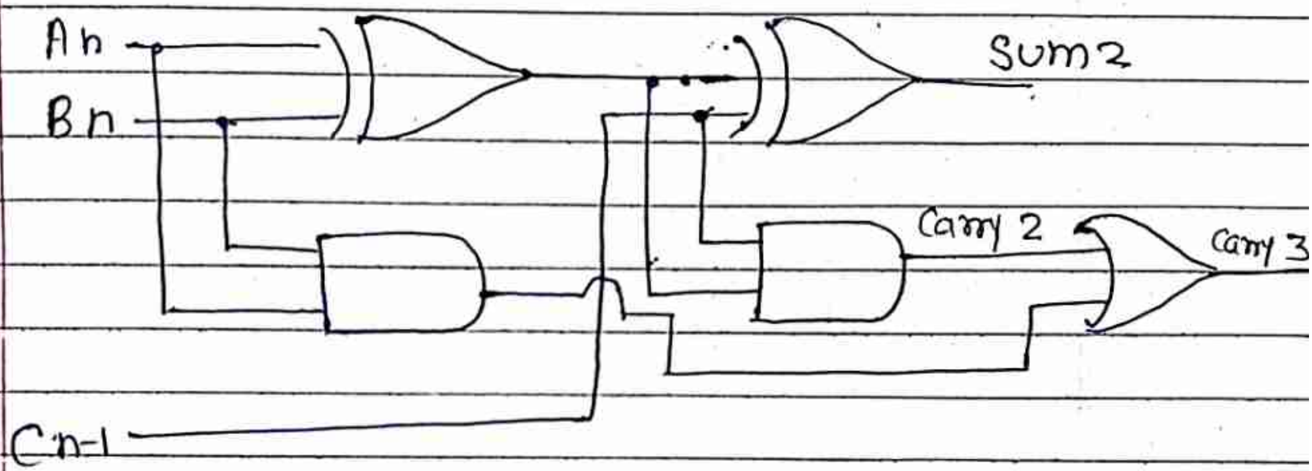
Carry = AB

- ckt for half adder.





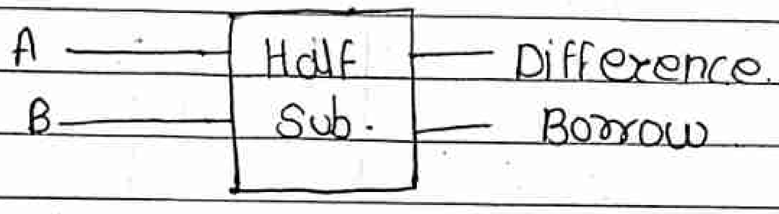
- Full addr ckt. (design ckt)



Truth table for full

* Half Subtractor.

A logic ckt for the substraction of B (subtrahends) from A (minuend) where A & B are one bit numbers is reffered to as half subtractor.



- Truth table.

	Inputs		outputs	
	A	B	Diff.	Borrow.
0	0	0	0	0
1	0	1	1	1
2	1	0	1	0
3	1	1	0	0

Diff = $\Sigma m(1, 2)$.

K-map

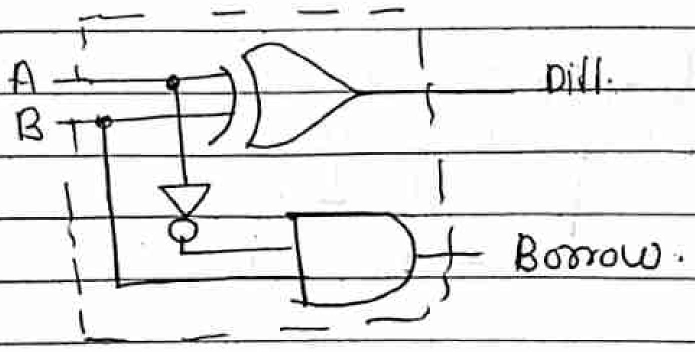
$\therefore$ Diff = $\overline{A}B + A\overline{B}$

Diff = $A \oplus B$

Borrow = $\Sigma m(1)$

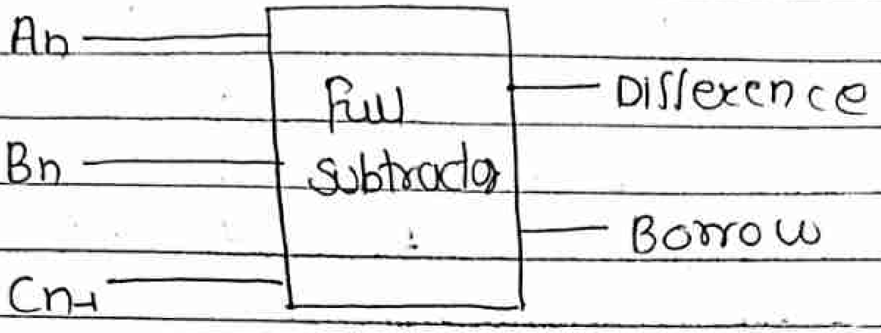
K-map.

$\therefore$ Borrow = $\overline{A}B$



* Full subtractor

A logic ckt which will perform multibit substraction where in Borrow from prevows been position may also be there



⊙ Truth Table

	Inputs			Outputs	
	An	Bn	Cn-1	Diff	Borrow.
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

$\cdot \text{Diff} = \Sigma m(1, 2, 4, 7)$

Cn-1 \ AnBn	00	01	11	10
0	0	1 2	6	1 4
1	1 1	3	1 7	5

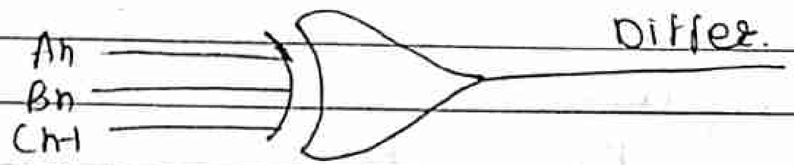
$$\text{Diff} = \overline{A_n}\,\overline{B_n}\,C_{n-1} + \overline{A_n}\,B_n\,\overline{C_{n-1}} + A_n B_n C_{n-1} + A_n \overline{B_n}\,\overline{C_{n-1}}$$

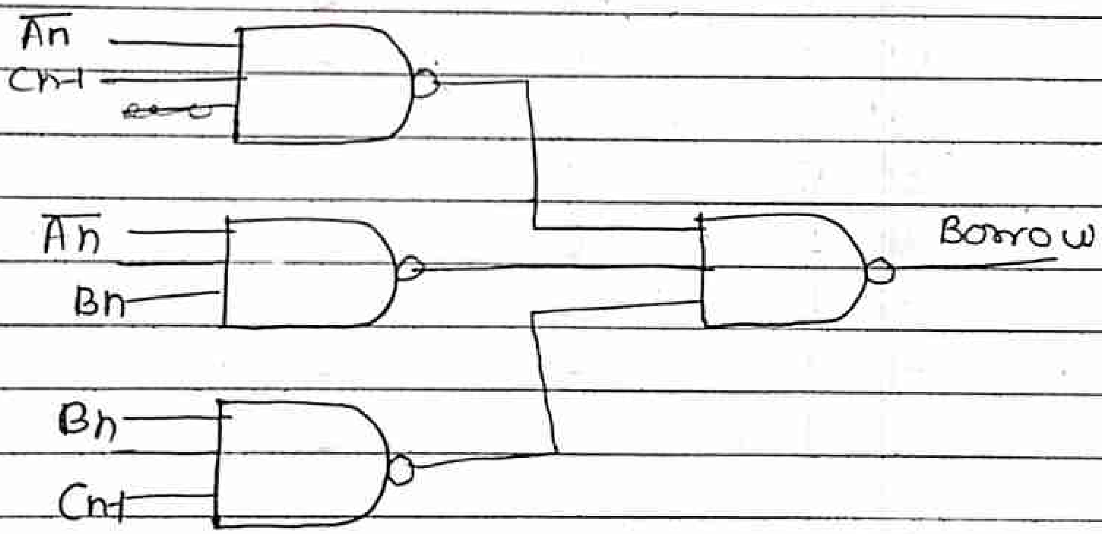
$\text{Diff} = A_n \oplus B_n \oplus C_{n-1}$

$\text{Borrow} = \sum m(1, 2, 3, 7).$

$C_{n-1} \backslash A_nB_n$	00	01	11	10
0	$_0$	1 $_2$	$_6$	$_4$
1	1 $_1$	1 $_3$	1 $_7$	$_5$

$\text{Borrow} = \overline{A_n} C_{n-1} + \overline{A_n} B_n + B_n + C_{n-1}$

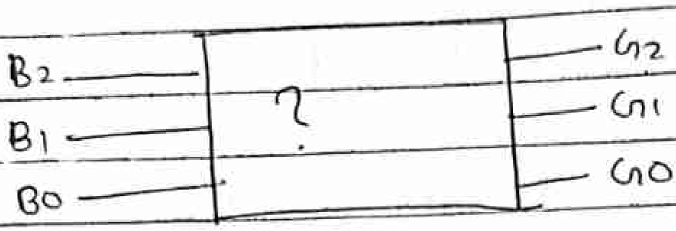




logic ckt.

* Code Converter :

Que: Convert 3-digit binary to gray code.



: Truth table

	Inputs			outputs		
	B_2	B_1	B_0	G_2	G_1	G_0.
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

→

$G_2 = \Sigma m(4,5,6,7)$

B_0 \ B_2B_1	00	01	11	10
0	0	2	1 6	1 4
1.	1	3	1 7	1 5

$$\boxed{G_2 = B_2}$$

$G_1 = \sum m\,(2,3,4,5)$

$B_0 \backslash B_2B_1$	00	01	11	10
0	$_0$	1_2	$_6$	1_4
1	$_1$	1_3	$_7$	1_5

$G_1 = \overline{B_2}B_1 + B_2B_1$

$\boxed{G_1 = B_2 \oplus B_1}$

→

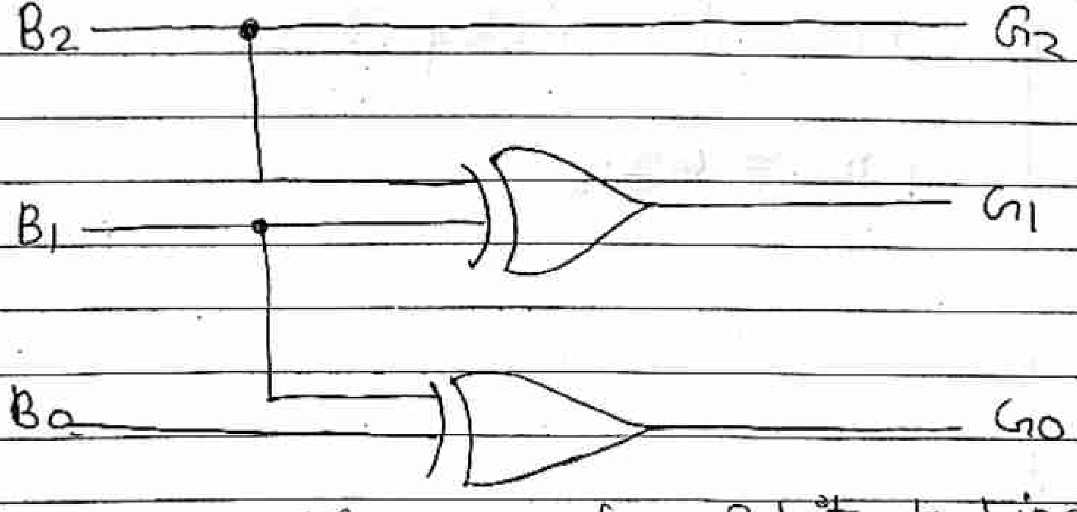
$G_0 = \sum m\,(1,2,5,6)$

$B_0 \backslash B_2B$	00	01	11	10
0	$_0$	1_2	1_6	$_4$
1	1_1	$_3$	$_7$	1_5

$G_0 = B_1\overline{B_0} + B_0\overline{B_1}$

$\boxed{G_0 = B_1 \oplus B_0}$

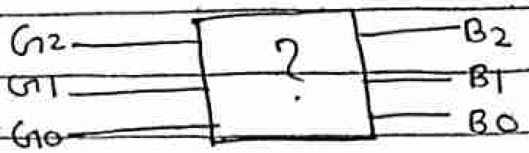
- logical ckt.



ckt diagram for 3-bit to binary converter

Q. Convert 3 bit gray to binary code.

Solⁿ ⟹



- Truth table

	Inputs			outputs		
	G_2	G_1	G_0	B_2	B_1	B_0.
0	0	0	0	0	0	0
1	0	0	1	0	0	1
3	0	1	1	0	1	0
2	0	1	0	0	1	1
6	1	1	0	1	0	0
7	1	1	1	1	0	1
5	1	0	1	1	1	0
4	1	0	0	1	1	1

$B_2 = \sum m(4,5,6,7)$

$G_0 \backslash G_2G_1$	00	01	11	10
0	$_{0}$	$_{2}$	1_{6}	1_{4}
1	$_{1}$	$_{3}$	1_{7}	1_{5}

$$\boxed{B_2 = G_2}$$

$B_1 = \Sigma m(2, 3, 5, 4)$

G_0 \ G_2G_1	00	01	11	10
0	0	1 2	0 6	1 4
1	1	1 3	0 7	1 5

$B_1 = \overline{G_2}G_1 + G_2\overline{G_1} \quad = G_2 \oplus G_1$

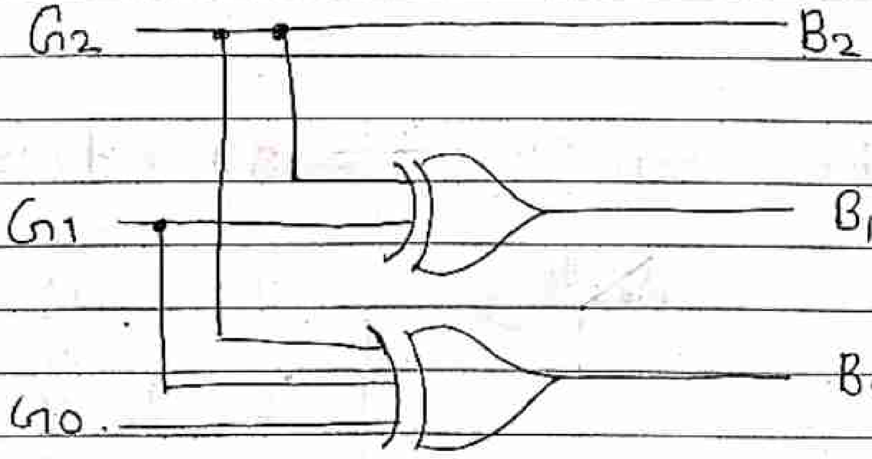
~~$B_0 = \Sigma m(1, 2, 7, 4)$~~ $\Sigma m(1, 2, 4, 7)$

G_0 \ G_2G_1	00	01	11	10
0	0	1 2	6	1 4
1	1 1	3	1 7	5

$B_0 = \overline{G_2}\,\overline{G_1}\,G_0 + \overline{G_2}\,G_1\overline{G_0} + G_2G_1G_0 + G_2\overline{G_1}\,\overline{G_0}$

$B_0 = G_2 \oplus G_1 \oplus G_0$

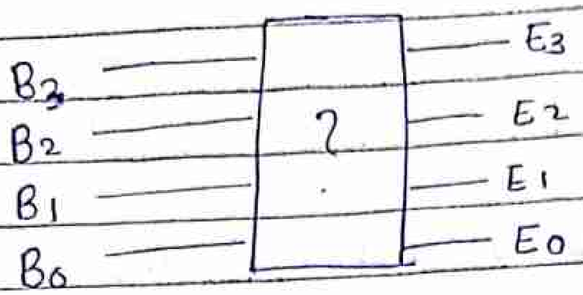
* logic ckt diagram.



(3 bit gray to binary code converter)

Imp
Que: Design BCD to excess -3 convertor.

⇒



Truth table :

	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

$E_3 = \Sigma m(5,6,7,8,9) + d(10,11,12,13,14,15)$

B_1B_0 \ B_3B_2	00	01	11	10
00	0	4	X 12	1 8
01	1	1 5	X 13	1 9
11	3	1 7	X 15	X 11
10	2	1 6	X 14	X 10

$E_3 = B_3 \cancel{B_2} + B_2 B_0 + B_2 B_1$ — (1)

$E_2 = \Sigma m(1,2,3,4,9) + d(10,11,12,13,14,15)$

B_1B_0 \ B_3B_2	00	01	11	10
00	0	1 4	X 12	8
01	1 1	5	X 13	1 9
11	1 3	7	X 15	X 11
10	1 2	6	X 14	X 10

$E_2 = \overline{B_2} B_0 + \overline{B_2} B_1 + B_2 \overline{B_1}\, \overline{B_0}$ — (2)

$E_1 = \Sigma m(0,3,4,7,8) + d(10,11,12,13,14,15)$

B_1B_0 \ B_3B_2	00	01	11	10
00	1 0	1 4	X 12	1 8
01	1	5	X 13	9
11	1 3	1 7	X 15	X 11
10	2	6	X 14	X 10

$E_1 = \overline{B_1}\, \overline{B_0} + B_1 B_0$

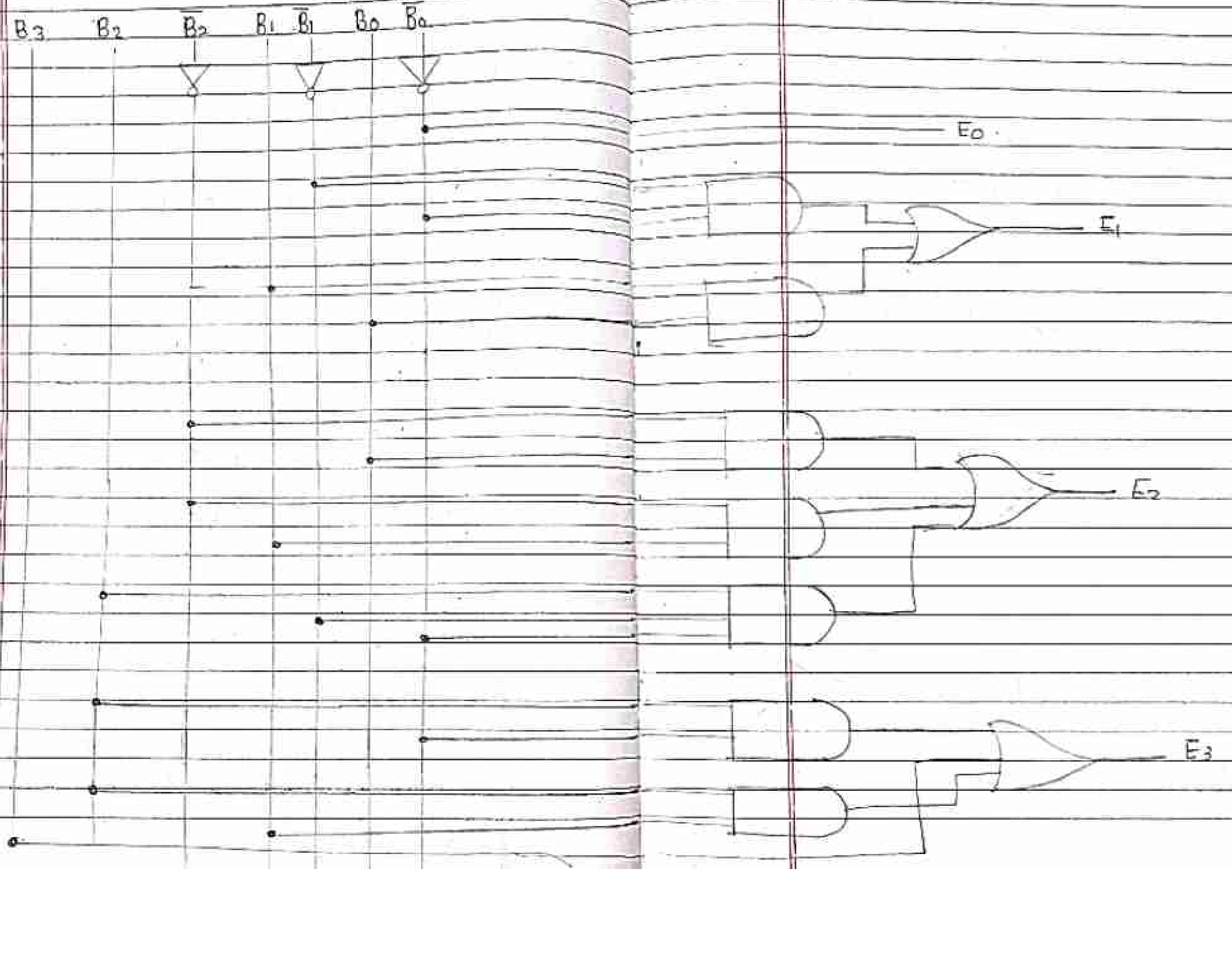
$\boxed{E_1 = B_1 \odot B_0}$

$E_0 = \Sigma m(0,2,4,6,8) + d(10,11,12,13,14,15)$

B_1B_0 \ B_3B_2	00	01	11	10
00	1 0	1 4	X 12	1 8
01	1	5	X 13	9
11	3	7	X 15	X 11
10	1 2	1 6	X 14	X 10

$\boxed{E_0 = \overline{B_0}}$

* Ckt For BCD to excess-3 convertor



* Multiplexer (mux) (Data selector)

→ Many i/ps & single output

→ select input lines available

→ Active low enable pin is available for cascading purpose

→ It is also called as data selector.

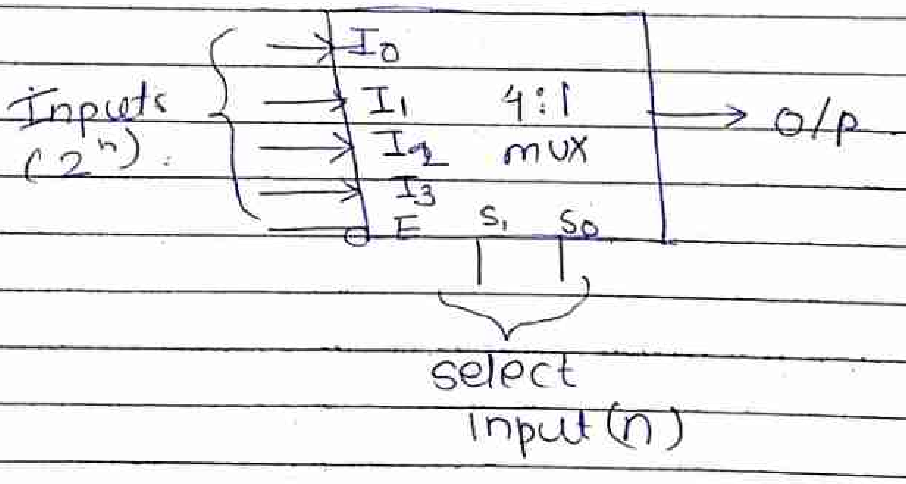
→ standard size available are 2:1, 4:1, 8:1 & 16:1

→ Advantages

1) Simplification of logic expression is not required.

2) It minimises the IC package out

3) Logic design is simplified

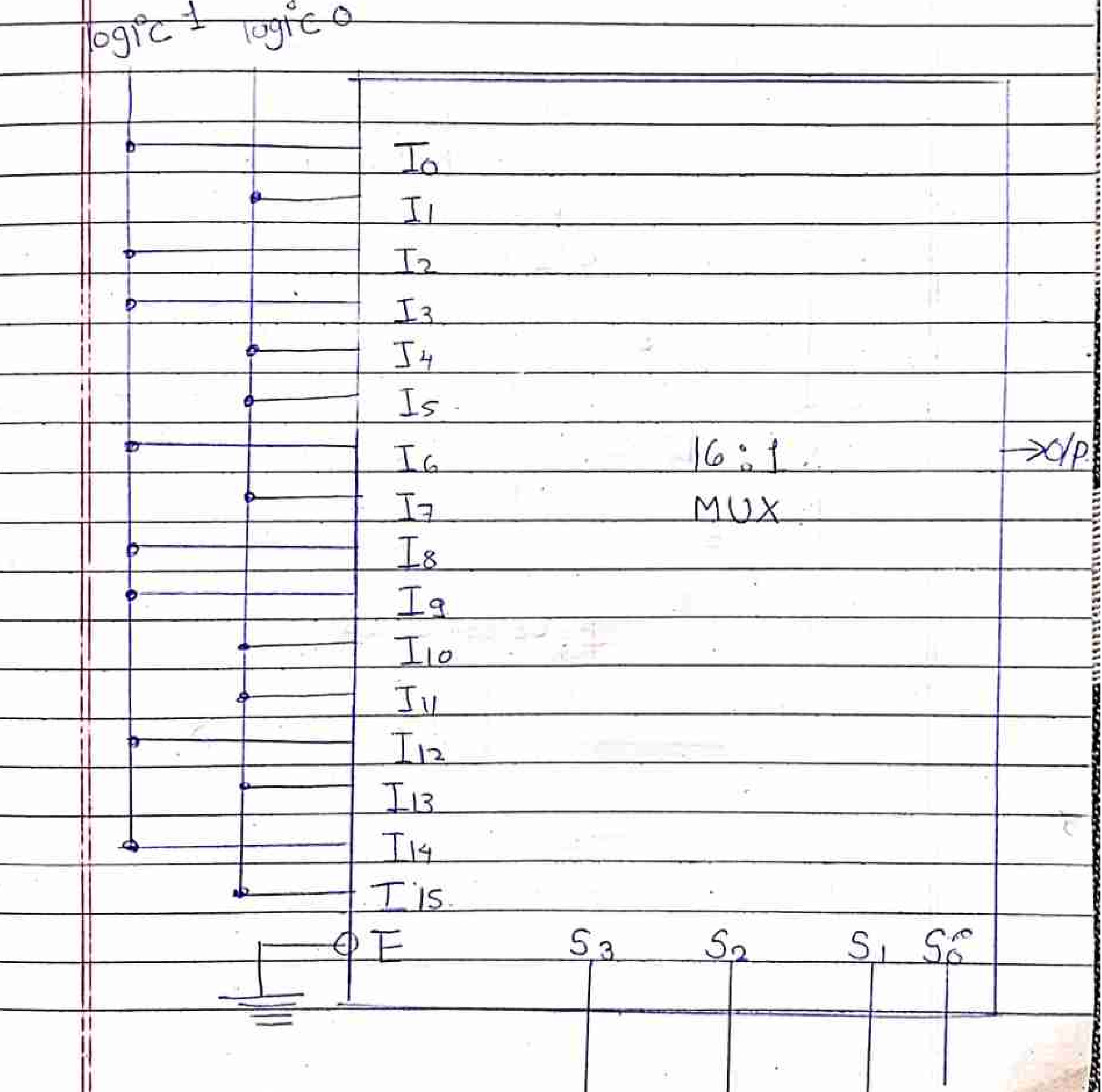


Que: Impliment the expression using a multiplexer

$F(A,B,C,D) = \Sigma m(0,2,3,6,8,9,12,14)$

Sol⇒ The problom will solve using 16:1 MUX, the inputs available in 16:1 MUX are 16

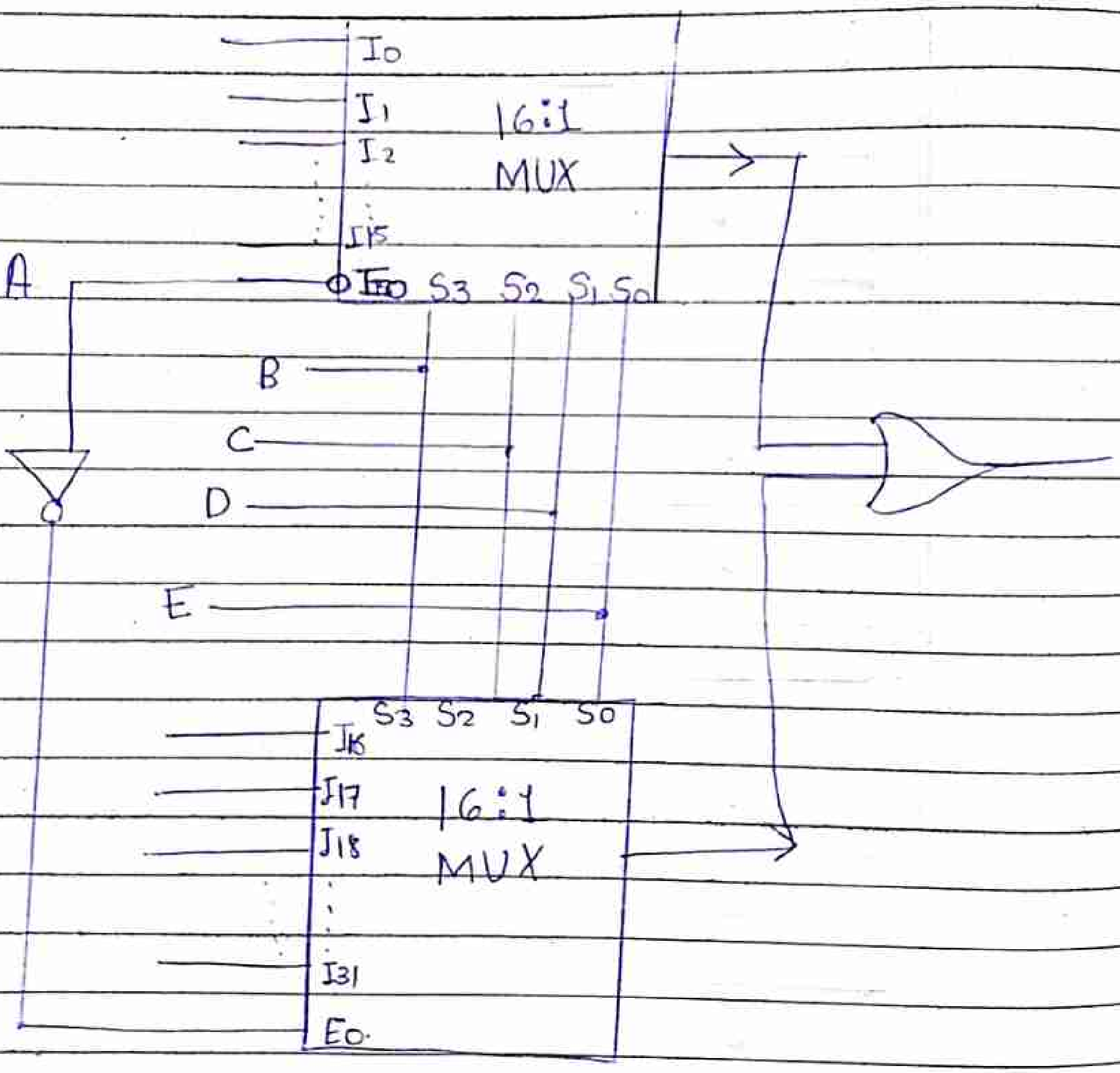
The select lines avaiable for (2^n) 16:1 are 4 (n)



Que: Design 32:1 MUX using 16:1 MUX

Soln ⇒ Number of 16:1 MUX required are 2

The select lines required for 32:1 ~~mug~~ MUX 5
For 16:1 mux select lines required as 4



Que: Design 64:1 MUX using 8:1 MUX

Sol^{n} ⇒

For 64:1 MUX the number of select inputs are 6

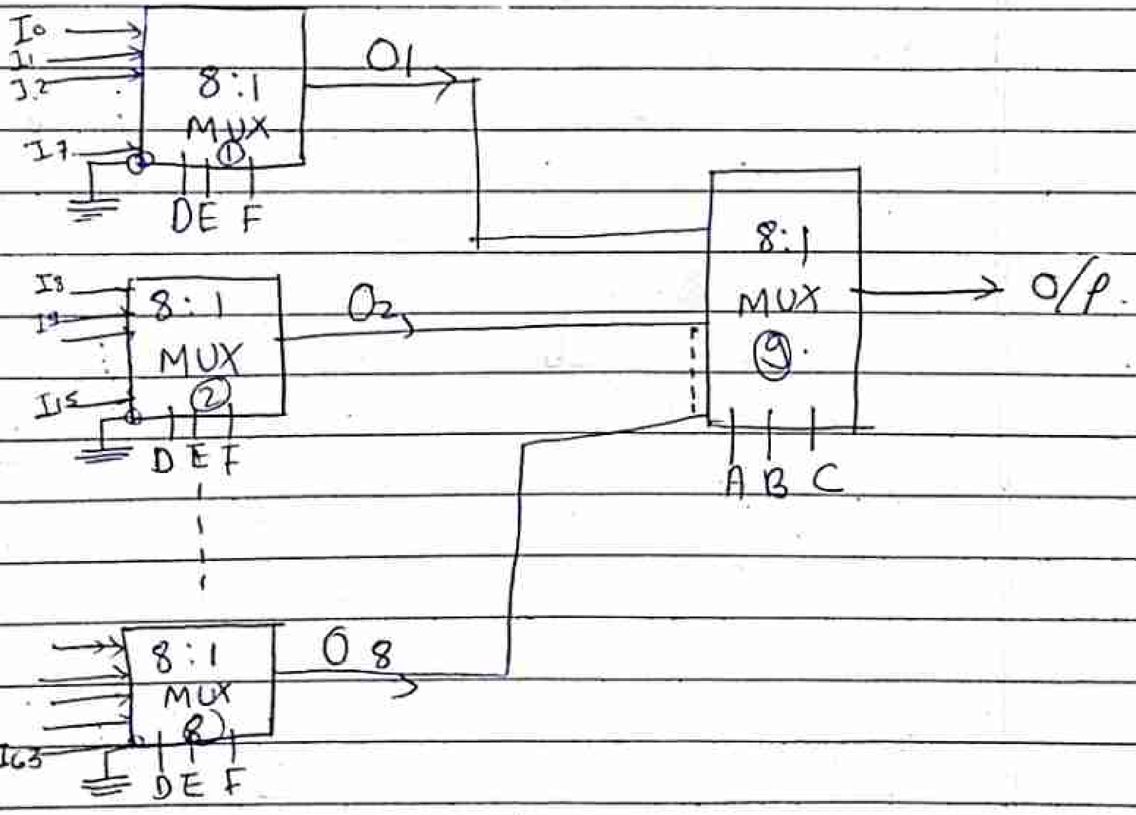
∴ 6 select lines are required

For 8:1 MUX number of select line required are 3

Number of IC 8:1 required in first stage of design are $\frac{64:1}{8:1} = 8$

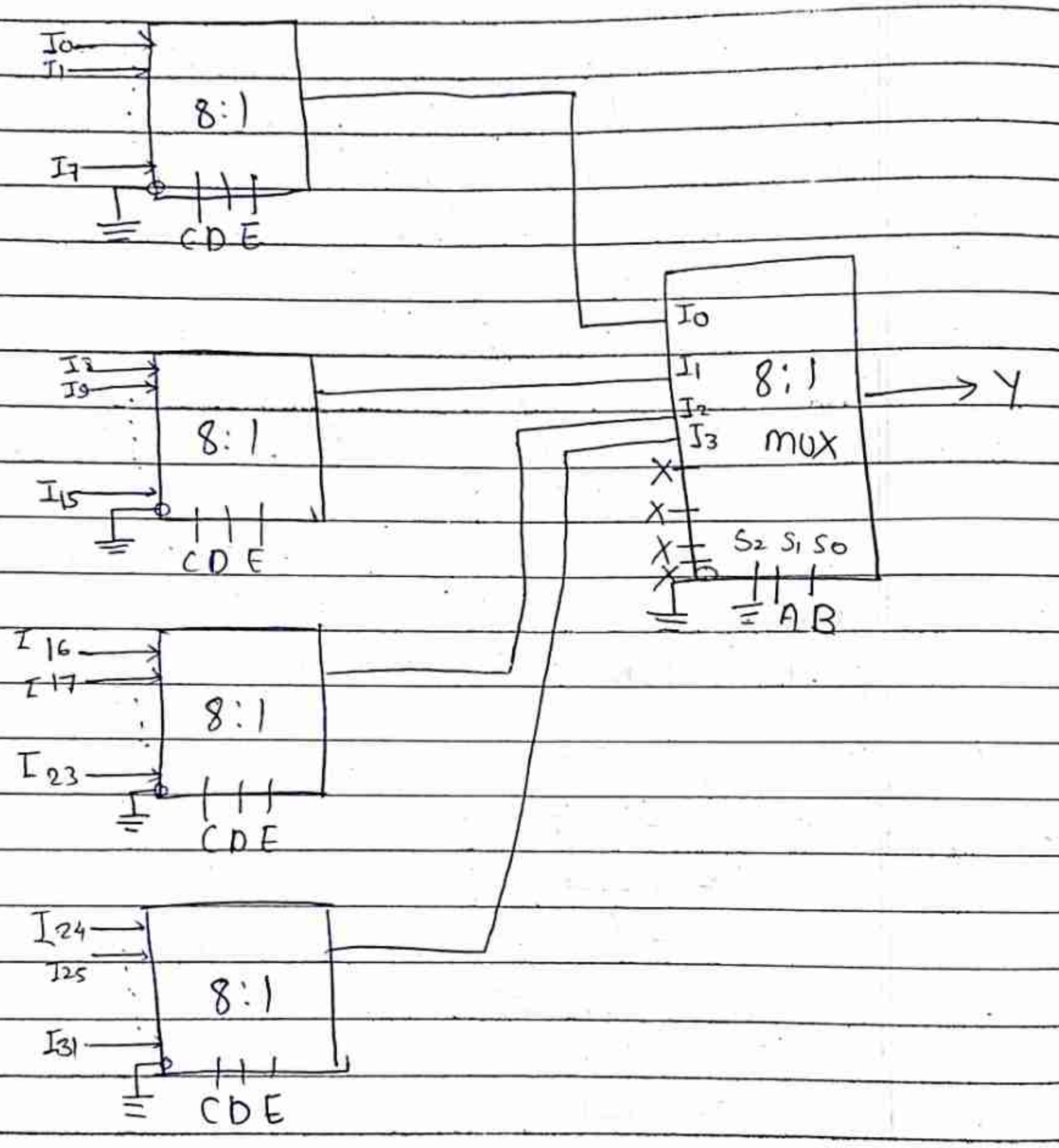
The the number of 8:1 required in second stage of design is $\frac{8:1}{8:1} = 1$.

The variables used to connect select Select lines are A, B, C, D, E, F where A is MSB and F is LSB



Que: Design 32:1 MUX using 8:1 MUX only

⟹



W2018 - Q5
W2017 - Q.5(b)
S2019 - Q6.
S2018 - Q6.
S2017 -

Rules for implementation Table

1. If two min terms in a column are not circled apply zero two the corresponding multiplexer input.

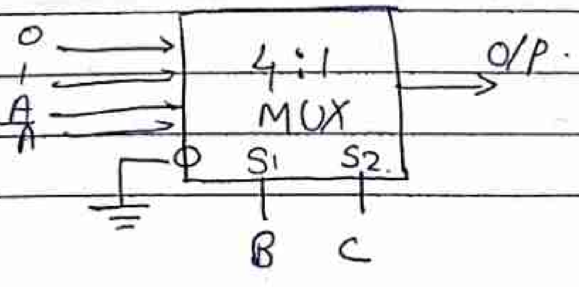
2. If two minterms are encircled apply ~~to~~ 1 to the corresponding multiplexer input

3. If the bottom minterm is encircled and top is not encircled apply A to corresponding to ~~the~~ ~~corresponding~~ input.

4. If the top minterm is encircled and bottom is not circled apply $\bar{A}$ to corresponding the multiplexer input

$f = \Sigma m(1, 3, 5, 6)$

	I_0	I_1	I_2	I_3
$\bar{A}$	0	①	2	③
A	④	⑤	⑥	7
	0	1	A	$\bar{A}$



ABC.

	I_0	I_1	I_2	I_3
$\overline{A}$	ABC 000	ABC 001	ABC 010	ABC 011
A	ABC 100	ABC 101	ABC 110	ABC 111

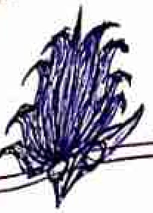
	I_0	I_1	I_2	I_3
$\overline{A}$	0	1	2	3
A	4	5	6	7.

	I_0	I_1	I_2	I_3
$\overline{C}$	000	010	100	110
C	001	011	101	111

	I_0	I_1	I_2	I_3
$\overline{C}$	0	2	4	6
C	1	3	5	7

	I_0	I_1	I_2	I_3
$\overline{B}$	000	001	100	101
B	010	011	110	111

	I_0	I_1	I_2	I_3
$\overline{B}$	0	1	4	5
B	2	3	6	7



Que: Implement the following function using 4:1 MUX

4 marks.

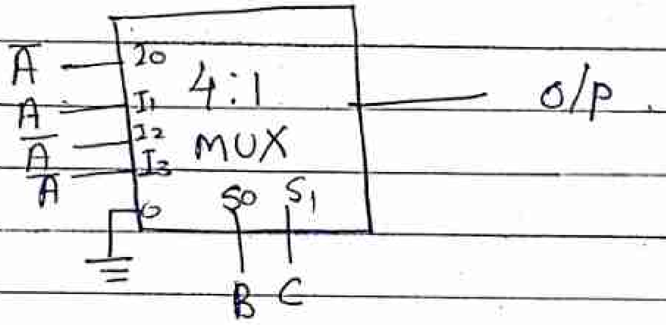
$f(A,B,C) = \Sigma m\,(0,2,3,5)$

Assume the used with the A, B & C

Soln ⇒

	I_0	I_1	I_2	I_3
$\bar{A}$	$\textcircled{0}$	1	$\textcircled{2}$	$\textcircled{3}$
A	4	$\textcircled{5}$	6	7.
	$\bar{A}$	A	$\bar{A}$	$\bar{A}$

we are daking variable A' to input side



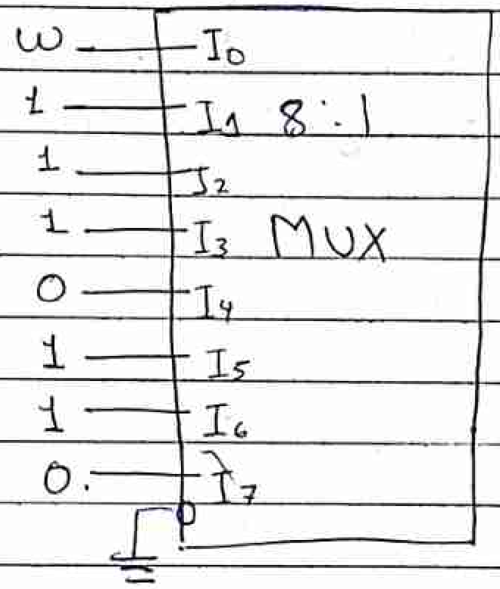
Que: I

Que: Implement a given function using 8:1 MUX $f(W,X,Y,Z) = \Pi M(0,4,7,12,15)$

$\Rightarrow F(W,X,Y,Z) = \Pi M(0,4,7,12,15)$.

$f(W,X,Y,Z) = \Sigma m(1,2,3,5,6,8,9,10,11,13,14)$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\overline{W}$	0	1	2	3	4	5	6	7
W	8	9	10	11	12	13	14	15
	w	1	1	1	0	1	1	0

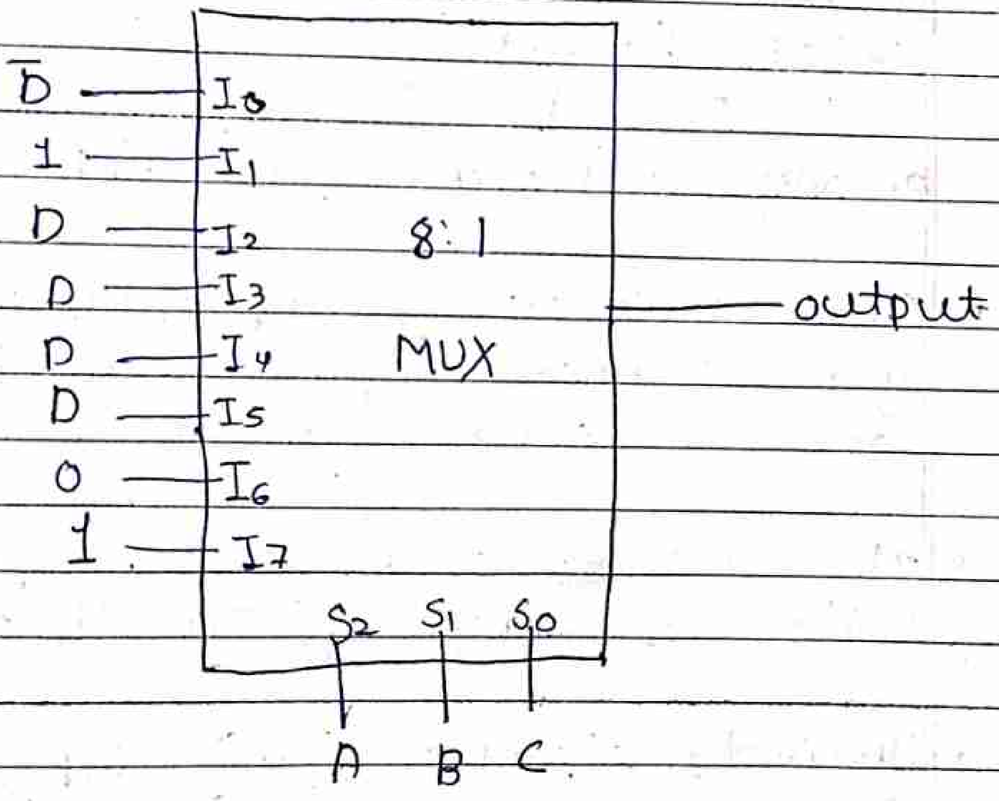


Que: Implement the following function using 8:1 MUX select A, B, C as select lines.

→

∴ $f(A, B, C, D) = \Sigma m(0, 2, 3, 5, 7, 9, 11, 14, 15)$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\overline{D}$	⓪	②	4	6	8	10	12	⑭
D	1	③	⑤	⑦	⑨	⑪	13	⑮
	$\overline{D}$	1	D	D	D	D	0	1



Encoder

encoder has 2^n (or less than 2^n) input lines and 'n' outputs line

The output lines generated the binary code corresponding to input value

It is a device whose inputs are decimal digits and / or alphabetic Characters and whose outputs are the coded representation of those inputs

Encoding is a process of converting familar numbers or symbols into a coded format

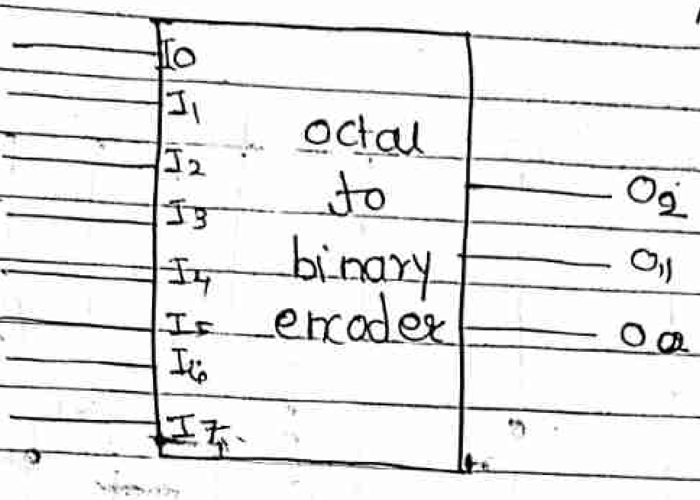
Example is octal to binary encoder, decimal to BCD encoder

- The encoder is implemented with OR gates whose inputs are determined directly from truth table
- In encoder only one input can be active at any given time
- Ambiguty → when all o/p - '0' is generated when all i/ps are zero

NO-Kmap required.

ques * Design octal to binary encoder

⟹



Truth Table

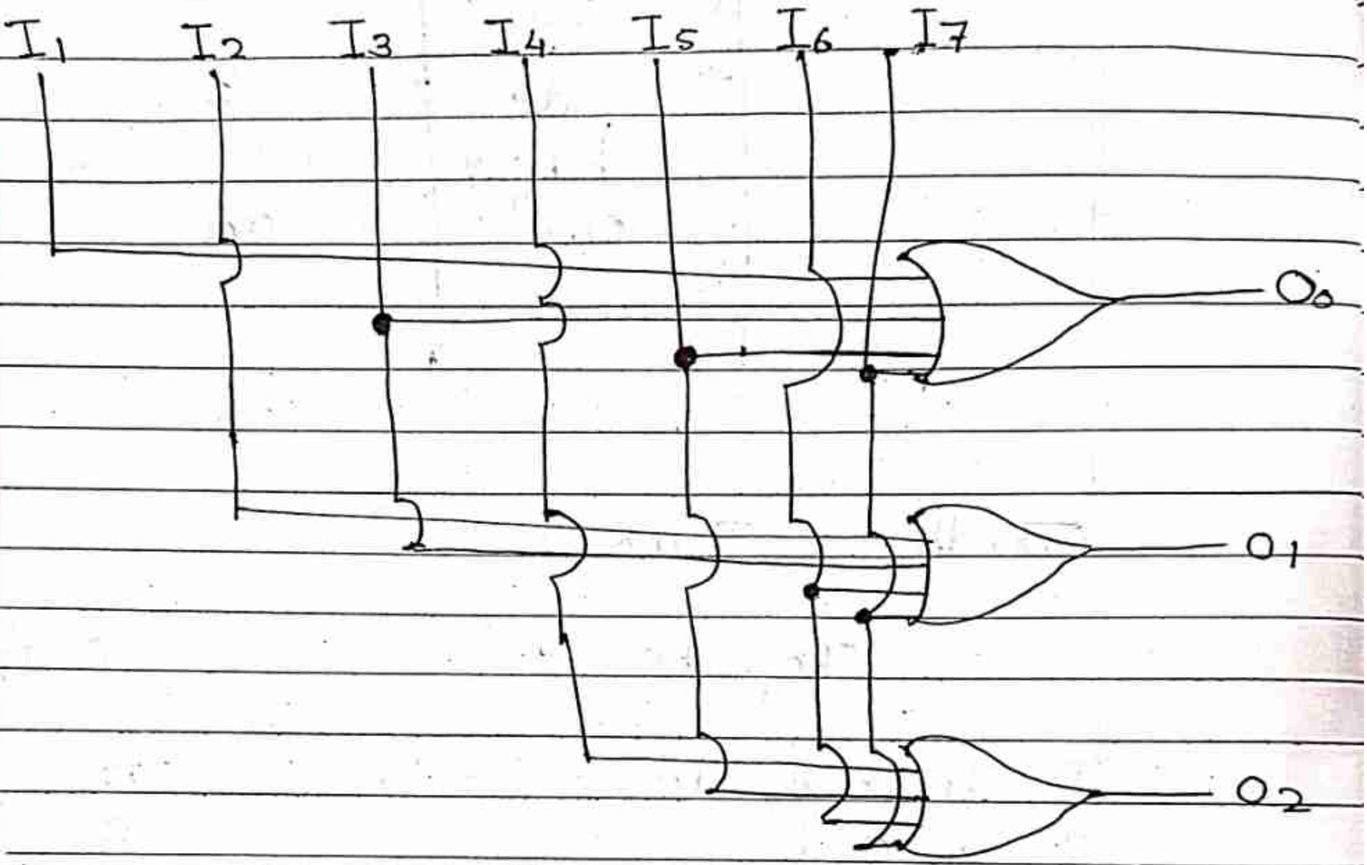
Inputs								Outputs		
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	O_2	O_1	O_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$O_0 = I_1 + I_3 + I_5 + I_7$

$O_1 = I_2 + I_3 + I_6 + I_7$

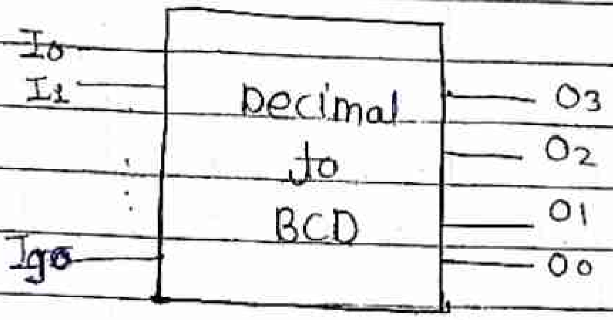
$O_2 = I_4 + I_5 + I_6 + I_7$

Design of octol to binary encoder.



logic diagram for ~~binar~~ octal to binary encoder.

Que : Design Decimal to BCD encoder



Truth Table

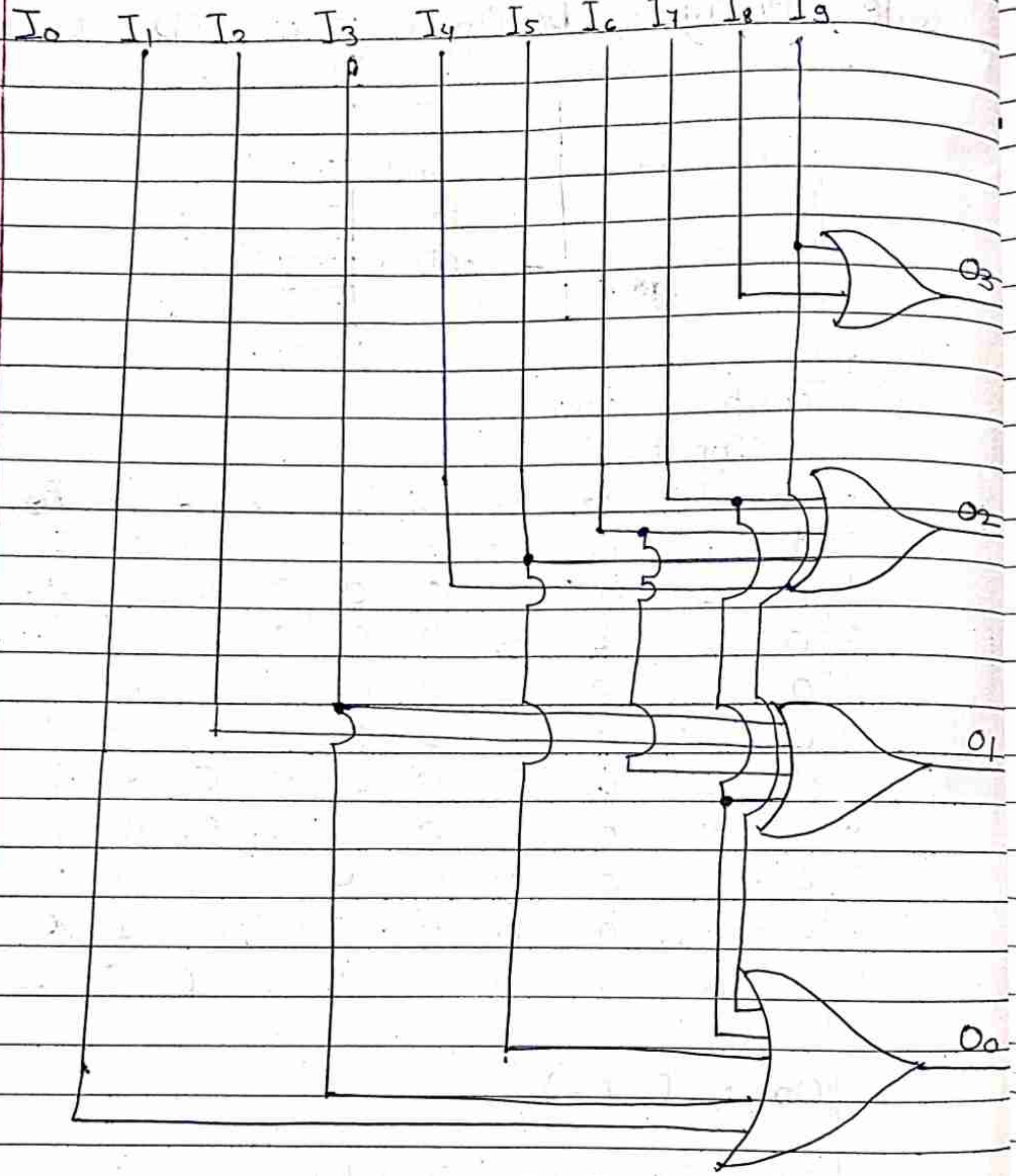
	Inputs										outputs			
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	O_3	O_2	O_1	O_0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	0	0	0	1	0	1
6	0	0	0	0	0	0	1	0	0	0	0	1	1	0
7	0	0	0	0	0	0	0	1	0	0	0	1	1	1
8	0	0	0	0	0	0	0	0	1	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	1	1	0	0	1

$O_3 = I_8 + I_9$

$O_2 = I_4 + I_5 + I_6 + I_7$

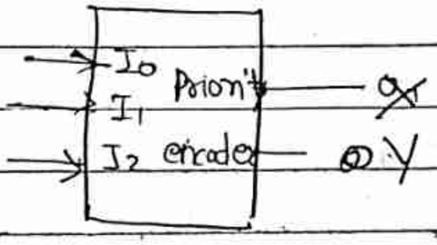
$O_1 = I_2 + I_3 + I_6 + I_7$

$O_0 = I_1 + I_3 + I_5 + I_7 + I_9$


I0
I1
I2
I3
I4
I5
I6
I7
I8
I9
O3
O2
O1
O0

Que: Design three bit priorty encoder

→ Let three bits of priorty encoder are I_2 I_1 I_0
Let outputs are denoted by X & Y



- priority Table.

Inputs	priority X	Y	
I_0	0	1	lowest
I_1	1	0	↓
I_2	1	1	Highest.
All i/op are zero	0	0.	

- Truth Table.

	I_2	I_1	I_0	x	y
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	1	0
3	0	1	1	1	0
4	1	0	0	1	1
5	1	0	1	1	1
6	1	1	0	1	1
7	1	1	1	1	1

$\therefore X = \sum m(2,3,4,5,6,7)$

$\therefore$

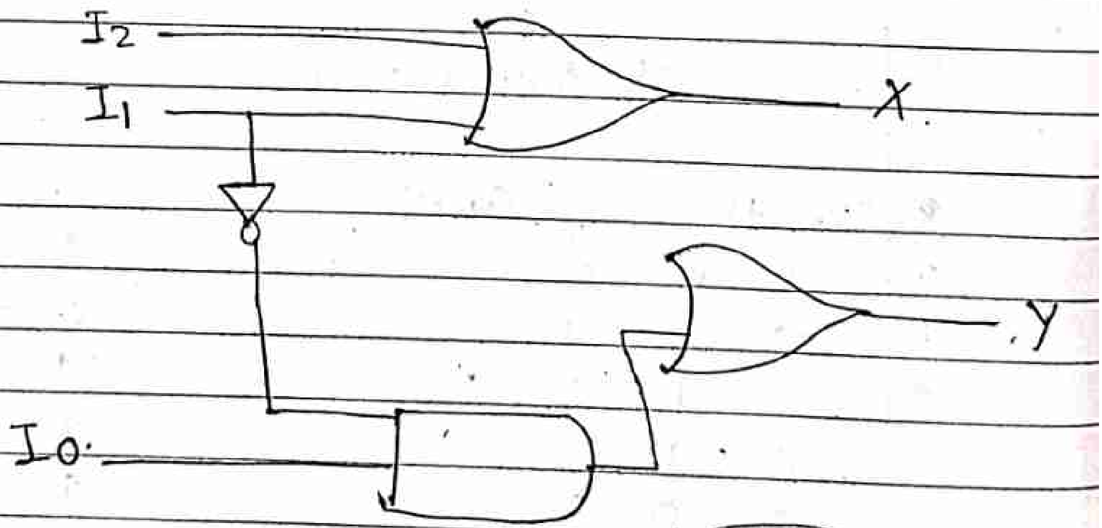
$I_0 \backslash I_2 I_1$	00	01	11	10
0	$_{0}$	1_{2}	1_{6}	1_{4}
1	$_{1}$	1_{3}	1_{7}	1_{5}

$$\boxed{X = I_1 + I_2}$$

$\therefore Y = \sum m(1,4,5,6,7)$

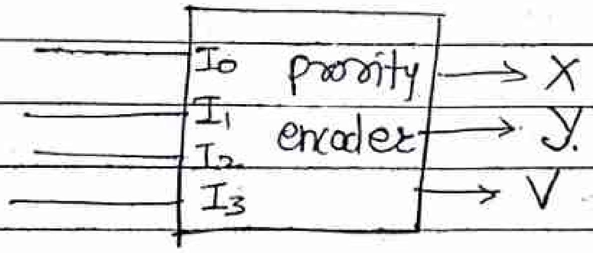
$I_0 \backslash I_2 I_1$	00	01	11	10
0	$_{0}$	$_{2}$	1_{6}	1_{4}
1	1_{1}	$_{3}$	1_{7}	1_{5}

$X = I_2 + \overline{I_1} I_0$



logic circuit for priority three bit encoder que: Design 4 input prionty encoder

→ Assume 4 inputs of prionty encoder are denoted by I3, I2, I1, I0 and three outputs are denoted by X, Y V (valid output indicator).



- Priority Table:

I		X	Y	V	
I_0	Φ	Ø	Ø	1	lowest
I_1		0	1	1	↓
I_2		1	0	1	
I_3		1	1	1	Higest.
All i/ps zero.		X	X	0.	

- Truth Table:

	Inputs				outputs.		
	I_3	I_2	I_1	I_0	X	Y	V.
0)	0	0	0	0	X	X	0
1)	0	0	0	1	0	0	1
2)	0	Φ	1	0	0	1	1
3)	0	Ø	1	1	0	1	1
4)	0	Φ	0	0	1	0	1
5)	0	Φ	0	1	1	0	1
6)	0	1	1	0	1	0	1

7)	0	1	1	1	1	0	1
8)	1	0	0	0	1	1	1
9)	1	0	0	1	1	1	1
10)	1	0	1	0	1	1	1
11)	1	0	1	1	1	1	1
12)	1	Φ	0	0	1	1	1
13)	1	Φ	0	1	1	1	1
14)	1	1	1	0	1	1	1
15)	1	1	1	1	1	1	1

$X = \Sigma m(4,5,6,7,8,9,10,11,12,13,14,15,)$ ~~6~~$+d(0)$

$Y = \Sigma m(2,3,8,9,10,11,12,13,14,15)+d(0)$

$V = \Sigma m(1,2,3,4,5,6,7,8,9,10,12,13,14,15)$

I_1I_0 \ I_3I_2	00	01	11	10
00	X 0	1 4	1 12	1 8
01	1	1 5	1 13	1 9
11	3	1 7	1 15	1 11
10	2	1 6	1 14	1 10

$X = I_2 + I_3$

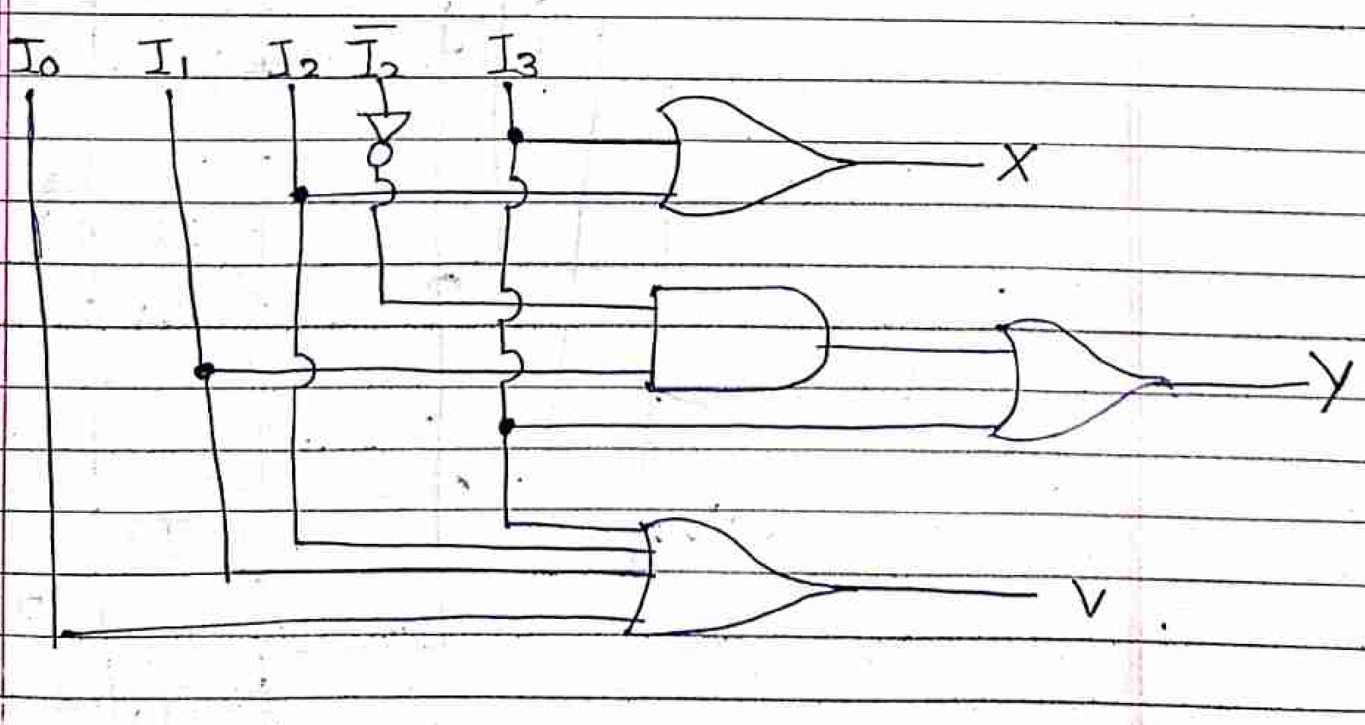
$I_1I_0 \backslash I_3I_2$	00	01	11	10
00	X 0	4	1 12	1 8
01	1	5	1 13	1 9
11	1 3	7	1 15	1 11
10	1 2	6	1 14	1 10

$$Y = I_3 + \overline{I_2} I_1$$

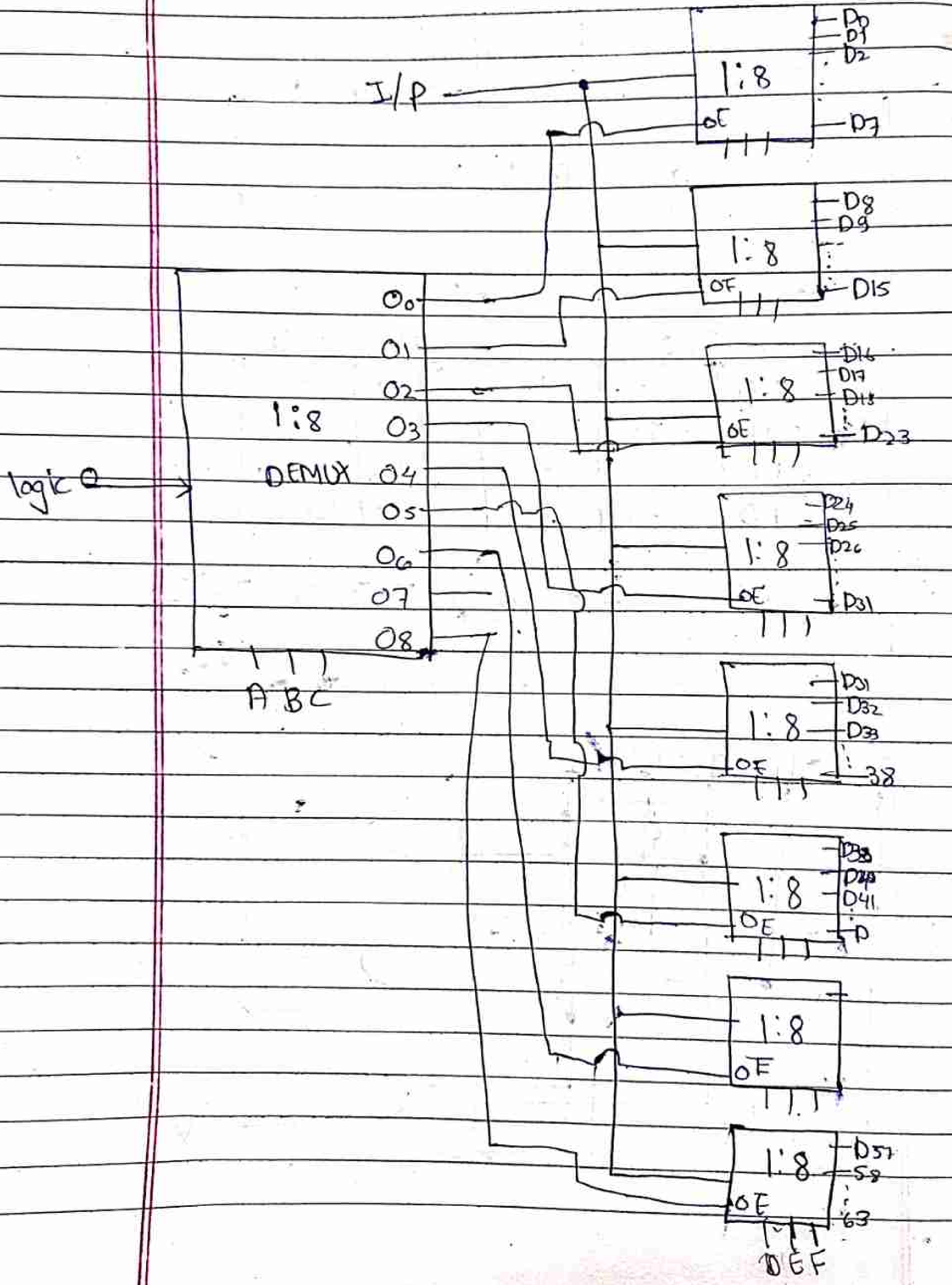
$I_1I_0 \backslash I_3I_2$	00	01	11	10
00	0	1 4	1 12	1 8
01	1 1	1 5	1 13	1 9
11	1 3	1 7	1 15	1 11
10	1 2	1 6	1 14	1 10

~~$V = I_3 + I_2 + \overline{I_2} I_1 + \overline{I_2} I_0$~~

$$V = I_3 + I_2 + I_1 + I_0$$



Que: 1:64 Demultiplexer using 1:8 DEMUX



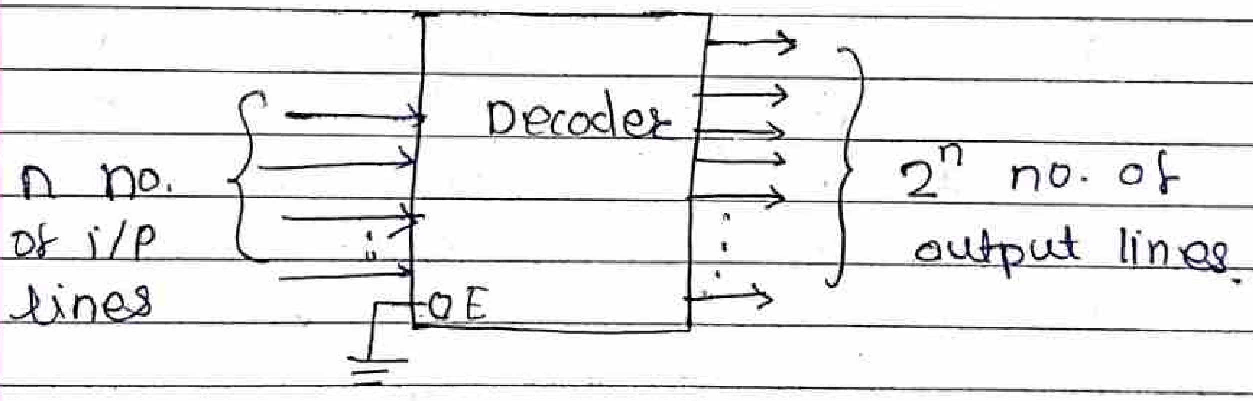
* Decoder :

It is a combinational ckt that converts binary information from N input line two maximum 2^n unquie ~~many~~ output lines

For each of these input combination only one of the n output line will be active(1) all other outputs will remain inactive (logic 0)

·AND gate or NAND gates are used to implement decoder.

Example of Decoder 3 line to 8 line decoder, 4 line to 16 line decoder, BCD to seven segment decoder



Que10. Implement the following using 3:8 decoder circuit.

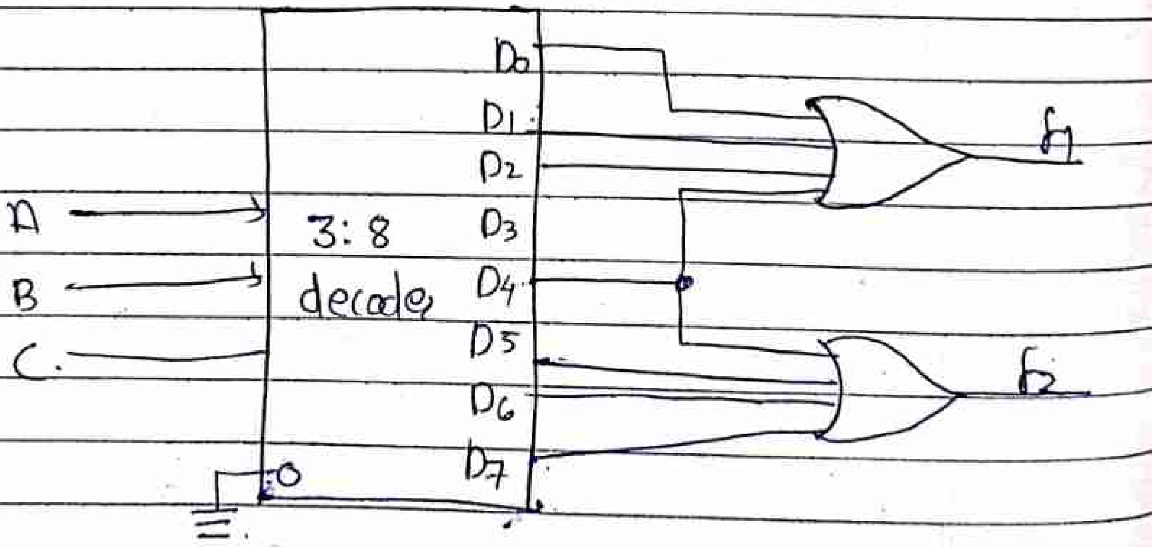
8-marks

$f_1(A,B,C) = \Sigma m(0,1,2,4)$

$f_2(A,B,C) = \Sigma m(4,5,6,7)$

→ Truth table :

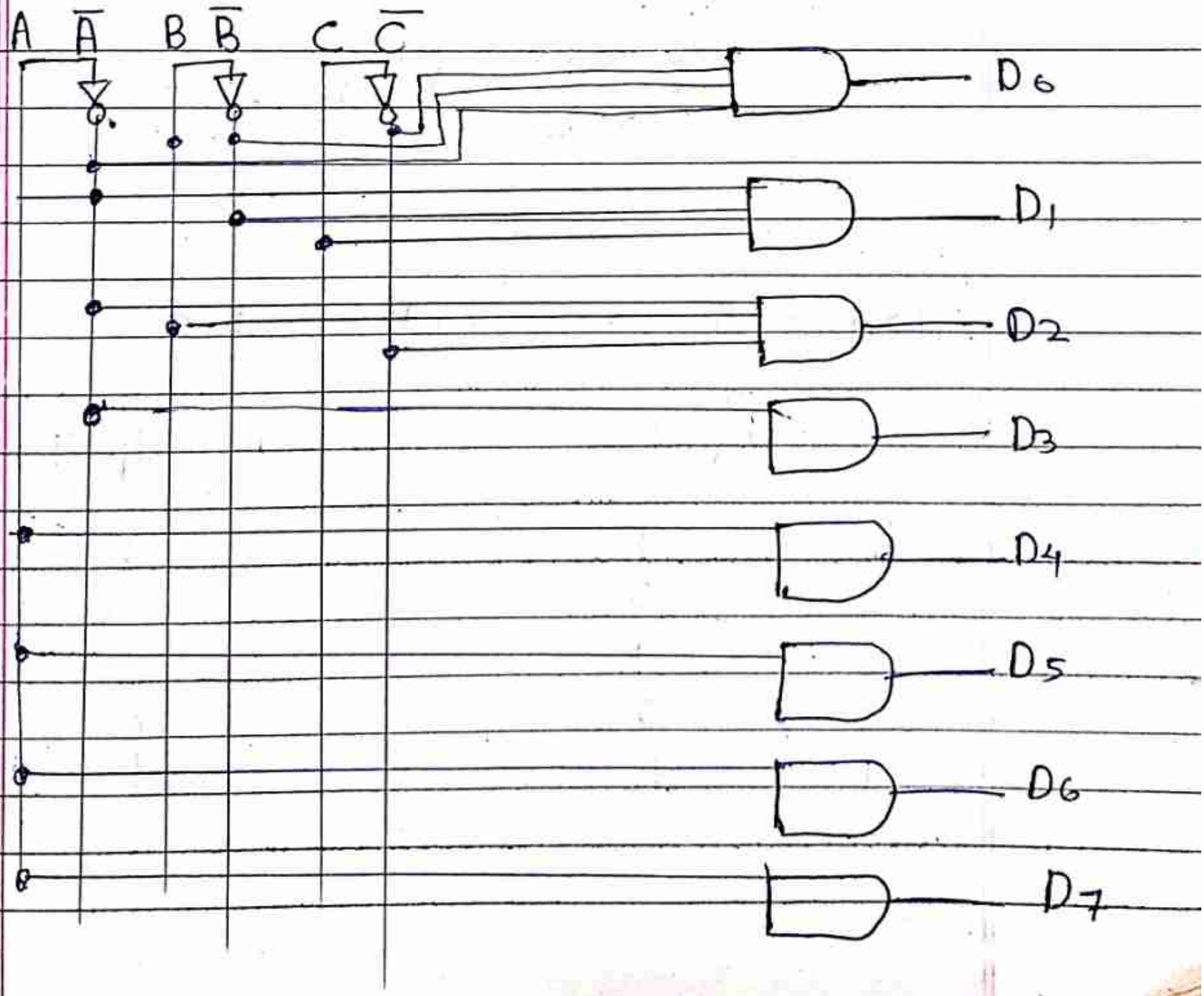
Inputs			outputs							
A	B	C	D_0	B_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



Que: Design 3 line to 8 line decoder

⟹ Truth Table

Input			outputs							
A	B	C	D0	D1	D2	D3	D4	D5	D6	D7
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


A
$\overline{A}$
B
$\overline{B}$
C
$\overline{C}$
D0
D1
D2
D3
D4
D5
D6
D7