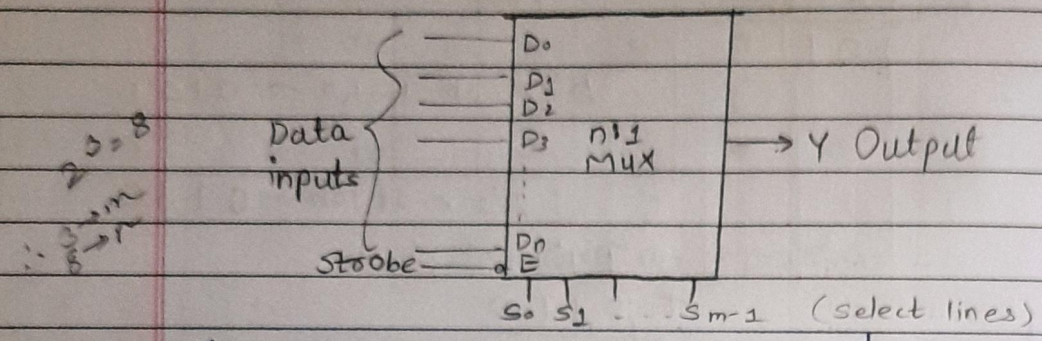
09-10-23

Page No.
Date

Unit 03:

* Multiplexer (MUX):



$2^3 = 8$

$\therefore 3 \rightarrow m$

- A multiplexer is a combinational ckt. that has 2^m input lines and a single output line

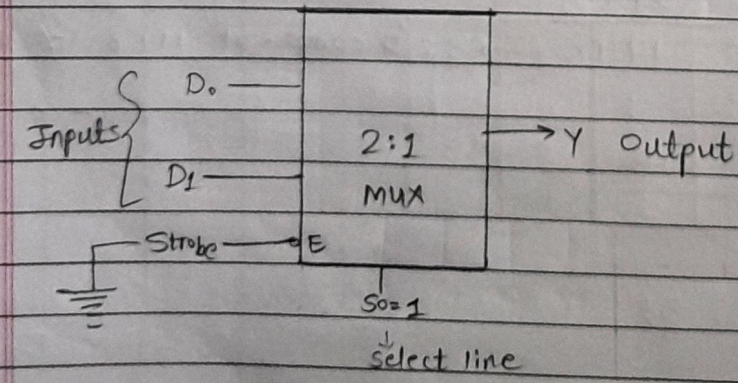
- The relationship between input and select lines is given by,

$$2^m = n$$

where, m → no. of select lines
n → no. of inputs

* Types of multiplexer:

1. 2:1 multiplexer:



$2 \rightarrow 2^1 : S_0$

There is inverter to strobe, so we ground it so that '0' will invert & become '1' & strobe enables (works) in '1' condn only

Strobe	Input	Select line S_0	Output
1	D_0	0	D_0
1	D_1	1	D_1

$Y = \overline{S_0} D_0 + S_0 D_1$

2. 4:1 multiplexer:

$4 \rightarrow 2^2 = 2$
S_0
S_1

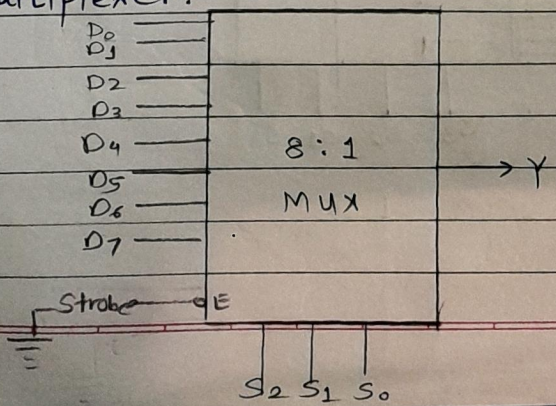
Strobe	Input	Select line		Output
		S_1	S_0	
1	D_0	0	0	D_0
1	D_1	0	1	D_1
1	D_2	1	0	D_2
1	D_3	1	1	D_3

$Y = \overline{S_1}\,\overline{S_0} D_0 + \overline{S_1} S_0 D_1 + S_1 \overline{S_0} D_2 + S_1 S_0 D_3$

$\overline{S_1}\,\overline{S_0} D_0$ → 0 0 D_0

3. 8:1 multiplexer:

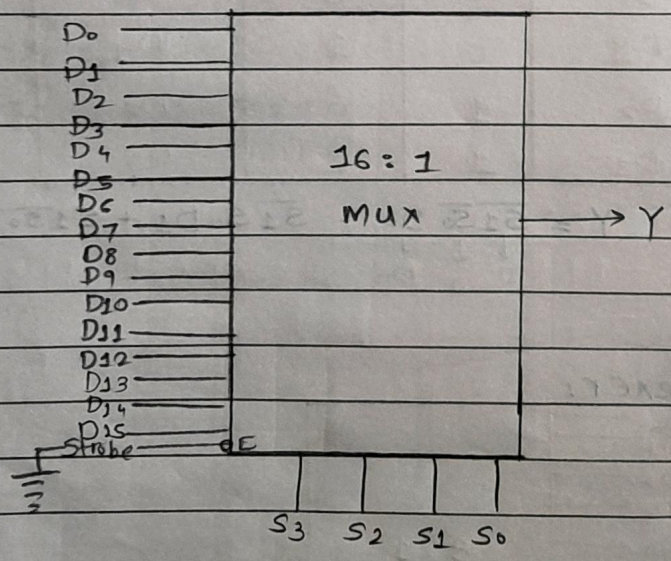
$2^3 = 8$



Strobe	Input	Select line			Output
		S_2	S_1	S_0	
1	D_0	0	0	0	D_0
1	D_1	0	0	1	D_1
1	D_2	0	1	0	D_2
1	D_3	0	1	1	D_3
1	D_4	1	0	0	D_4
1	D_5	1	0	1	D_5
1	D_6	1	1	0	D_6
1	D_7	1	1	1	D_7

$$Y = \bar{S_0}\bar{S_1}\bar{S_2}D_0 + S_0\bar{S_1}\bar{S_2}D_1 + \bar{S_0}S_1\bar{S_2}D_2 + S_0S_1\bar{S_2}D_3 + \bar{S_0}\bar{S_1}S_2D_4 + S_0\bar{S_1}S_2D_5 + \bar{S_0}S_1S_2D_6 + S_0S_1S_2D_7$$

4. 16 : 1 multiplexer

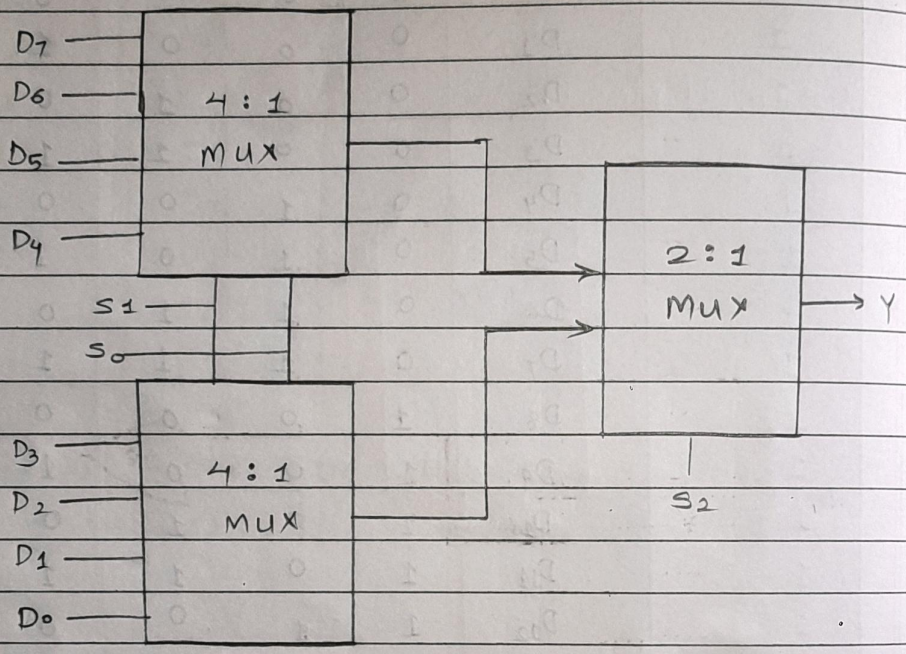


Strobe	Input	Select line				Output
		S_3	S_2	S_1	S_0	
1	D_0	0	0	0	0	D_0
1	D_1	0	0	0	1	D_1
1	D_2	0	0	1	0	D_2
1	D_3	0	0	1	1	D_3
1	D_4	0	1	0	0	D_4
1	D_5	0	1	0	1	D_5
1	D_6	0	1	1	0	D_6
1	D_7	0	1	1	1	D_7
1	D_8	1	0	0	0	D_8
1	D_9	1	0	0	1	D_9
1	D_{10}	1	0	1	0	D_{10}
1	D_{11}	1	0	1	1	D_{11}
1	D_{12}	1	1	0	0	D_{12}
1	D_{13}	1	1	0	1	D_{13}
1	D_{14}	1	1	1	0	D_{14}
1	D_{15}	1	1	1	1	D_{15}

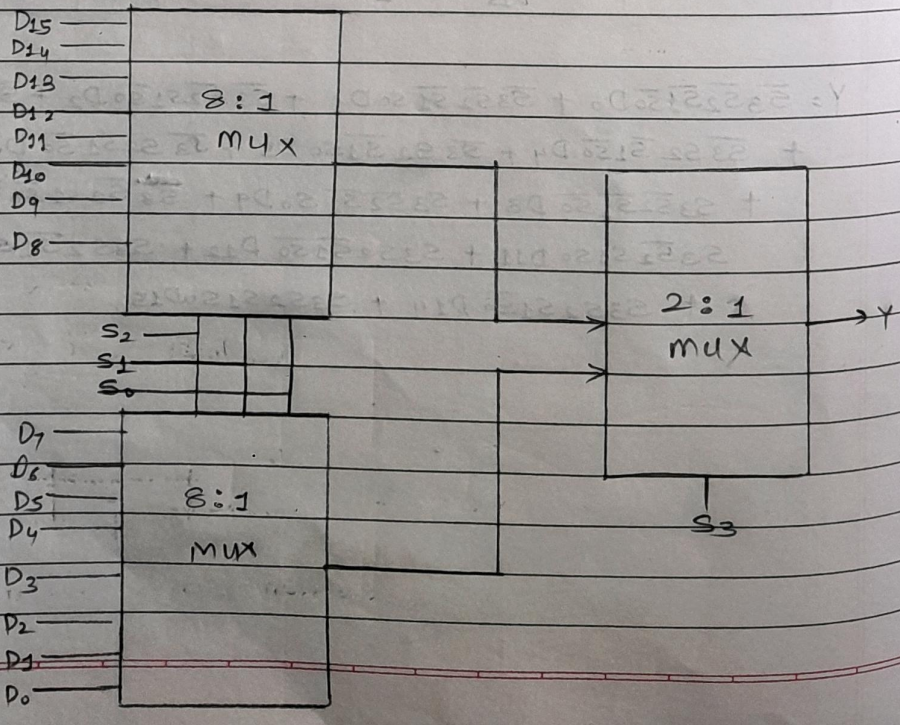
$$Y = \bar{S_3}\bar{S_2}\bar{S_1}\bar{S_0}D_0 + \bar{S_3}\bar{S_2}\bar{S_1}S_0D_1 + \bar{S_3}\bar{S_2}S_1\bar{S_0}D_2 + \bar{S_3}\bar{S_2}S_1S_0D_3 + \bar{S_3}S_2\bar{S_1}\bar{S_0}D_4 + \bar{S_3}S_2\bar{S_1}S_0D_5 + \bar{S_3}S_2S_1\bar{S_0}D_6 + \bar{S_3}S_2S_1S_0D_7 + S_3\bar{S_2}\bar{S_1}\bar{S_0}D_8 + S_3\bar{S_2}\bar{S_1}S_0D_9 + S_3\bar{S_2}S_1\bar{S_0}D_{10} + S_3\bar{S_2}S_1S_0D_{11} + S_3S_2\bar{S_1}\bar{S_0}D_{12} + S_3S_2\bar{S_1}S_0D_{13} + S_3S_2S_1\bar{S_0}D_{14} + S_3S_2S_1S_0D_{15}$$

Page No.
Date

8:1 multiplexer using 4:1 and 2:1 multiplexer



16:1 multiplexer using 8:1 and 2:1 multiplexer



10-10-23

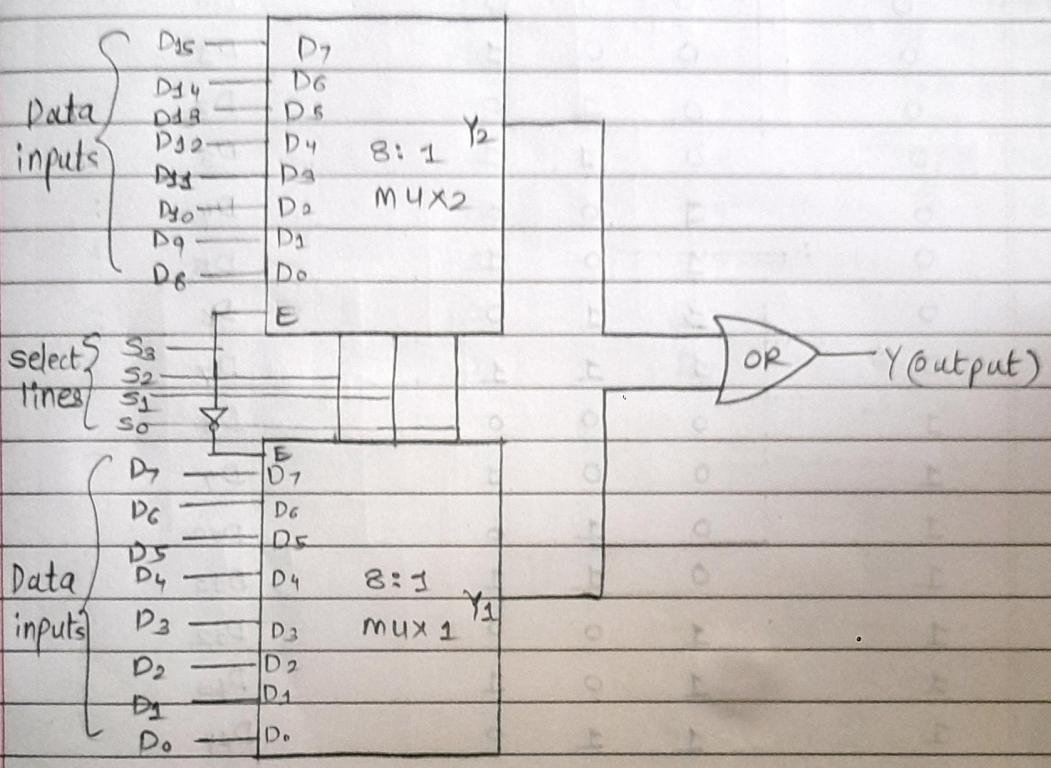
Explanation of 3-4 lines may be asked in exam → Refer unofficial grp 09-10-23 (photos)

Page No.
Date

* Multiplexer tree: Here S3 ≡ E

1. To design 16:1 MUX using 8:1 MUX

→ When $S_3 = 0$, MUX 1 is enabled & when $S_3 = 1$ MUX 2 is enabled



For 8:1 → IC → 74151

If NOT gate to MUX2, then D0 to D7 to for MUX2

select lines

16:1 → 4

8:1 → 3

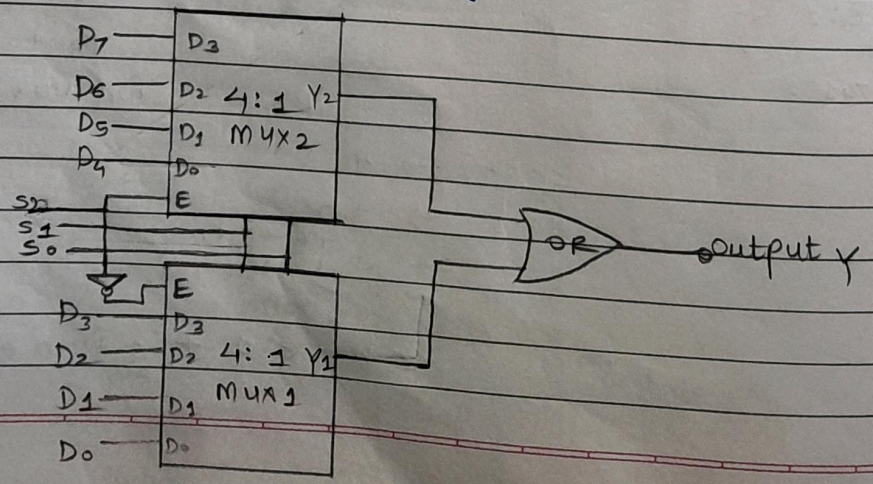
8:1 → 3 } 6 → not required → we req. only 4.

∴ 3 from 8:1 + 1 from enable E

Truth table:

E → ≡ S3	Select lines S_2	S_1	S_0	Output Y
0	0	0	0	D_0
0	0	0	1	D_1
0	0	1	0	D_2
0	0	1	1	D_3
0	1	0	0	D_4
0	1	0	1	D_5
0	1	1	0	D_6
0	1	1	1	D_7
1	0	0	0	D_8
1	0	0	1	D_9
1	0	1	0	D_{10}
1	0	1	1	D_{11}
1	1	0	0	D_{12}
1	1	0	1	D_{13}
1	1	1	0	D_{14}
1	1	1	1	D_{15}

2. To design 8:1 MUX using 4:1 MUX



Page No.
Date

Truth table:

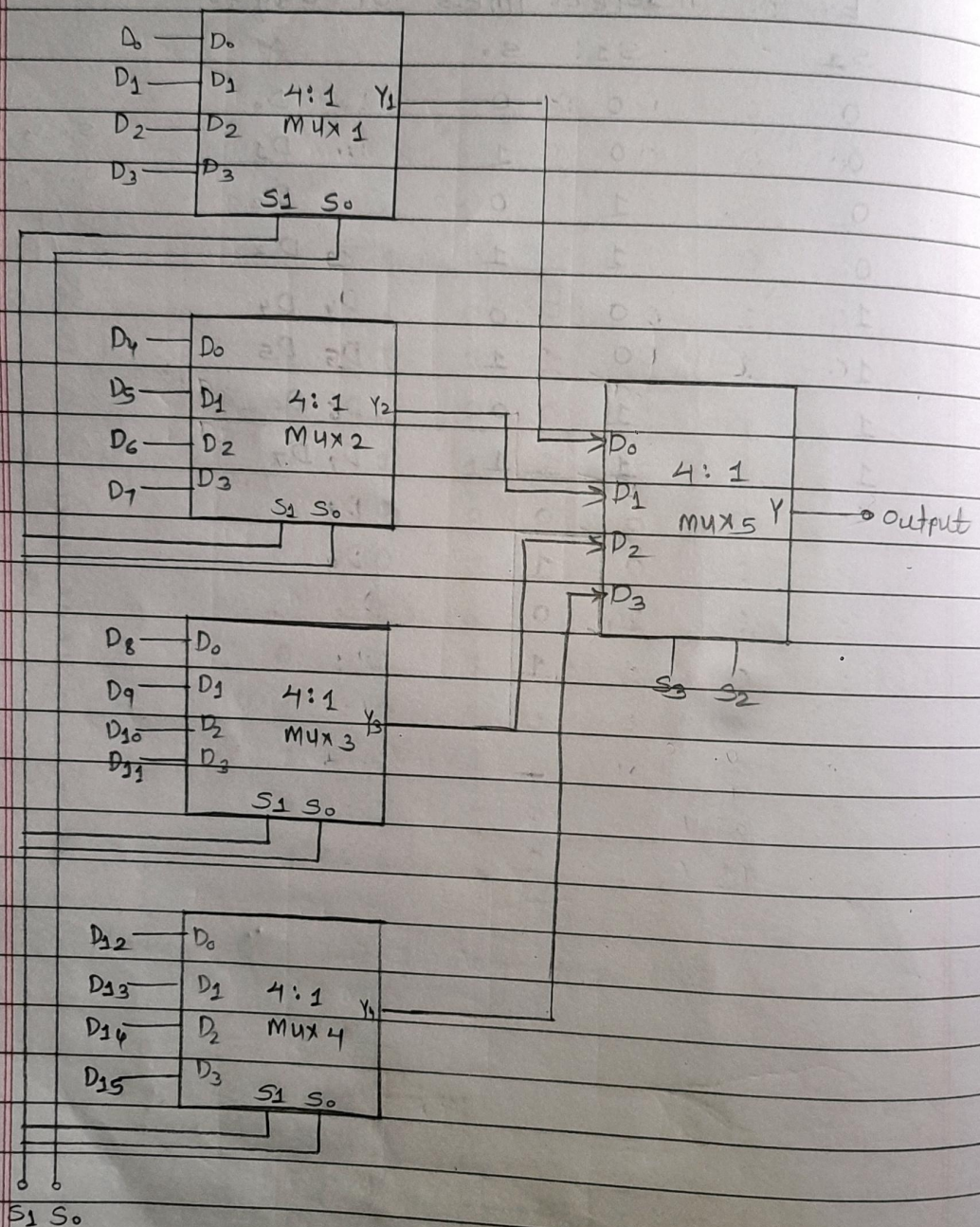
E	Select lines		Output
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

When S_2, S_1, S_0 are 0, 0, 0, then o/p is first i/p D_0 & so on...

Strobe can be either 1 or 0
If IC's are more than 2 for cascading, then we can use strobe

Page No.
Date

3. To design 16:1 MUX by using 4:1 MUX



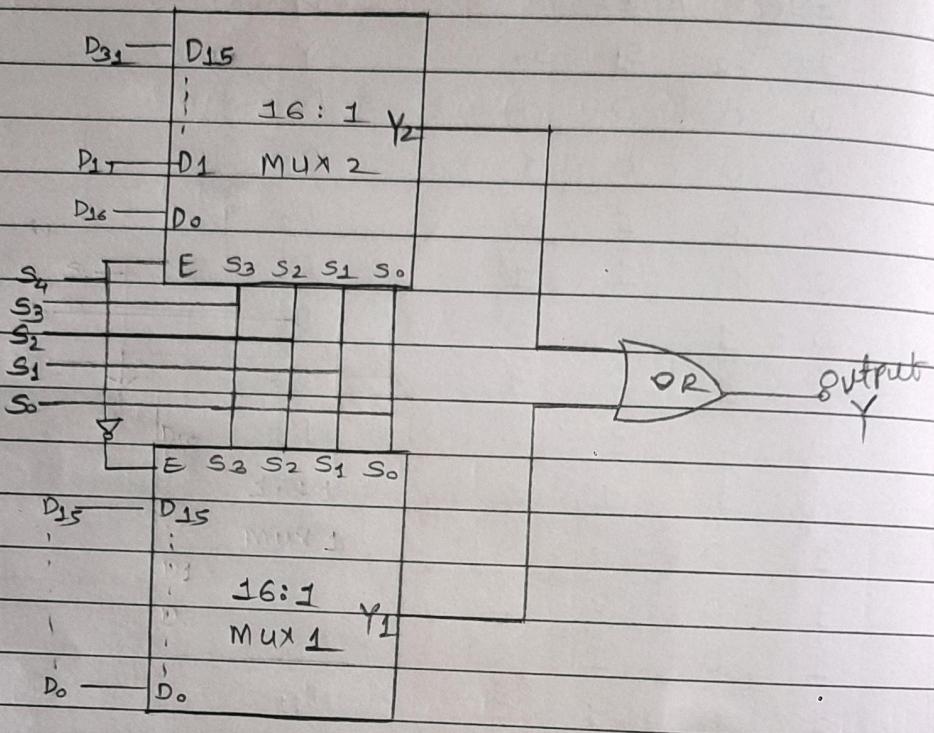
Page No.
Date

Truth table:

Select lines				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	D_0
0	0	0	1	D_1
0	0	1	0	D_2
0	0	1	1	D_3
0	1	0	0	D_4
0	1	0	1	D_5
0	1	1	0	D_6
0	1	1	1	D_7
1	0	0	0	D_8
1	0	0	1	D_9
1	0	1	0	D_{10}
1	0	1	1	D_{11}
1	1	0	0	D_{12}
1	1	0	1	D_{13}
1	1	1	0	D_{14}
1	1	1	1	D_{15}

H.W

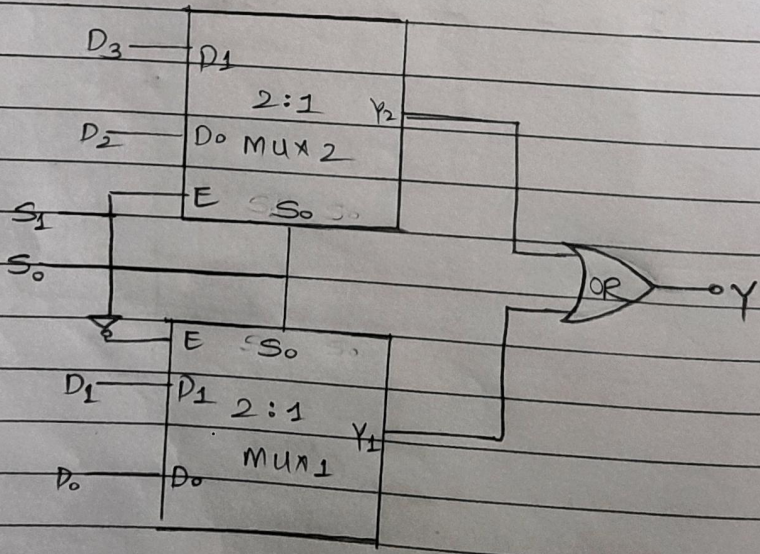
4. To design 32:1 Mux by using 16:1 Mux.



Truth table & circuit

H.W

5. To design 4:1 Mux by using 2:1 Mux



Page No.
Date

Truth table:

E	Select line	Output	
S_1	S_0	Y	
0	0	D_0	} MUX-1 selected
0	1	D_1	
1	0	D_2	} MUX-2 selected
1	1	D_3	

1-10-23

Page No.
Date

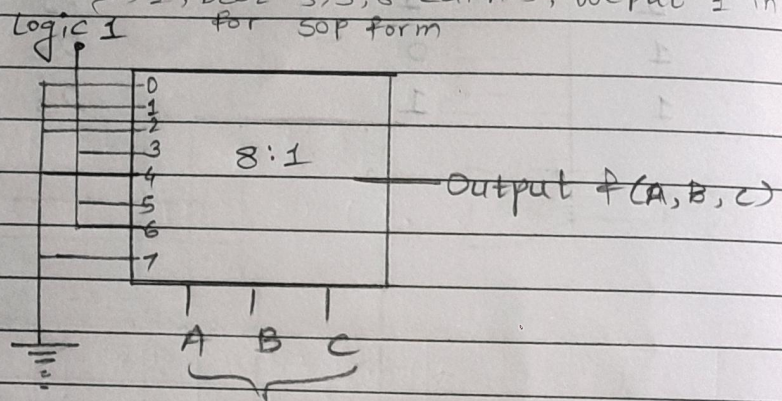
* Implementation of boolean function using Mux:

SM

Q. Implement the following fucⁿ using 8:1 Mux.

$$f(A, B, C) = \Sigma m(3,5,6)$$

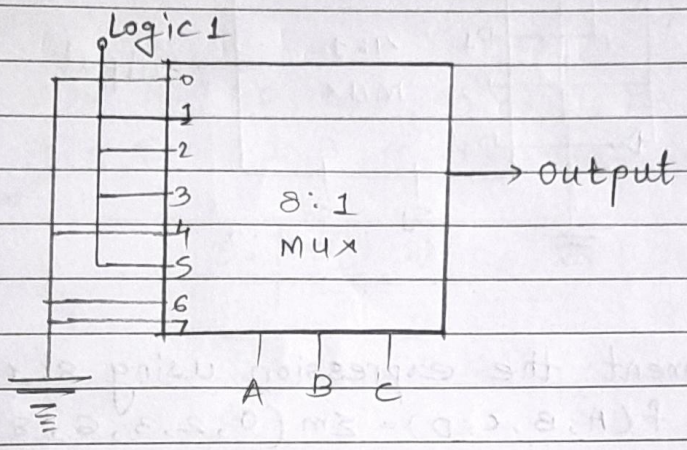
→ 1, bcoz 3,5,6 cell nos, we put 1 in kmap for SOP form



Q. Implement the boolean fucⁿ given by truth table using multiplexer.

	A	B	C	Y	
0	0	0	0	0	
1	0	0	1	1	→ 1
2	0	1	0	1	→ 2
3	0	1	1	1	→ 3
4	1	0	0	0	
5	1	0	1	1	→ 5
6	1	1	0	0	
7	1	1	1	0	

$\therefore \quad Y = \Sigma m(1, 2, 3, 5)$



Q. Implement the boolean function using 4:1 MUX.

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

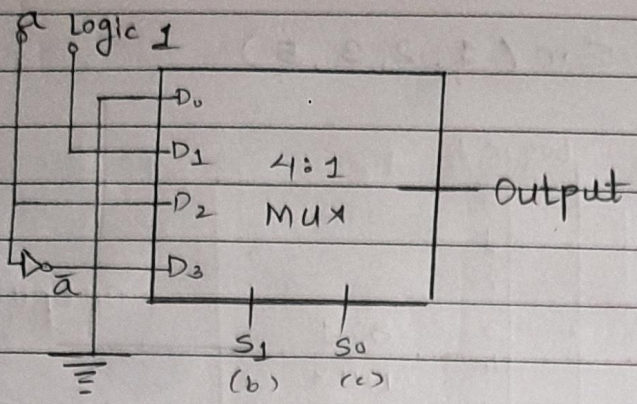
↳ This can be implemented by using MUX tree also.

MSB ← a b c → LSB

↳ by using 4:1, we have only 2 select lines b & c

for a we'll make a table as follows:

	D_0	D_1	D_2	D_3
$\bar{a}$	0	(1)	2	(3)
a	4	(5)	(6)	7
Input to MUX	0	1	a	$\bar{a}$

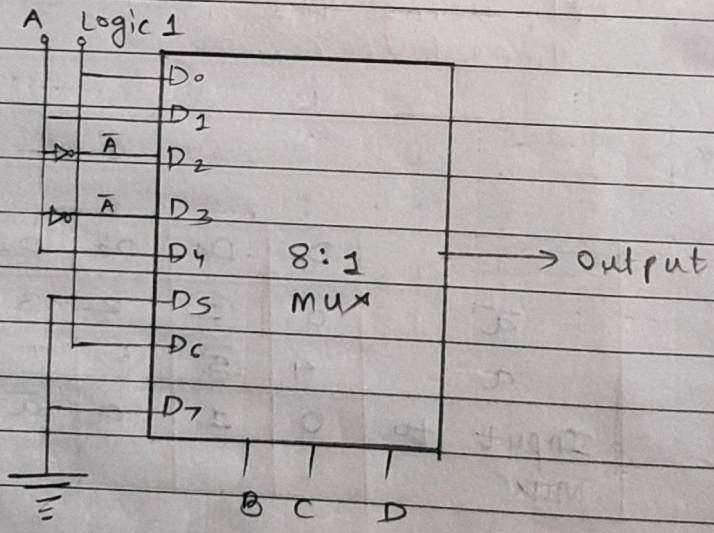


Q. Implement the expression using 8:1 MUX.

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

$\Rightarrow$

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	1	(2)	(3)	4	5	(6)	7
A	(8)	(9)	10	11	(12)	13	(14)	15
I/P to MUX	1	A	$\bar{A}$	$\bar{A}$	A	0	1	0



HW Q. Implement full adder using 8:1 mux

HW Q. Design the full subtractor using 4:1 mux.
↳ Cascading

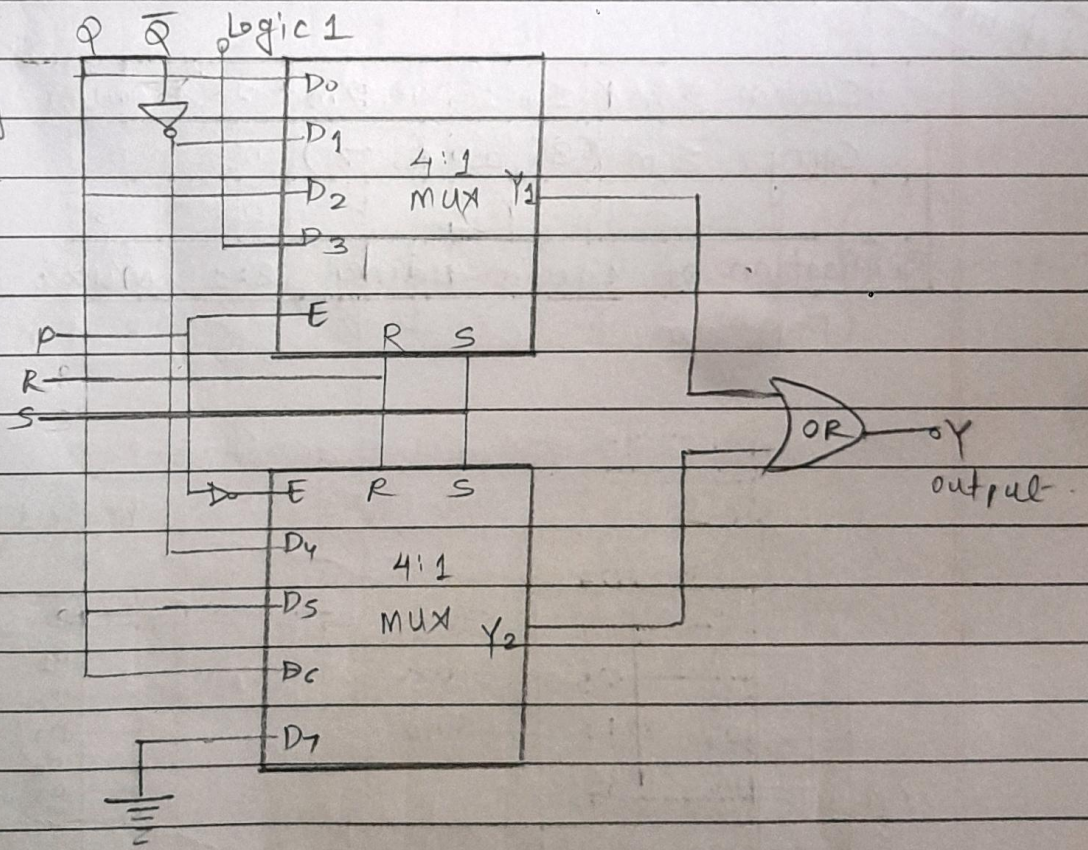
Page No.
Date

Q. $f(P,Q,R,S) = \pi M(0,2,5,6,7,9,12,15)$
Implement the boolean equation by using 4:1 MUX

⇒ $f(P,Q,R,S) = \pi M(0,2,5,6,7,9,12,15)$
$f(P,Q,R,S) = \Sigma m(1,3,4,8,10,11,13,14)$

Here, we did cascading of MUX bcoz, 4:2 → 2 select lin. P,Q,R,S → 4 select lin. ∴ P & Q are remaining. If only Q is remaining then no need to cascading.

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{Q}$	0	(1)	2	(3)	(4)	5	6	7
Q	(8)	9	(10)	(11)	12	(13)	(14)	15
I/P to MUX	Q	$\bar{Q}$	Q	1	$\bar{Q}$	Q	Q	0



Page No.
Date

M.U

9. Implement full adder using 8:1 MUX.

	Inputs A	B	Cin	Outputs Sum	Carry
D_0	0	0	0	0	0
D_1	0	0	1	1 → 1	0
D_2	0	1	0	1 → 2	0
D_3	0	1	1	0	1 → 3
D_4	1	0	0	1 → 4	0
D_5	1	0	1	0	1 → 5
D_6	1	1	0	0	1 → 6
D_7	1	1	1	1 → 7	1 → 7

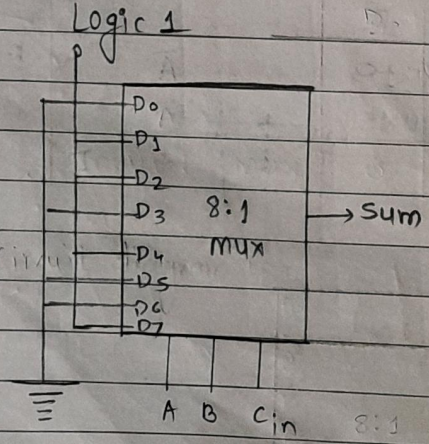
8:1 → 3 selection lin.
A, B, Cin → 3 selection lin.
∴ No need of cascading, no need of table.

Sum: $\Sigma m(1, 2, 4, 7)$

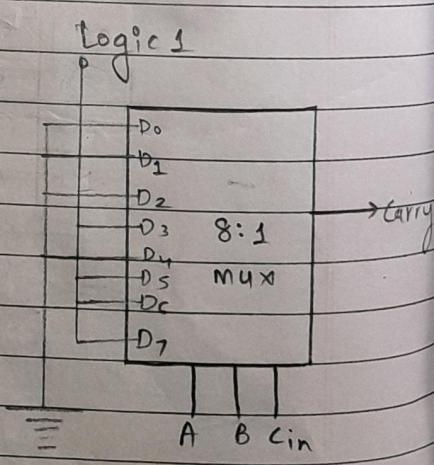
Carry: $\Sigma m(3, 5, 6, 7)$

Realisation of circuit using 8:1 MUX:

For sum



For carry



2 ways → reduction method
→ cascading

Page No.
Date

Q. Design the full subtractor using 4:1 MUX

4:1 → 2 select lin
A B Bin → 3 select lin
↓
Table

	Inputs			Outputs		
	A	B	B_{in}	Diff.		Borrow
0	0	0	0	0	D_0	0
1	0	0	1	1	D_1	1 D_1
2	0	1	0	1	D_2	1 D_2
3	0	1	1	0	D_3	1 D_3
4	1	0	0	1	D_4	0
5	1	0	1	0	D_5	0
6	1	1	0	0	D_6	0
7	1	1	1	1	D_7	1 D_7

(Total = 8)

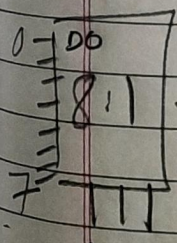
	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
I/P to Mux	A	$\bar{A}$	$\bar{A}$	A

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

8:1

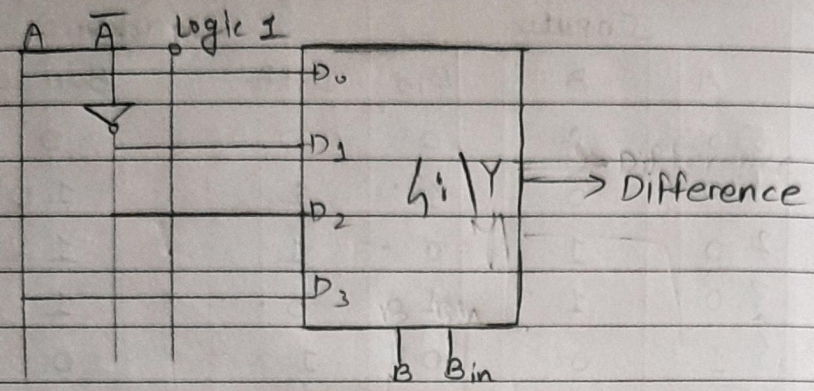
	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
I/P to Mux	0	$\bar{A}$	$\bar{A}$	1

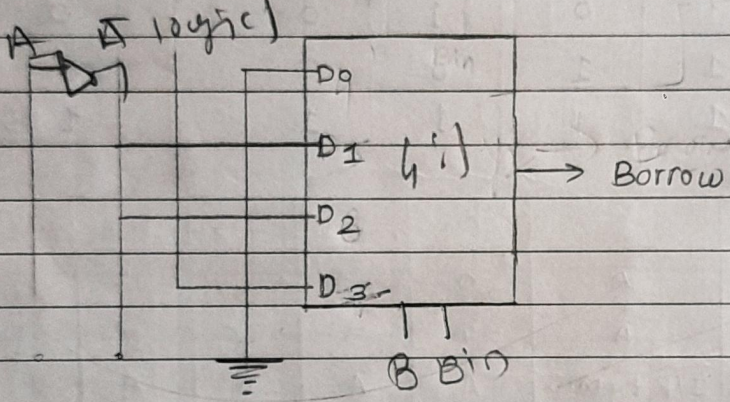
Borrow = $\Sigma m(1,2,3,7)$



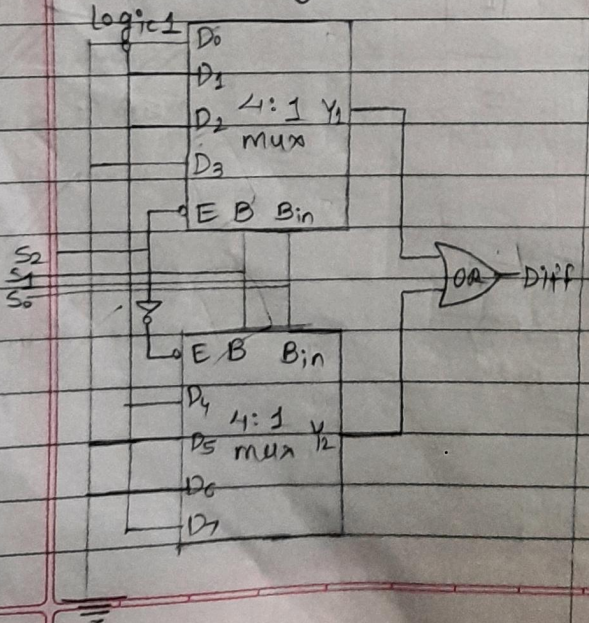
↓ reduction method

Realisation of circuit using 4:1 MUX by using reduction method:





Alternate way: Cascading method



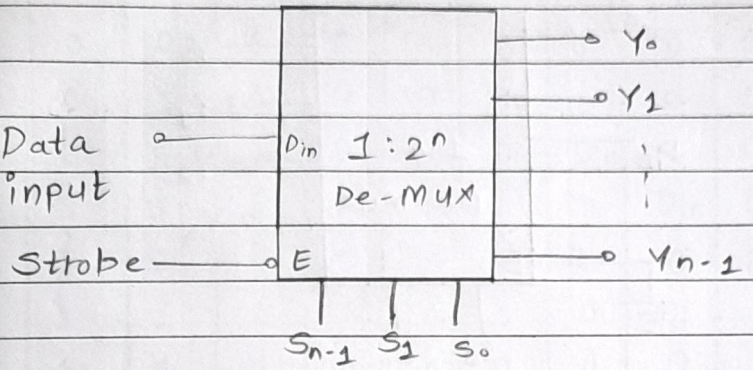
Same for borrow

16-10-23

5-6 theory may be asked on De-Mux & types.

Page No.
Date

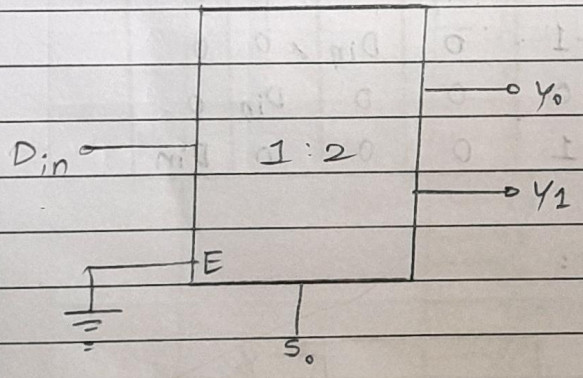
De-multiplexer:



Theory (5-6 lines)

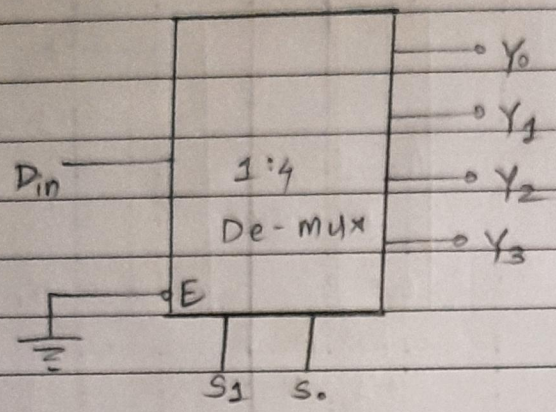
Type of De-Mux:

1) 1 : 2 De-Mux:



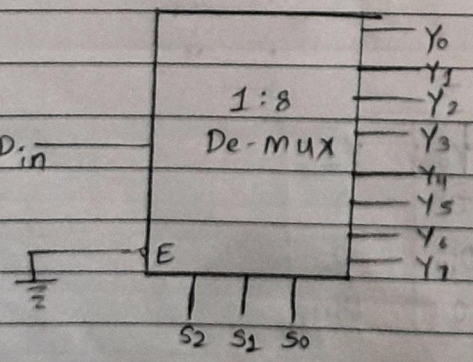
E	S0	Y0	Y1
0	X	0	0
1	0	Din	0
1	1	0	Din

2) 1:4 De-Mux :



E	S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	X	X	0	0	0	0
1	0	0	Din	0	0	0
1	0	1	0	Din	0	0
1	1	0	0	0	Din	0
1	1	1	0	0	0	Din

3) 1:8 De-Mux :

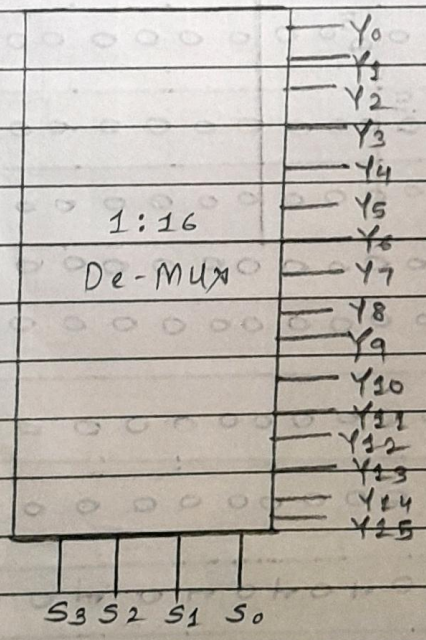


1) 1:8 Demux
2) 1:16 Demux

Page No.
Date

Inputs				Outputs							
E	S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	Din	0	0	0	0	0	0	0
1	0	0	1	0	Din	0	0	0	0	0	0
1	0	1	0	0	0	Din	0	0	0	0	0
1	0	1	1	0	0	0	Din	0	0	0	0
1	1	0	0	0	0	0	0	Din	0	0	0
1	1	0	1	0	0	0	0	0	Din	0	0
1	1	1	0	0	0	0	0	0	0	Din	0
1	1	1	1	0	0	0	0	0	0	0	Din

HW
4) 1:16 De-MUX :

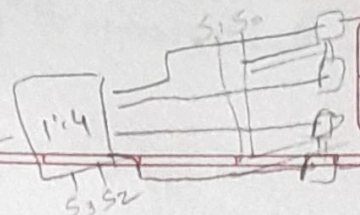


E	S_3	S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}
0	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	Din	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	Din	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	Din	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	Din	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	Din	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	Din	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	Din	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	Din	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	Din	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	Din	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	Din	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	Din	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Din	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	Din	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Din	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Din

So on

$\rightarrow Y_0 = E\,\overline{S_3}\,\overline{S_2}\,\overline{S_1}\,\overline{S_0}\,Din$

H.W 1) Implement 1:16 De-MUX by using 1:4 DeMUX

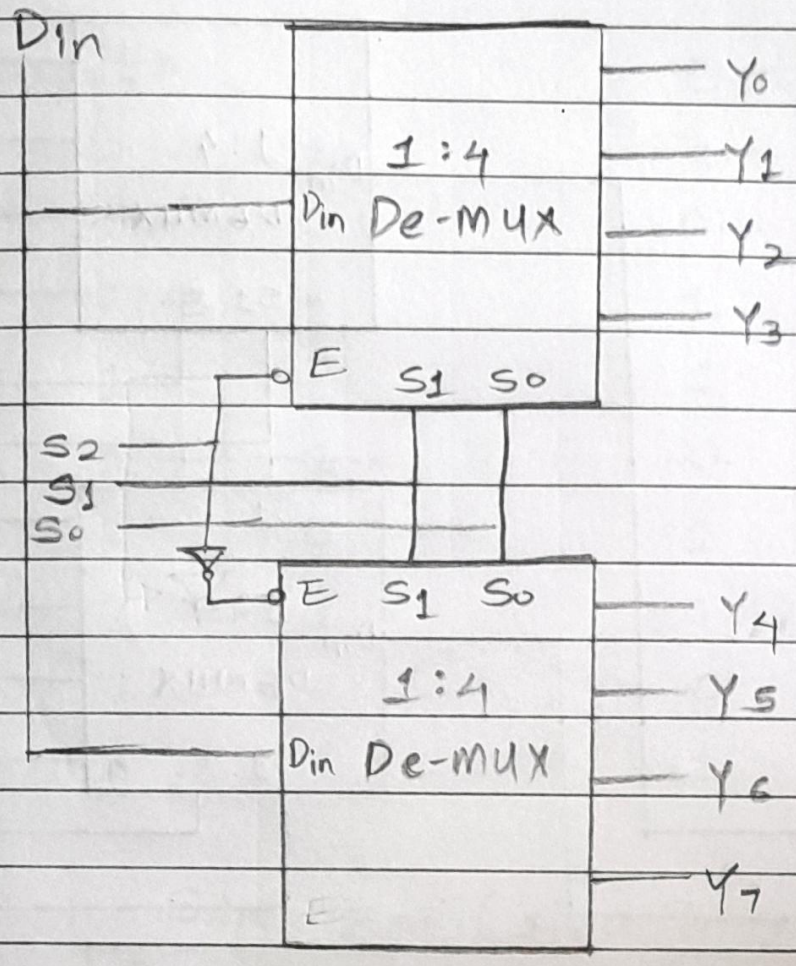


Page No.

Date

DEMUX TREE:

Implement 1:8 De-MUX by using 1:4 De-MUX



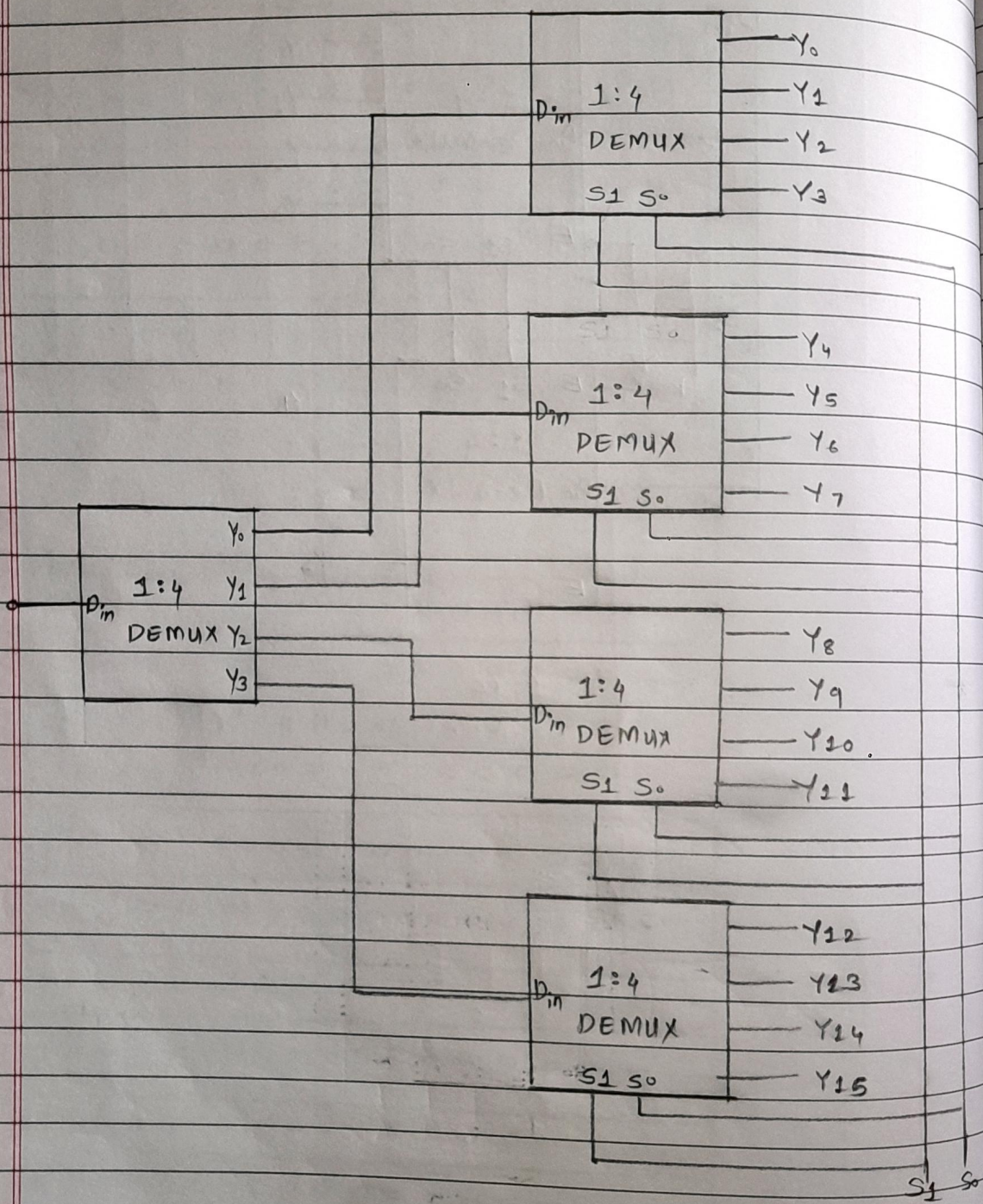
Due in exam may be 1:4, 1:8 apki bhi.

2) Design & implement full adder by using 1:8 De-mux

3) Design & Implement full subtractor by using 1:8 De-mux

Page No. De-MUX
Date

Implement 1:16 De-Mux by using 1:4 De-Mux



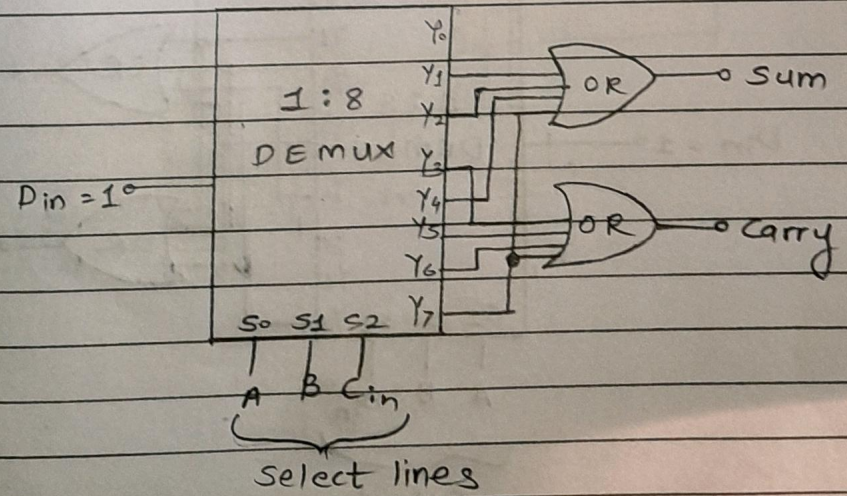
Que in exam full adder can be using 1:4 or 1:8 or apti Demux

Design and implement full adder by using 1:8 DeMux :

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

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

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

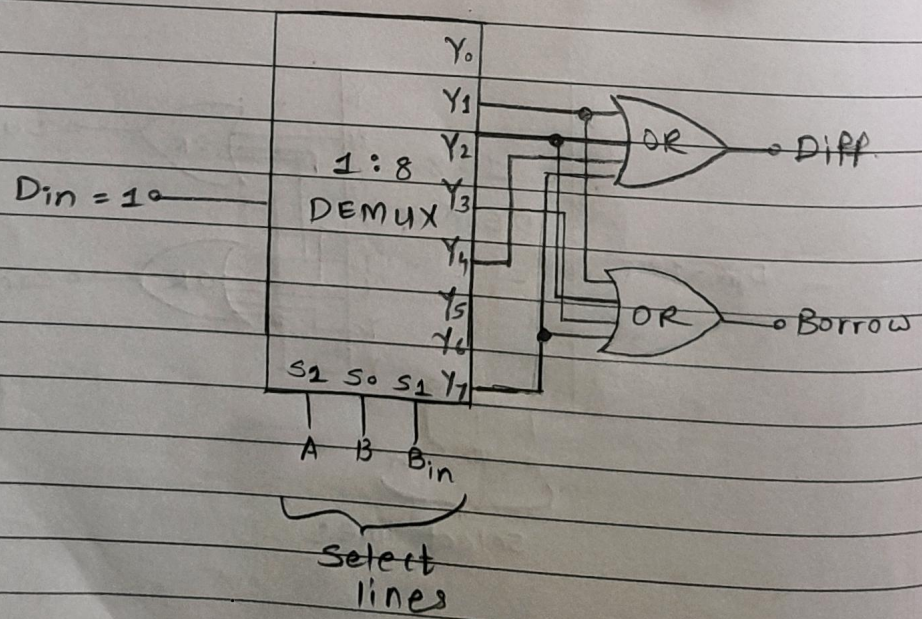


H.W *

Design and implement full subtractor using appropriate demultiplexer:

Input			Output	
A	B	Bin	Difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Difference = $\Sigma m(1,2,4,7)$; Borrow = $\Sigma m(1,2,3,$

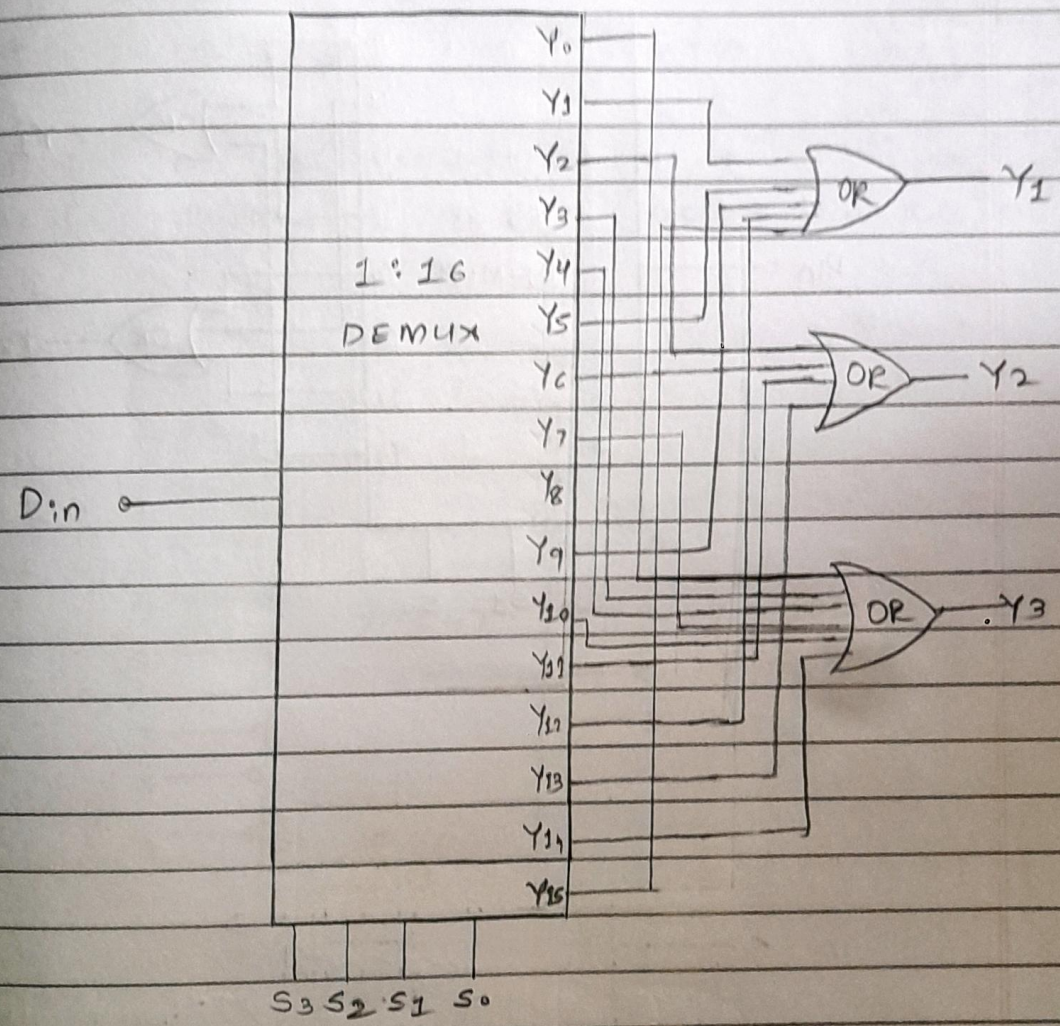


Page No.
Date

Q. $Y_1(ABCD) = \Sigma m(1, 5, 9, 12, 15)$

$Y_2(ABCD) = \Sigma m(2, 6, 11, 13)$

$Y_3(ABCD) = \Sigma m(0, 3, 4, 7, 10, 14)$

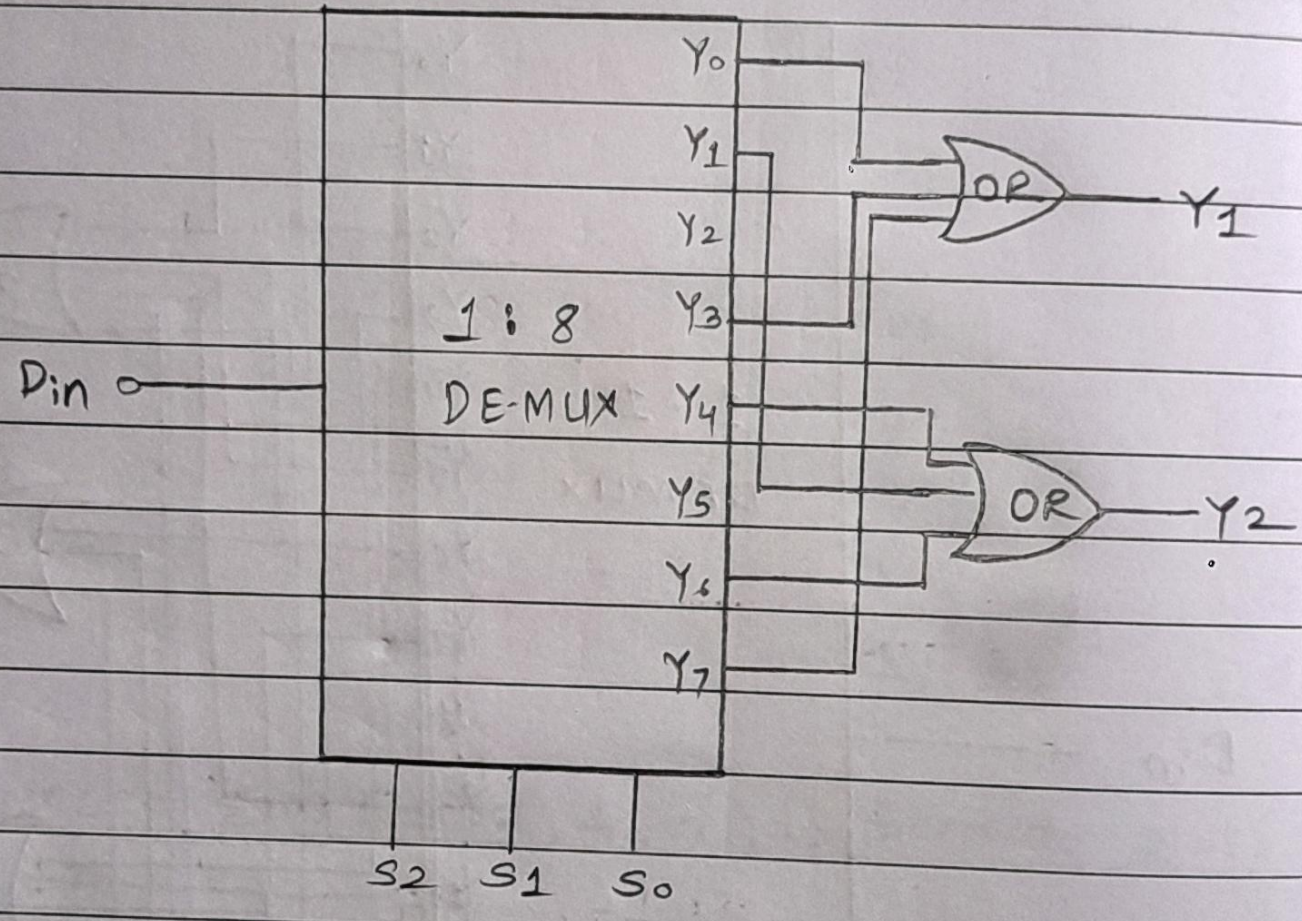


Page No.
Date

Q. $Y_1(ABC) = \Sigma m(0,3,7)$

$Y_2(ABC) = \Sigma m(1,4,6)$

$\Longrightarrow$

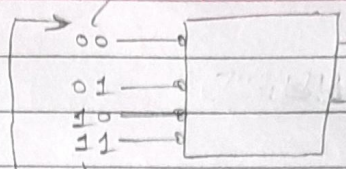


2^n → input lines
n → output

Page No.
Date

Same logic for all i/Ps

→ Lowest priority



→ Highest priority

when 01 i/p is selected
↓
Status of 10 & 11 is '0' bcoz they've higher priority than 01
↓
status of 00 is don't care (x) as it has less priority than 01

This means if you give input '0' then it'll invert & become '1' and vice-versa

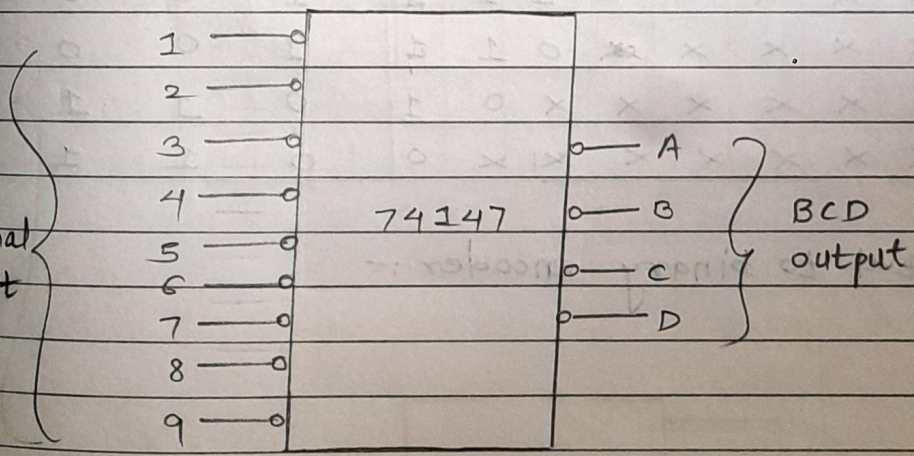
* Encoders:

* Types of encoders:

1. Priority encoder
2. Decimal to BCD encoder
3. Octal to Binary encoder
4. Hexadecimal to Binary encoder.

* Decimal to BCD encoder:-

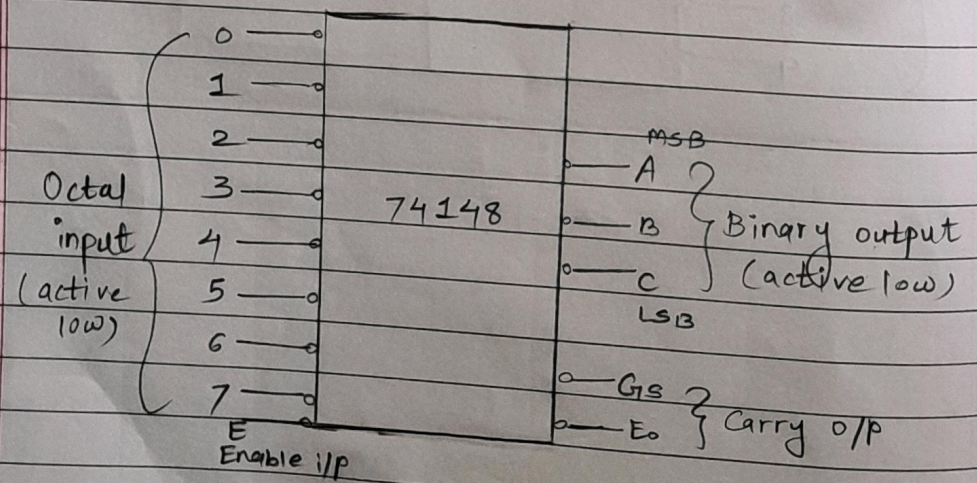
↳ IC required : 74147 (active low IC)



Truth table:

Active low decimal i/p									Active low BCD o/p			
1	2	3	4	5	6	7	8	9	A	B	C	D
1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0
x	0	1	1	1	1	1	1	1	1	1	0	1
x	x	0	1	1	1	1	1	1	1	1	0	0
x	x	x	0	1	1	1	1	1	1	0	1	1
x	x	x	x	0	1	1	1	1	1	0	1	0
x	x	x	x	x	0	1	1	1	1	0	0	1
x	x	x	x	x	x	0	1	1	1	0	0	0
x	x	x	x	x	x	x	0	1	0	1	1	1
x	x	x	x	x	x	x	x	0	0	1	1	0

Octal to binary encoder :-



→ when enable=1, it'll invert & becomes '0'. ∴ all one don't care o/p should be '0', but it'll also invert & become '1'

So, to enable for works, we should give it '0' which will become '1'

Page No.
Date

Truth table:

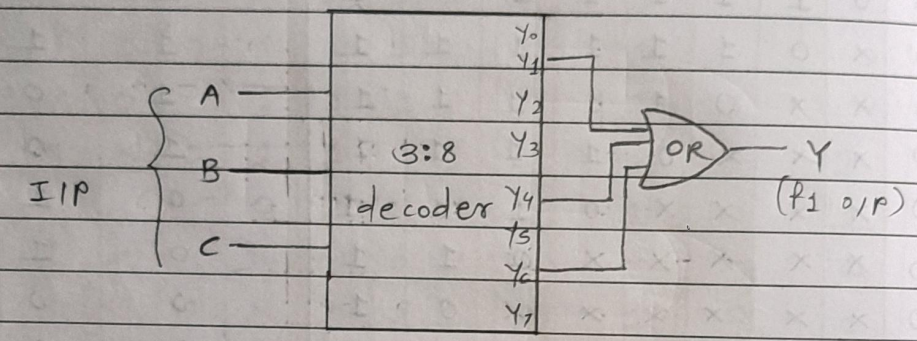
Active low octal i/p									Active low Binary o/p		
E	0	1	2	3	4	5	6	7	A	B	C
1	x	x	x	x	x	x	x	x	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1
0	x	0	1	1	1	1	1	1	1	1	0
0	x	x	0	1	1	1	1	1	1	0	1
0	x	x	x	0	1	1	1	1	1	0	0
0	x	x	x	x	0	1	1	1	0	1	1
0	x	x	x	x	x	0	1	1	0	1	0
0	x	x	x	x	x	x	0	1	0	0	1
0	x	x	x	x	x	x	x	0	0	0	0

23-10-23

Page No.
Date

* Decoders:

Q. Implement the following boolean function:
$f_1(A\,B\,C) = \Sigma m(1, 4, 6)$ using 3:8 decoder:



Q. Design and implement 3-bit binary to grey code converter by using suitable decoder.

No. of inputs: 03 = B_2, B_1, B_0
No. of outputs: 03 = G_2, G_1, G_0

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

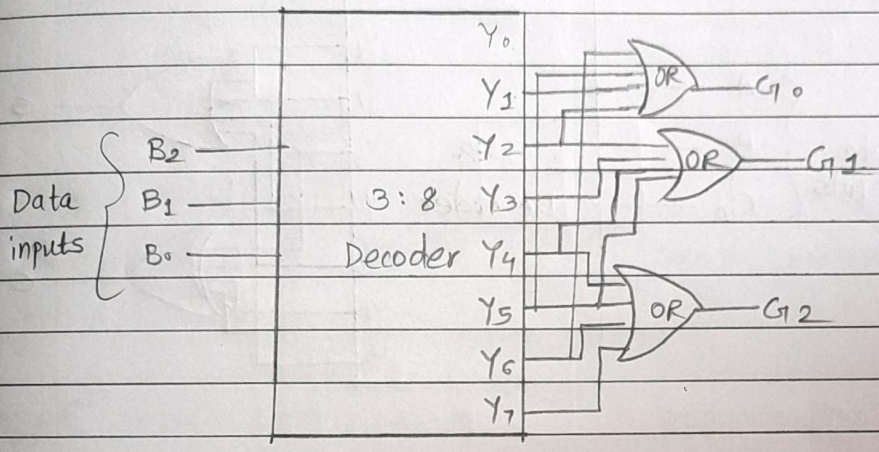
Page No.
Date

Final boolean expression for output:

$G_2(B_2, B_1, B_0) = \Sigma m(4,5,6,7)$

$G_1(B_2, B_1, B_0) = \Sigma m(2,3,4,5)$

$G_0(B_2, B_1, B_0) = \Sigma m(1,2,5,6)$



Q. Design and implement full adder by using suitable decoder.

⇒ No. of inputs : 03 = A, B, Cin

No. of outputs: 02 = S(sum), C(carry)

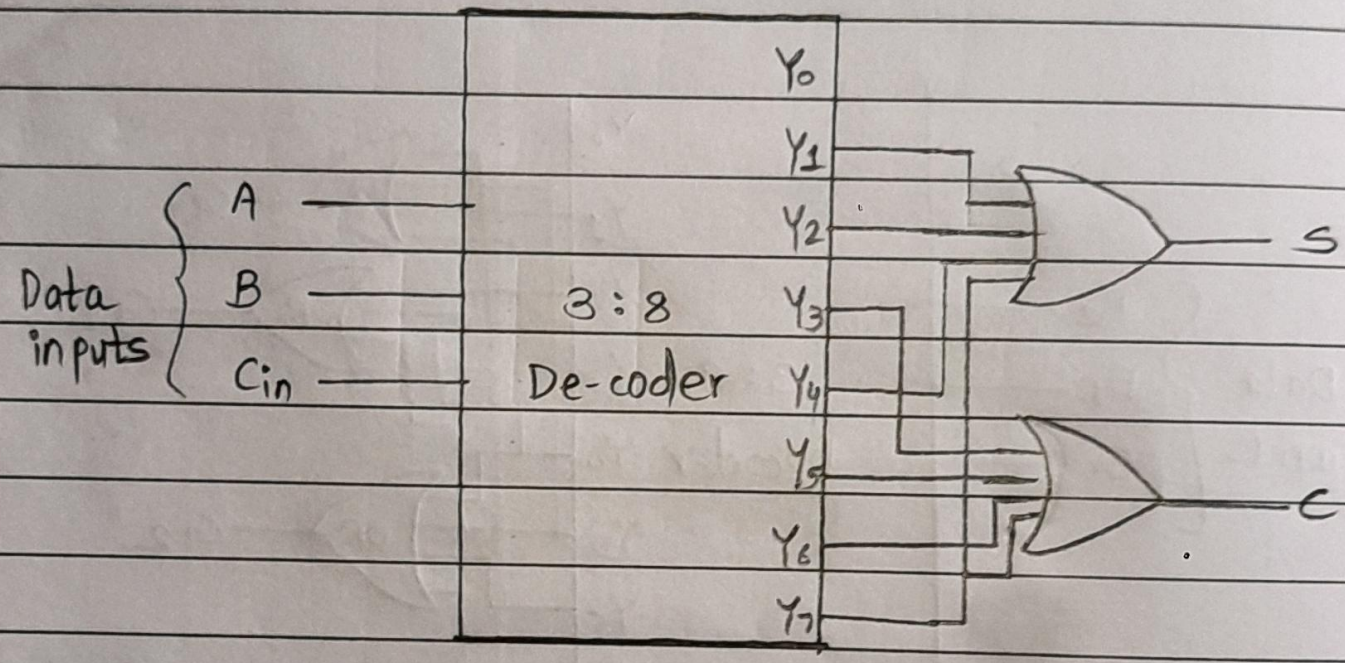
	Inputs			Outputs	
	A	B	Cin	sum (S)	Carry (C)
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

Page No.
Date

Final boolean expression for output:

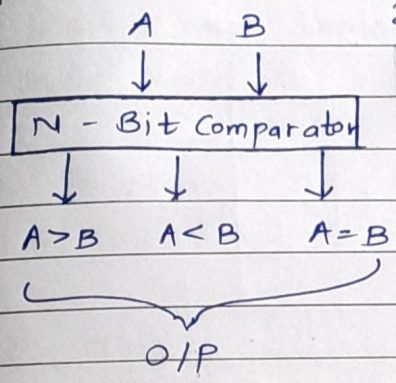
$S(A, B, C_{in}) = \Sigma m\ (1, 2, 4, 7)$

$C(A, B, C_{in}) = \Sigma m\ (3, 5, 6, 7)$



Page No.
Date

Comparators:



1. One - bit comparator:

Q. Design & implement 1-bit comp. by using logic gates.

No. of inputs : 02 = A, B

No. of outputs : 03 = A>B, A<B, A=B

Truth table :

Inputs		Outputs		
A	B	A > B	A < B	A = B
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

K-map :

k-map for A>B

A \ B	$\bar{B}$ 0	B 1
$\bar{A}$ 0	0	0
A 1	1	0

$\therefore \quad Y = A\bar{B}$

k-map for A < B

A \ B	0	1
0	0	1
1	0	0

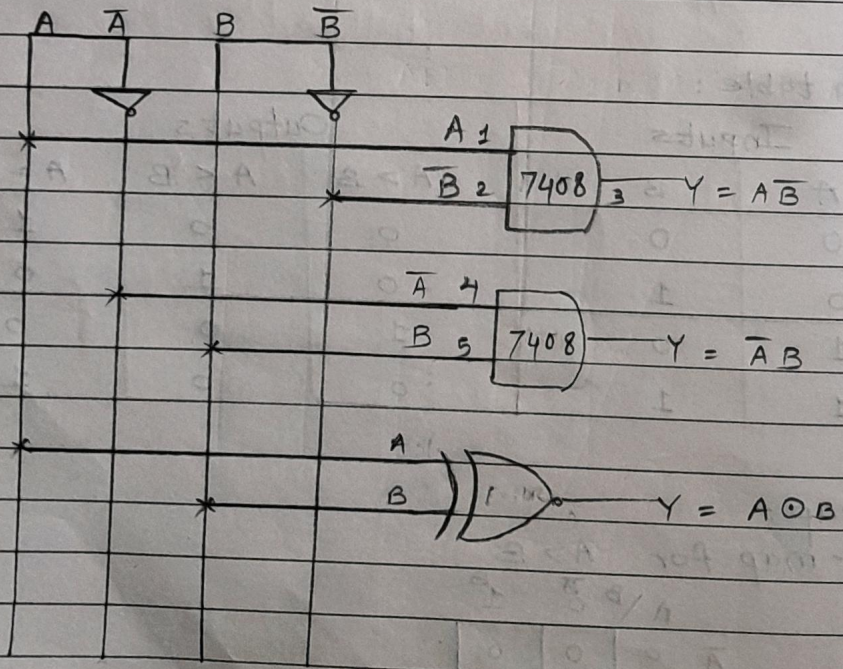
$\therefore Y = \bar{A}B$

k-map for A = B

A \ B	0	1
0	1	0
1	0	1

$\therefore Y = \bar{A}\bar{B} + AB = A \odot B$

Implementation:



2. Two-bit comparator:

Q. Design and implement 2-bit comp. by using basic gates.

No. of inputs : 02 (A, B) = A_0, A_1, B_0, B_1

No. of outputs: 03 (A, B) = A>B, A<B, A=B

Truth table:

	Inputs				outputs		
	A_1	A_0	B_1	B_0	A>B	A<B	A=B
0	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0
2	0	0	1	0	0	1	0
3	0	0	1	1	0	1	0
4	0	1	0	0	1	0	0
5	0	1	0	1	0	0	1
6	0	1	1	0	0	1	0
7	0	1	1	1	0	1	0
8	1	0	0	0	1	0	0
9	1	0	0	1	1	0	0
10	1	0	1	0	0	0	1
11	1	0	1	1	0	1	0
12	1	1	0	0	1	0	0
13	1	1	0	1	1	0	0
14	1	1	1	0	1	0	0
15	1	1	1	1	0	0	1

K-map:

K-map for A > B

A_1A_0 \ B_1B_0	00	01	11	10
00	0 (0)	0 (1)	0 (3)	0 (2)
01	1 (4)	0 (5)	0 (7)	0 (6)
11	1 (12)	1 (13)	0 (15)	1 (14)
10	1 (8)	1 (9)	0 (11)	0 (10)

$\therefore (Y)^{A>B} = A_1\overline{B_1} + A_0\overline{B_1}\,\overline{B_0} + A_1A_0\overline{B_0}$

K-map for A < B

A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

$\therefore Y = \overline{A_1}B_1 + \overline{A_0}B_1B_0 + \overline{A_1}\,\overline{A_0}B_0$

K-map for A = B

A_1A_0 \ B_1B_0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

$\therefore Y = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}A_0\overline{B_1}B_0 + A_1A_0B_1B_0 + A_1\overline{A_0}B_1\overline{B_0}$

$= A_1B_1(A_0B_0 + \overline{A_0}\,\overline{B_0}) + \overline{A_1}\,\overline{B_1}(\overline{A_0}\,\overline{B_0} + A_0B_0)$

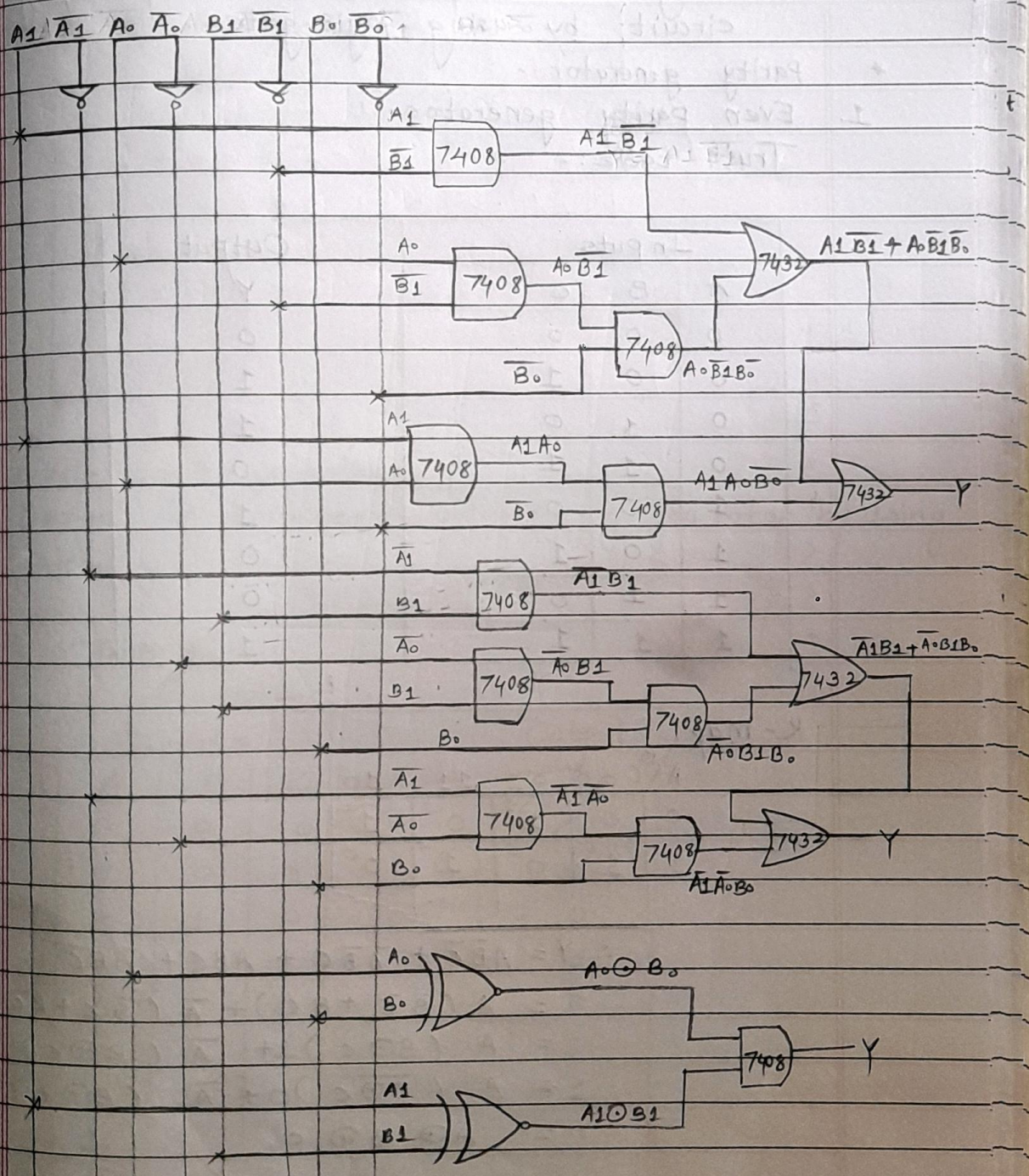
$= A_1B_1(A_0 \odot B_0) + \overline{A_1}\,\overline{B_1}(A_0 \odot B_0)$

$= A_0 \odot B_0\ (A_1B_1 + \overline{A_1}\,\overline{B_1})$

$= (A_0 \odot B_0)(A_1 \odot B_1)$

Page No.
Date

Implementation :



either 3bit or 2bit in exam

Page No.
Date

* Parity generator and checker:

Q. Design and implement 3-bit parity generator circuit by using logic gates:

* Parity generator :-

1. Even parity generator:

Truth table:

	Inputs			Output
	A	B	C	Y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

K-map:

A \ BC	00	01	11	10
0	0^0	$(1)^1$	0^3	$(1)^2$
1	$(1)^4$	0^5	$(1)^7$	0^6

$\therefore Y = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$

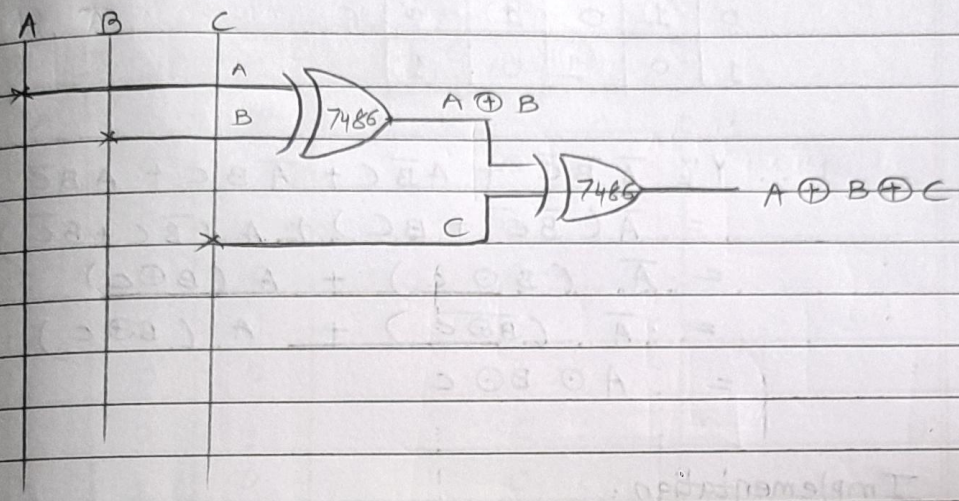
$= A(\bar{B}\bar{C} + BC) + \bar{A}(\bar{B}C + B\bar{C})$

$= A(B \odot C) + \bar{A}(B \oplus C)$

$= A(\overline{B \oplus C}) + \bar{A}(B \oplus C)$

$= A \oplus B \oplus C$ ✓

Implementation:



2. odd- parity generator:

Q. Design & implement odd-parity generator by using gates:

Truth table:

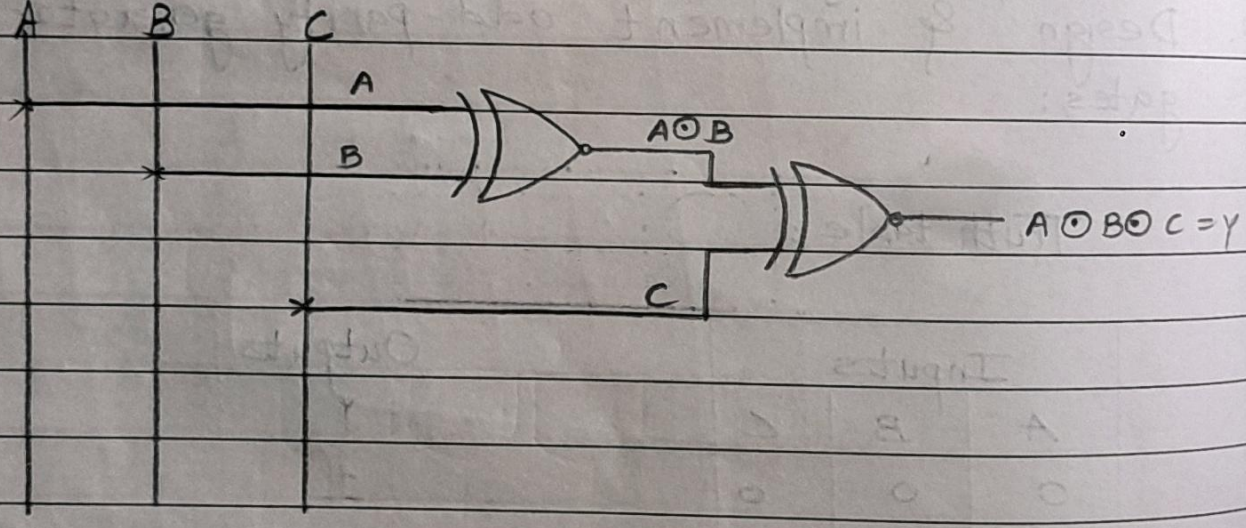
Inputs			Outputs
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

K-map:

A \ BC	00	01	11	10
0	(1)	0	(1)	0
1	0	(1)	0	(1)

$$\therefore Y = \bar{A}\bar{B}\bar{C} + A\bar{B}C + \bar{A}BC + AB\bar{C}$$
$$= \bar{A}(\bar{B}\bar{C} + BC) + A(\bar{B}C + B\bar{C})$$
$$= \bar{A}(B \odot C) + A(B \oplus C)$$
$$= \bar{A}(\overline{B \oplus C}) + A(B \oplus C)$$
$$= A \odot B \odot C$$

Implementation:



Page No.
Date

Parity checker:-

Truth table:

even parity o/p ⇒ 0
odd parity ⇒ 1

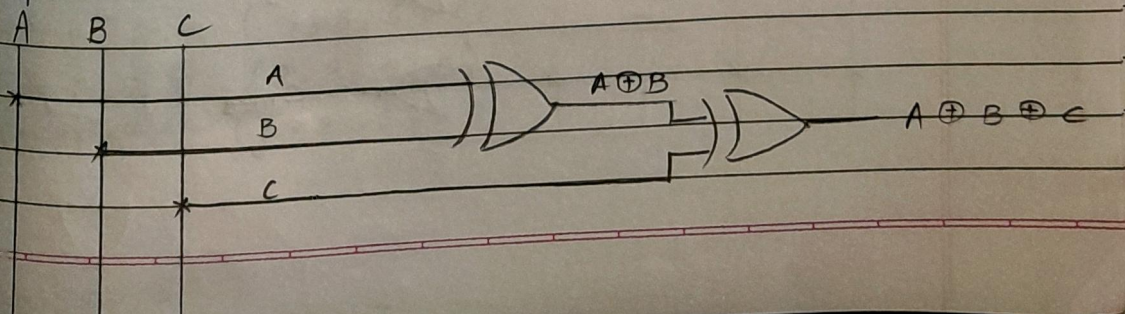
Input			Parity bit
A	B	C	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

K-map:

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	0	(1)	0	(1)
A 1	(1)	0	(1)	0

$$\therefore Y = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$$
$$= A(\bar{B}\bar{C} + BC) + \bar{A}(\bar{B}C + B\bar{C})$$
$$= A(B \odot C) + \bar{A}(B \oplus C)$$
$$= A(\overline{B \oplus C}) + \bar{A}(B \oplus C)$$
$$= A \oplus B \oplus C.$$

Implementation:

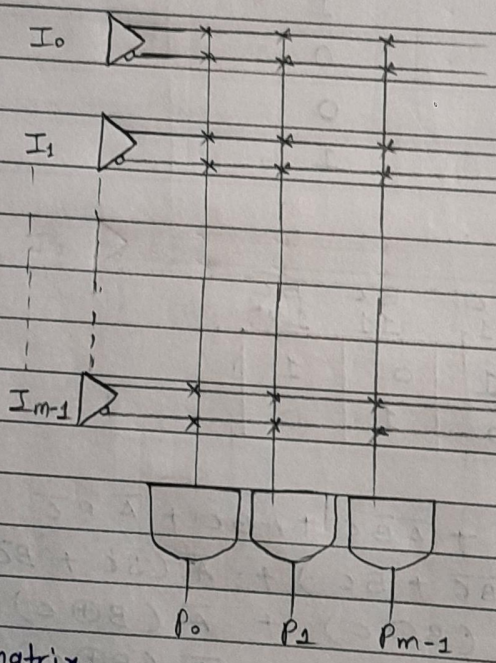


Used for SOP

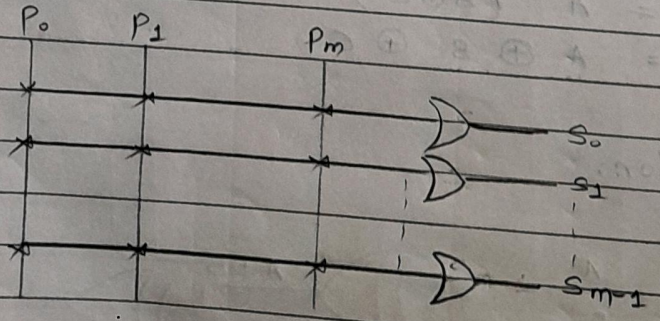
Page No.
Date

* Programmable Logic Design (PLD):
- PLD is a LSI (Large Scale Integration) circuit.
- PLD has 4 basic building blocks :-
  (i) Inverters / buffers → used to store digital information
  (ii) AND matrix (Array of AND gate)
  (iii) OR matrix (Array of OR gate)
  (iv) Tri-state buffers & programmable inverters

AND matrix:



OR matrix:



Page No.
Date

* There are 3 types of PLD circuits:
1. ROM (Read Only Memory)
2. PLA (Programmable Logic Array)
3. PAL (Programmable Array Logic)

1) ROM: Here, Fixed array → AND ; Programmable array → OR

Q. Design 3-bit binary to gray code converter using suitable ROM and de-coder.

⇒ No. of inputs: 03 = B_2, B_1, B_0
No. of outputs: 03 = G_{12}, G_{11}, G_{10}

Truth table:

Inputs			Outputs			
B_2	B_1	B_0	G_{12}	G_{11}	G_{10}	
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	1	1	2
0	1	1	0	1	0	3
1	0	0	1	1	0	4
1	0	1	1	1	1	5
1	1	0	1	0	1	6
1	1	1	1	0	0	7

The logic expression for output are:

$G_{12} = \Sigma m\ (4, 5, 6, 7)$

$G_{11} = \Sigma m\ (2, 3, 4, 5)$

$G_{10} = \Sigma m\ (1, 2, 5, 6)$

Q. Implement the foll. fun^{n} by using ROM

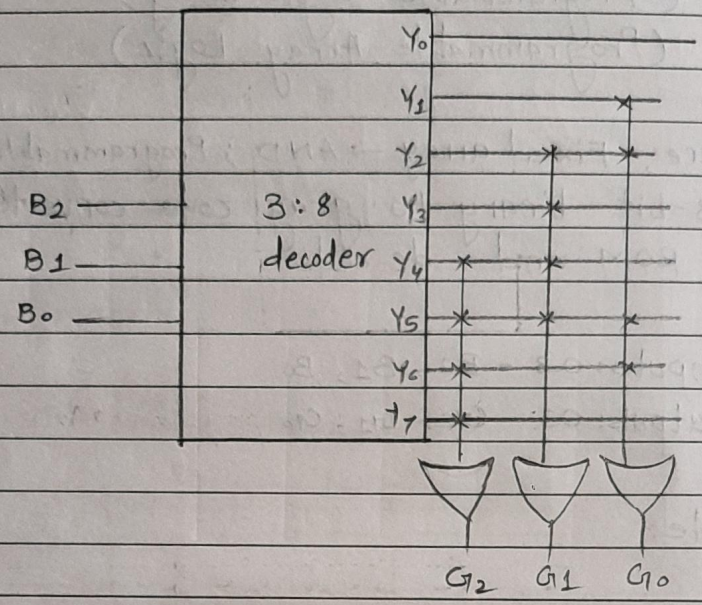
$f_1 = \Sigma m\,(1,3,4,6)$; $f_2 = \Sigma m(2,4,5,7)$

$f_3 = \Sigma m\,(0,1,5,7)$; $f_4 = \Sigma m(1,2,3,4$

Page No.

Date

Circuit diagram:



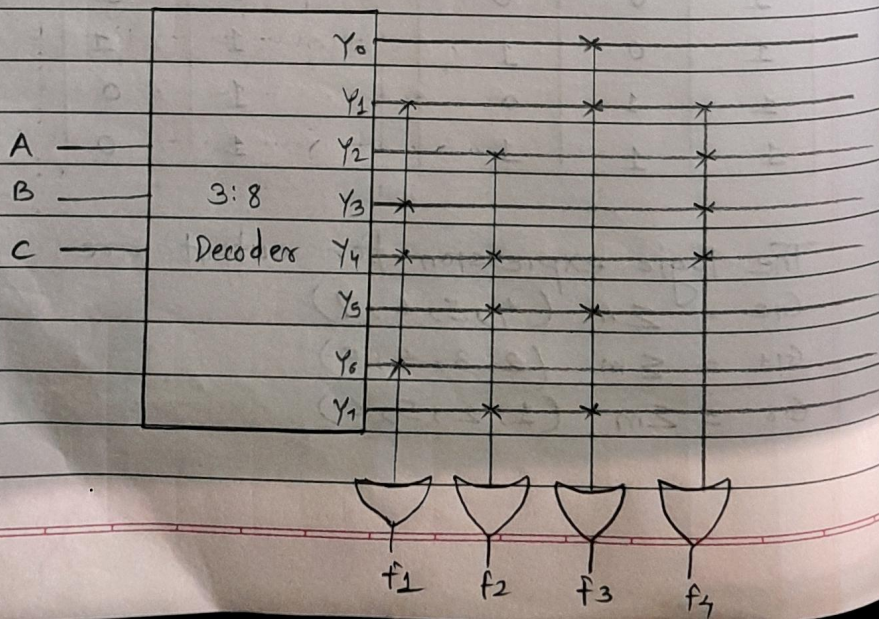
H.W

Q. Implement the following function by using ROM.

$f_1 = \Sigma m\,(1,3,4,6)$; $f_2 = \Sigma m\,(2,4,5,7)$;

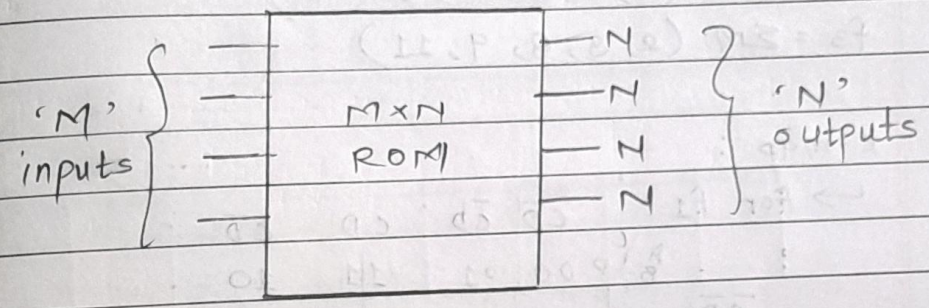
$f_3 = \Sigma m\,(0,1,5,7)$; $f_4 = \Sigma m\,(1,2,3,4)$

⟶ Circuit diagram:



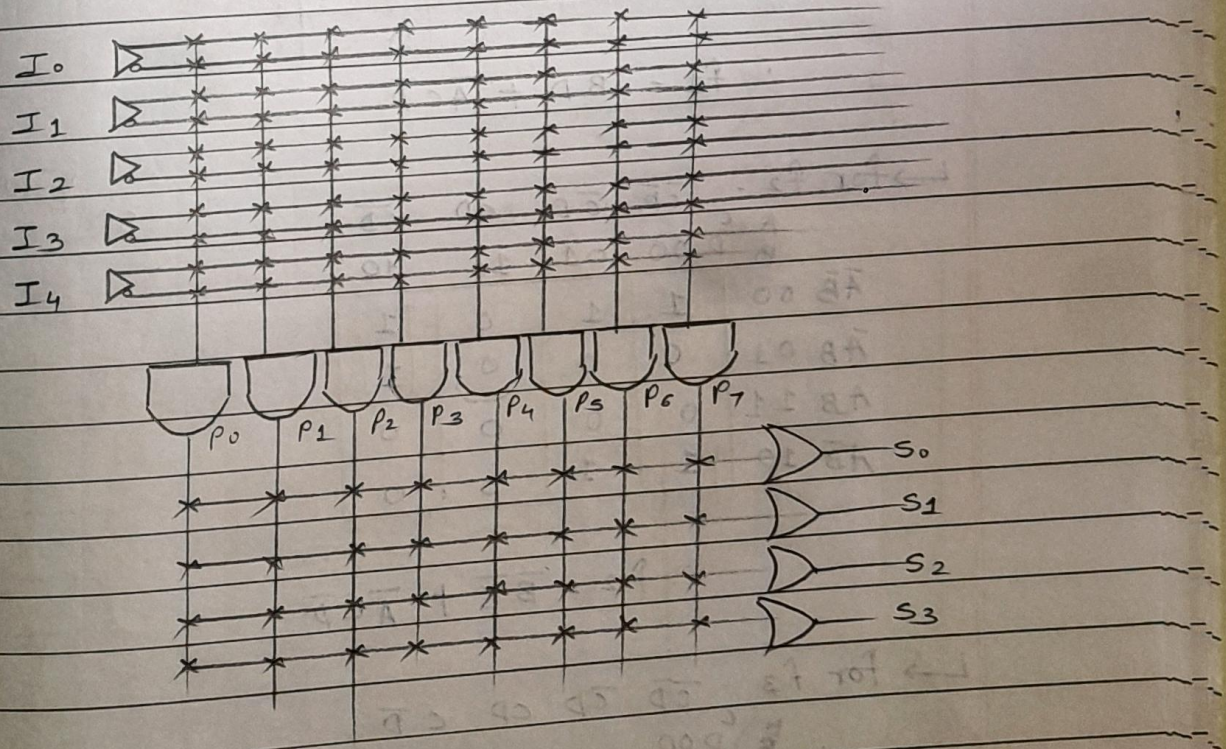
Page No.
Date

Block diagram for ROM:



2) PLA : Both AND & OR are programmable array

Block diagram for PLA :



Page No.
Date

Q. Design the circuit with optimum utilization of PLA implement the following fucⁿ:

$f_1 = \Sigma m(1, 2, 3, 6, 9, 11)$

$f_2 = \Sigma m(0, 1, 2, 6, 8, 9)$

$f_3 = \Sigma m(2, 3, 8, 9, 11)$

⟶ k-map :

↳ For f_1 -

AB \ CD	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$ 00	0	1	1	1
$\bar{A}B$ 01	0	0	0	1
AB 11	0	0	0	0
$A\bar{B}$ 10	0	1	1	0

$\therefore f_1 = \bar{B}D + \bar{A}C\bar{D}$

↳ for f_2 -

AB \ CD	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$ 00	1	1	0	1
$\bar{A}B$ 01	0	0	0	1
AB 11	0	0	0	0
$A\bar{B}$ 10	1	1	0	0

single line in ckt

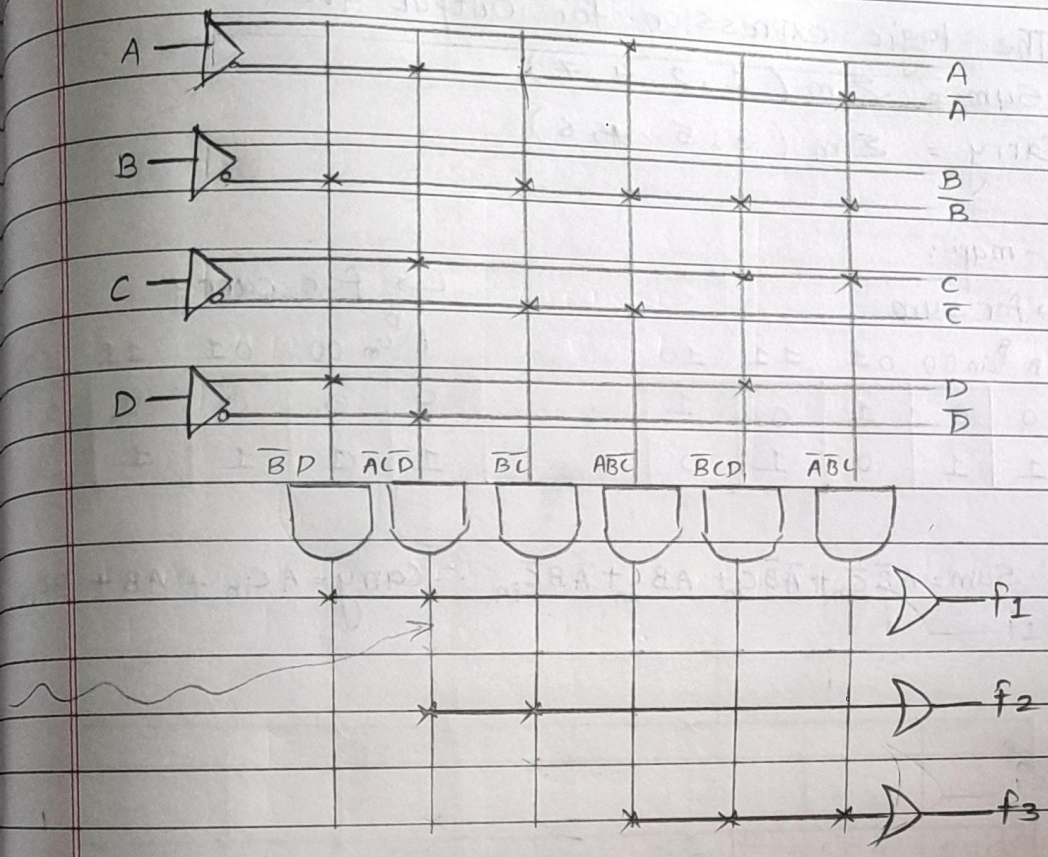
$\therefore f_2 = \bar{B}\bar{C} + \bar{A}C\bar{D}$

↳ for f_3 -

AB \ CD	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$ 00	0	0	1	1
$\bar{A}B$ 01	0	0	0	0
AB 11	0	0	0	0
$A\bar{B}$ 10	1	1	1	0

$\therefore f_3 = A\bar{B}\bar{C} + \bar{B}CD + \bar{A}\bar{B}C$

Prog Q. Implem. full adder using suitable PLA



Q. Implement full adder using suitable PLA

⇒ Truth table:

Inputs			Outputs	
A	B	Cin	Sum	Carry
0	0	0	0 0	0
0	0	1	1 1	0
0	1	0	1 2	0
0	1	1	0 3	1
1	0	0	1 4	0
1	0	1	0 5	1
1	1	0	0 6	1
1	1	1	1 7	1

The logic expression for output are:

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

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

k-map:

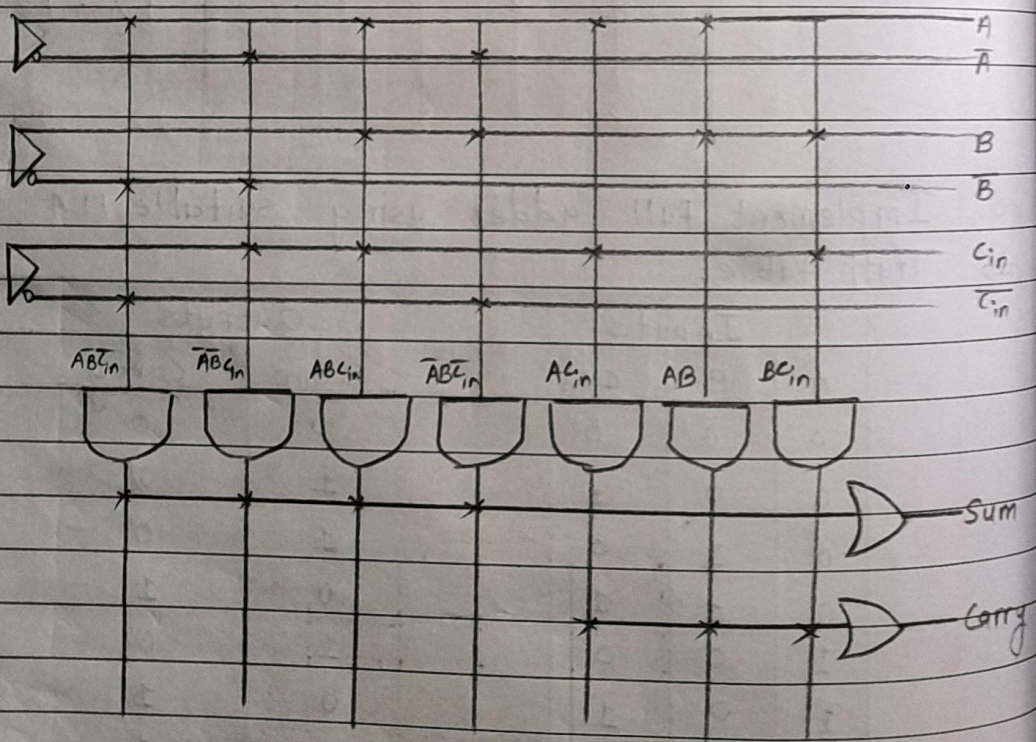
↳ for sum

A \ B C_{in}	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

↳ for carry

A \ B C_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

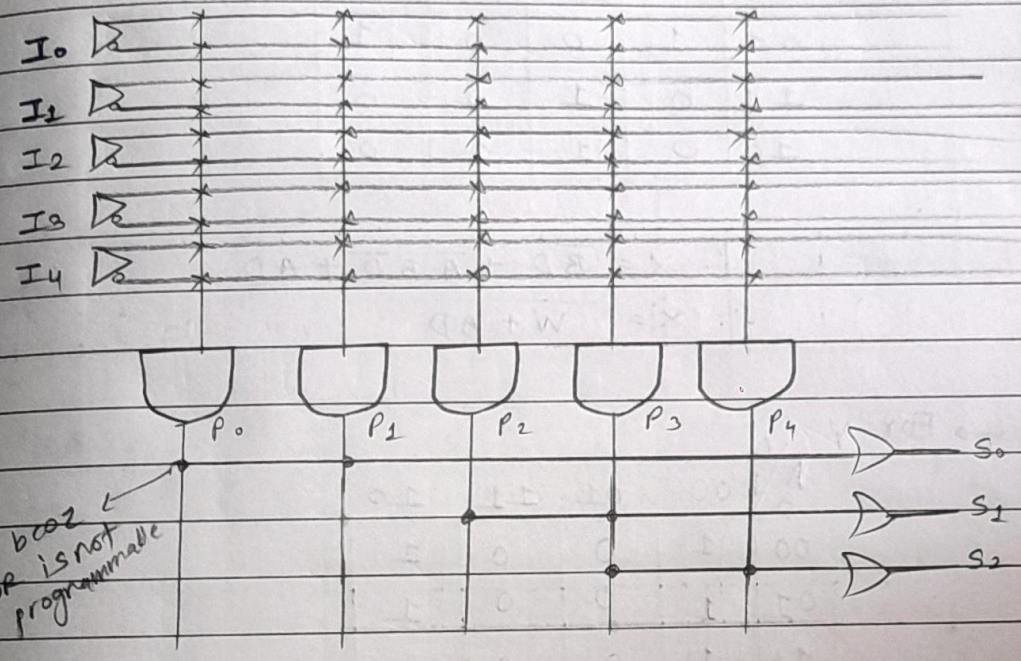
$\therefore$ Sum = $A\bar{B}\bar{C}_{in} + \bar{A}\bar{B}C_{in} + ABC_{in} + \bar{A}B\bar{C}_{in}$ $\therefore$ Carry = $AC_{in} + AB + BC_{in}$



Page No.
Date

PAL : Here, AND → Programmable array ; OR → Fixed array

Block diagram for PAL:



Q. Design the combinational circuit using suitable PAL.

$W(A,B,C,D) = \Sigma m(1,3,4,6,9,11)$

$X(A,B,C,D) = \Sigma m(1,3,4,6,9,11,13,15)$

$Y(A,B,C,D) = \Sigma m(0,2,4,6,8,12)$

⇒ K-map:

↳ for W

AB \ CD	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	0	0	0	0
10	0	1	1	0

$\therefore W = \bar{B}D + \bar{A}B\bar{D}$

↳ For x

AB \ CD	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	0	1	1	0
10	0	1	1	0

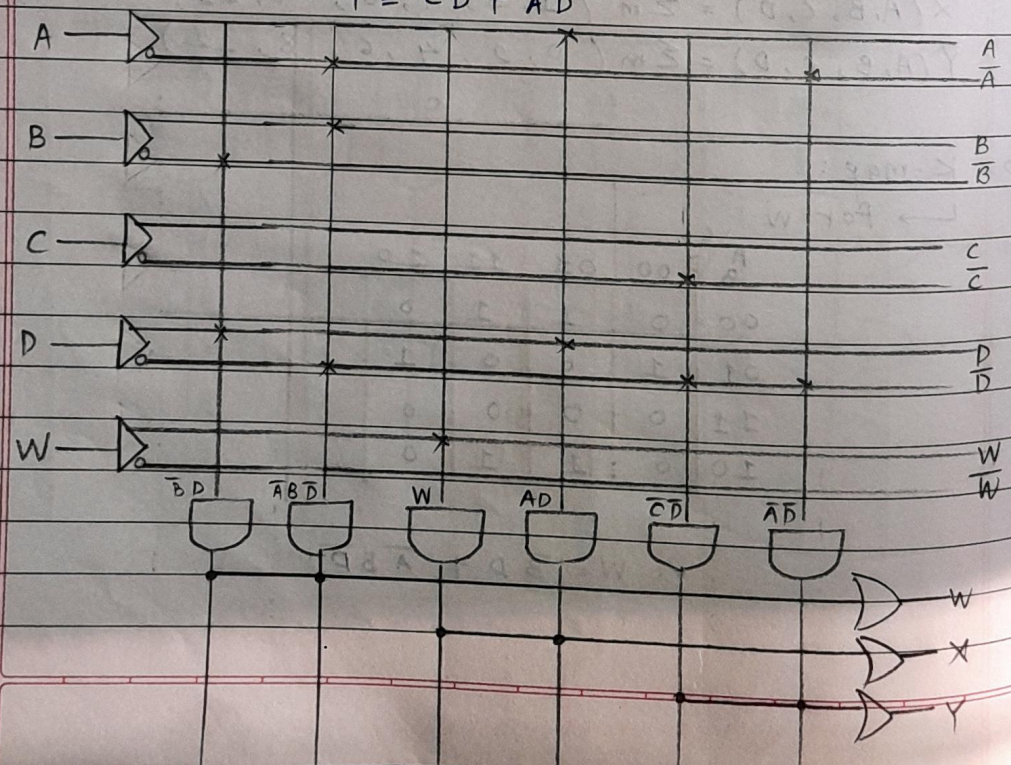
$\therefore X = \overline{B}D + \overline{A}B\overline{D} + AD$

$\therefore X = \overline{W + AD}$

↳ For Y

AB \ CD	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	0
10	1	0	0	0

$\therefore Y = \overline{C}\overline{D} + \overline{A}\overline{D}$



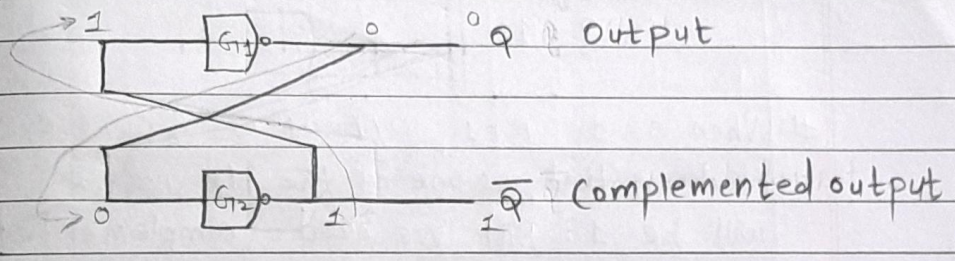
Q. Difference betn combination & seq. logic ckt

Unit 04:

NAND
↳ If one of the i/p is 0 → o/p is 1

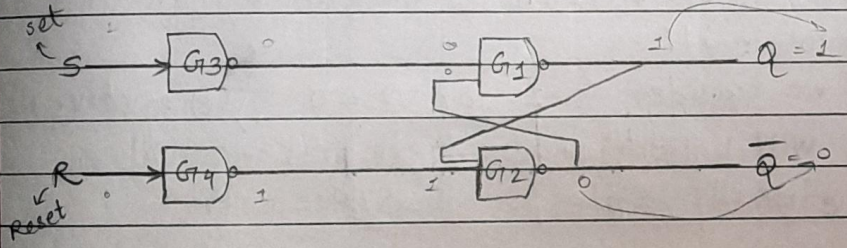
1-bit memory cell:

Cross-coupled inverter as a memory element:



Memory-cell with provision of entering the data:

↳ S-R flip flop



Truth table:

S	R	o/p
0	0	Q_n
0	1	0
1	0	1
1	1	Invalid state.

When S=0, R=1 o/p of G3 = 1 & that of G4 = 0 (∵ NAND ≥ 1 i/p acts as inverter) as G2's i/p is 0, we know that when 1 of the i/p is 0 in NAND, then o/p is 1. ∴ o/p of G2 = 1. This 1 will go to one of the i/p of G1 & One i/p for G1 is already 1. So, according to NAND, 1 1 → 0 . ∴ o/p = 0 → Q

In seq. ckt, o/p depends on :- 1) Present i/p
2) Past o/p
{ Combination of i/p & present state } generate Next state & Present state
Locla o/p

When S=1, R=0 o/p of G3 = 0 & that of G4 = 1
As One of the i/p of G1 is 0, o/p of G1 = 1 This 1 acts as i/p to G12 ∴ o/p of G12 = 0
Cross check: 0 will go to one of the i/p to G11, then 00 ∴ Q = 1

When S=1, R=1, o/p of G3 & G4 = 0
We know that, as one of the i/p is '0', o/p of both will be 1. But we need complemented output.
∴ This is not possible → Invalid.

When S=0, R=0, o/p of G3 & G4 = 1
Now, we are not sure what will be the o/p as there no '0' in i/p.
If we consider Q=1 or Q=0, the value of Q & $\bar{Q}$ will be same as their previous value.
i.e Hold State. ∴ o/p => Q_n

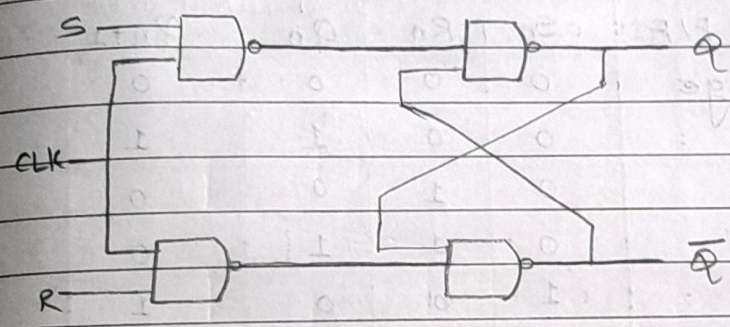
$\overline{1 \cdot Q} \Rightarrow \bar{1} + \bar{Q} = 0 + Q = Q$ → same as actual o/p
$\overline{1 \cdot \bar{Q}} \Rightarrow \bar{1} + \bar{\bar{Q}} = 0 + \bar{Q} = \bar{Q}$ ∴ No change
(Hold)

Flip-flop IC: 7490

In exam, all F/F are clocked only.

For S-R flop in exam

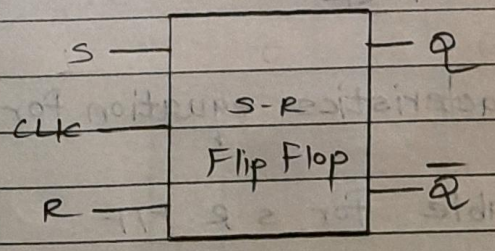
Clocked S-R Flip flop :



↳ Truth table of clocked S-R F/F

Input		Output	state of F/F
S	R	Q_{n+1}	
0	0	Q_n	No change
0	1	0	Reset
1	0	1	set
1	1	?	Invalid state

↳ Block diagram for clocked S-R F/F :



Not in exam from here for all F/F

↳ Characteristics table for clocked S-R F/F:

State of F/F	Sn	Rn	present state Qn	Next state Qn+1	
No change	0	0	0	0	0
	0	0	1	1	1
Reset	0	1	0	0	2
	0	1	1	0	3
set	1	0	0	1	4
	1	0	1	1	5
Invalid state	1	1	0	X	6
	1	1	1	X	7

↳ k-map [For Qn+1 / Next state]

Q \ SR	00	01	11	10
0	0 (0)	0 (2)	X (6)	1 (4)
1	1 (1)	0 (3)	X (7)	1 (5)

~~$= Q + \bar{S}R$~~

$Q_{n+1} = S + Q\bar{R}$

This is characteristics equation for S-R F/F.

Not in exam

↳ Excitation table for S-R F/F

Qn	Qn+1	Sn	Rn
0	0	same o/p in 0 & final state 0	0 (Diff o/p & final state) / 1
0	1	1	0
1	0	0	1
1	1	0	0
		1	0

} Qn & Qn+1 are 00 at 00 & 01 in table

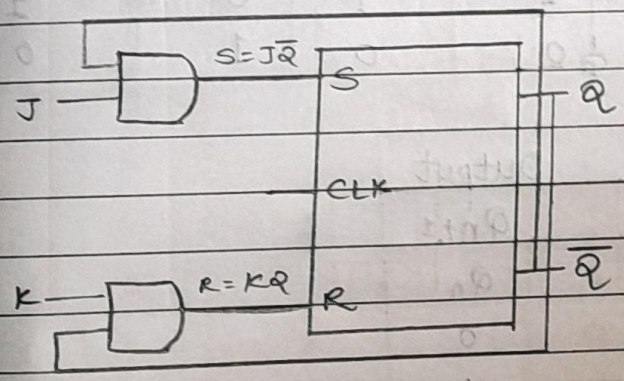
Exam

↳ Final Excitation table for S-R F/F

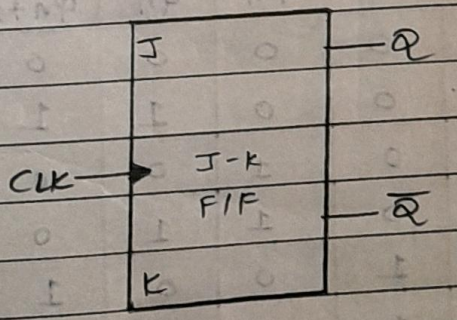
Q_n	Q_{n+1}	S_n	R_n
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

J-K Flip flop:

↳ circuit diagram for J-K F/F :



↳ Block diagram of J-K F/F:



↳ Truth table of J-K F/F:

Data inputs		Outputs S-R		Inputs to S-R		Outputs J-K	
Jn	Kn	Qn	$\overline{Qn}$	$S_n = J\overline{Q_n}$	$R_n = KQ_n$	Qn+1	
0	0	0	1	0	0 (at 00)	0	Qn
0	0	1	0	0	0 (No change means Qn)	1	Qn
0	1	0	1	0	0	0	Reset
0	1	1	0	0	1 → here, Qn doesn't matters	0	Reset
1	0	0	1	1	0	1	set
1	0	1	0	0	0	1	set
1	1	0	1	1	0	1	$\overline{Qn}$
1	1	1	0	0	1	0	$\overline{Qn}$

modified

Input		Output
J	K	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	$\overline{Qn}$

↳ Characteristics table :

State of F/F	Jn	Kn	Qn	Qn+1
No change	0	0	0	0
	0	0	1	1
Reset	0	1	0	0
	0	1	1	0
set	1	0	0	1
	1	0	1	1
Toggle state	1	1	0	1
	1	1	1	0

↳ k-map

Q \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$\therefore Q_{n+1} = J\bar{Q} + \bar{K}Q$

This is characteristics equation for J-K F/F.

↳ Excitation table:

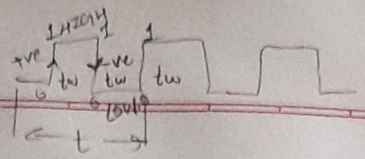
Q_n	Q_{n+1}	J_n	K_n
0	0	0	0
0	0	0	1
0	1	1	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

↳ Final excitation table:

Q_n	Q_{n+1}	J_n	K_n
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

20-11-23

* CLK



$t = 2t_w$

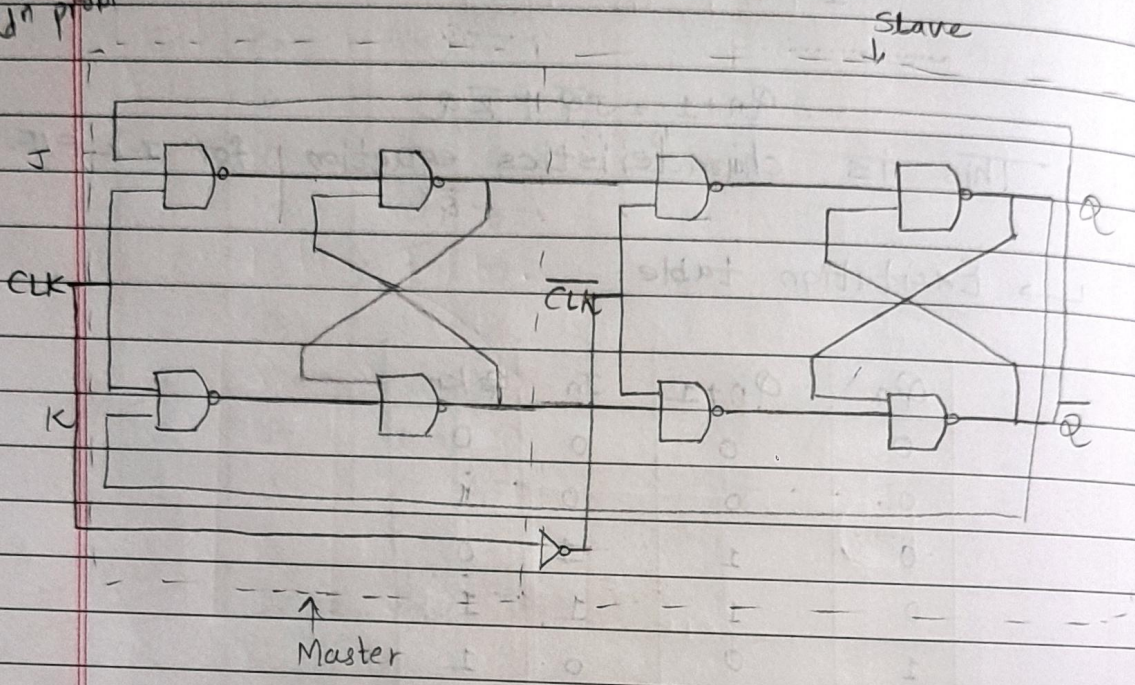
Page No.

Date $t_w > t_p$

* Master slave J-K F/F :

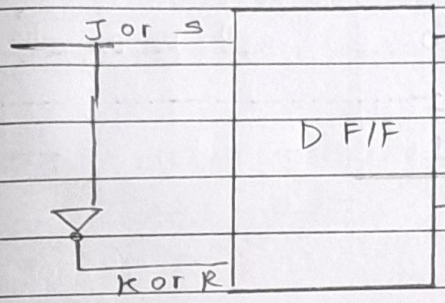
↳ To overcome the difficulty in toggle condⁿ in clock ↯ where it's value fluctuates

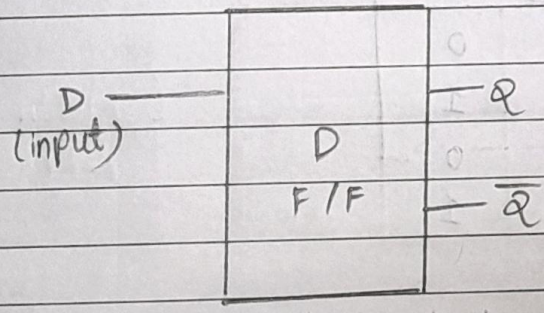
Due to race around condⁿ problem



D- flip flop (Delay F/F):

↳ Block diagram of D- F/F:





↳ Truth table of D F/F:

Input	Output
D	Q
0	0
1	1

↳ Characteristics equation:

$$Q_{n+1} = D$$

↳ Characteristics table for D-F/F:

Dn	Q	Qn+1
0	0	0
0	1	0
1	0	1
1	1	1

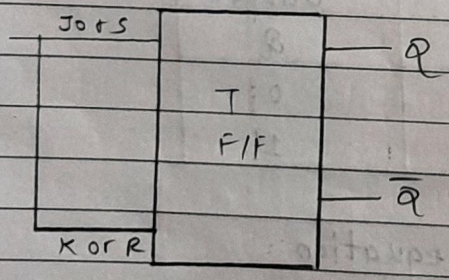
} when D is 0, next state (Q_{n+1}) will be 0, irrespective of Q

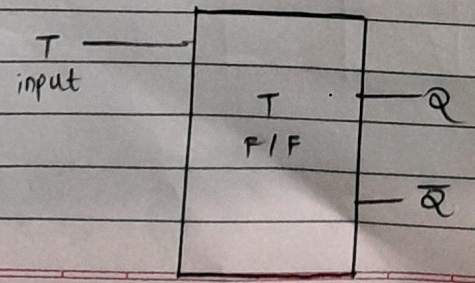
↳ Excitation table:

Qn	Qn+1	D
0	0	0
0	1	1
1	0	0
1	1	1

T-flip flop (toggle/switch F/F):

↳ Block diagram for T F/F:





↳ Truth table:

Input	Output
T	Q_{n+1}
0	Q_n
1	$\overline{Q_n}$

} 1st & last row of J-K

↳ Characteristics equation:

$$Q_{n+1} = T \oplus Q$$

↳ Characteristics table:

State of F/F	T	Q_n	Q_{n+1}
No change	0	0	0
	0	1	1
Toggle	1	0	1
	1	1	0

↳ Excitation table:

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

* Register:
↳ Chain of flip-flop.
↳ To save/store multiple bit data.

* Shift register:- (Implemented by D-F/F only) → as output doesn't change
- There are two types of shift register:-
  i) Unidirectional shift register
  ii) Bidirectional -"-

i) Unidirectional shift register:

02 ⇒ 32 16 8 4 2 1

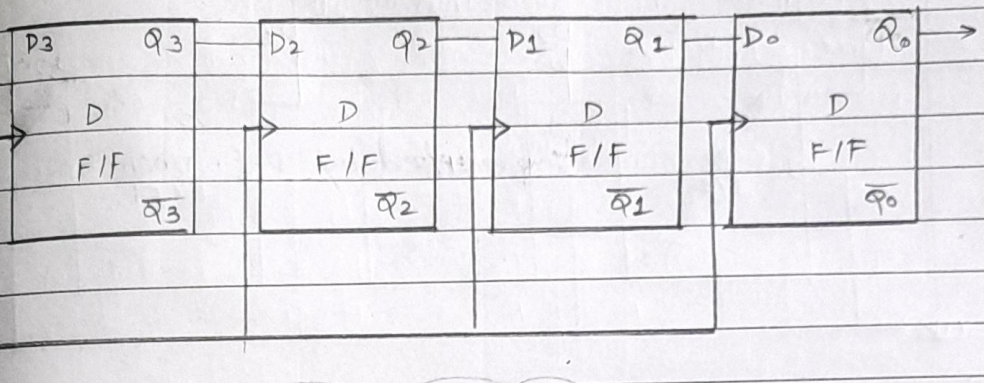
0 0 0 0 0 1 0 0 — 02 H
0 0 0 0 0 1 0 0 (empty space) — 04 H
0 0 0 0 1 0 0 0 — 08 H
0 0 0 1 0 0 0 0 — 10 H
} Multiply by 2 Left shift

0 0 0 1 0 0 0 0 — 8 H
0 0 0 0 0 1 0 0 — 4 H
0 0 0 0 0 0 1 0 — 2 H
} Divide by 2. Right shift

There are 4 types of unidirectional shift register.
1. SISO (Serial In serial Out)
2. SIPO (Serial In Parallel out)
3. PISO (Parallel In serial Out)
4. PIPO (Parallel In Parallel Out)

Page No.
Date

① SISO ⇒



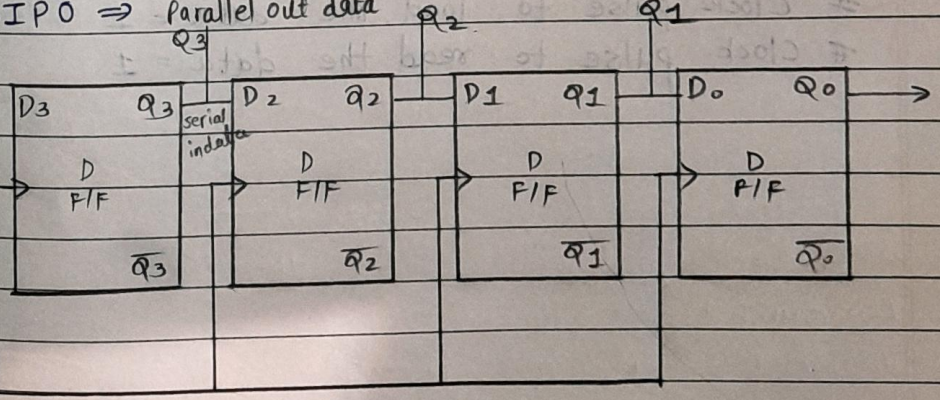
	Q_3	Q_2	Q_1	Q_0
	0	0	0	0
C_1	1	0	0	0
C_2	1	1	0	0
C_3	1	1	1	0
C_4	1	1	1	1

clock pulse

Clock pulse to load the data = N → 4 (C_1 → C_2 → C_3 → C_4)
Clock pulse to read the data = (N-1)

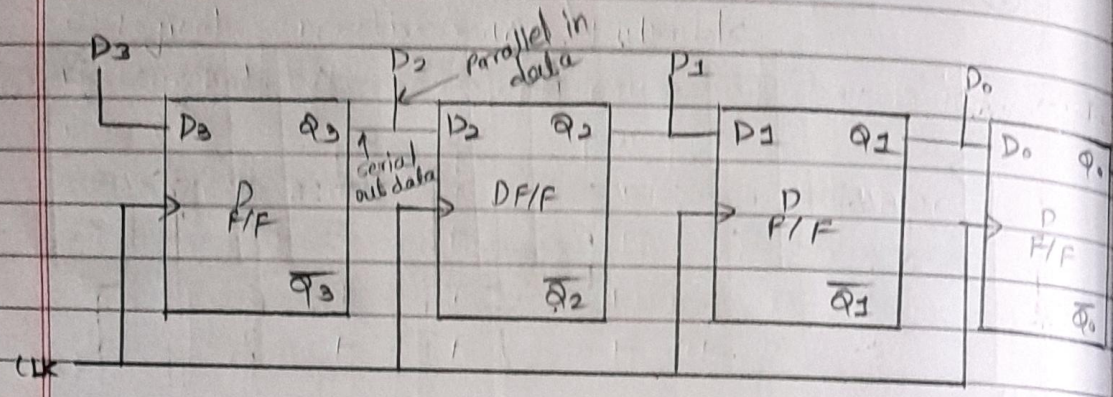
1 1 1 1
↳ 1 from last Q_0 o/p
remaining 3 i.e N-1

② SIPO ⇒ Parallel out data



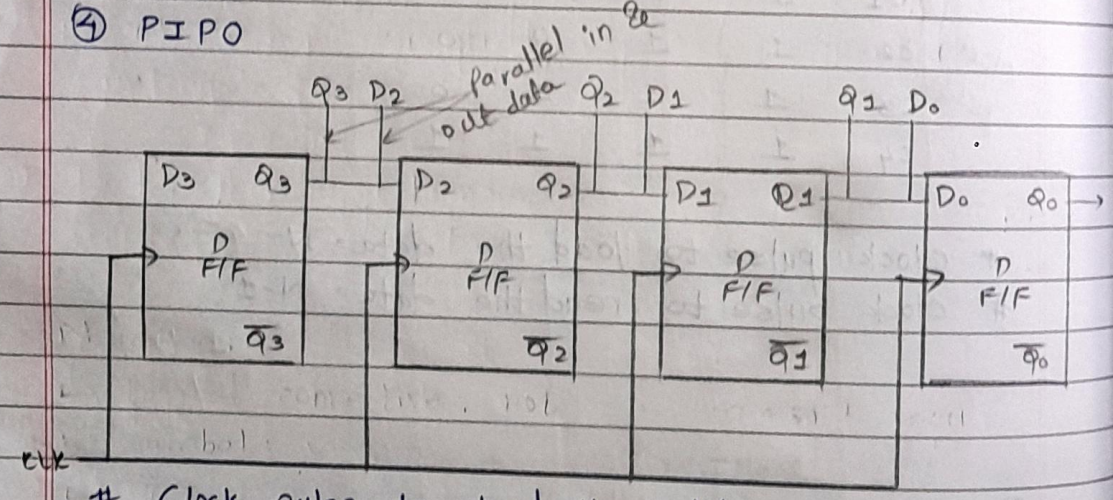
Clock pulse to load the data = N
Clock pulse to read the data = 1

③ PISO



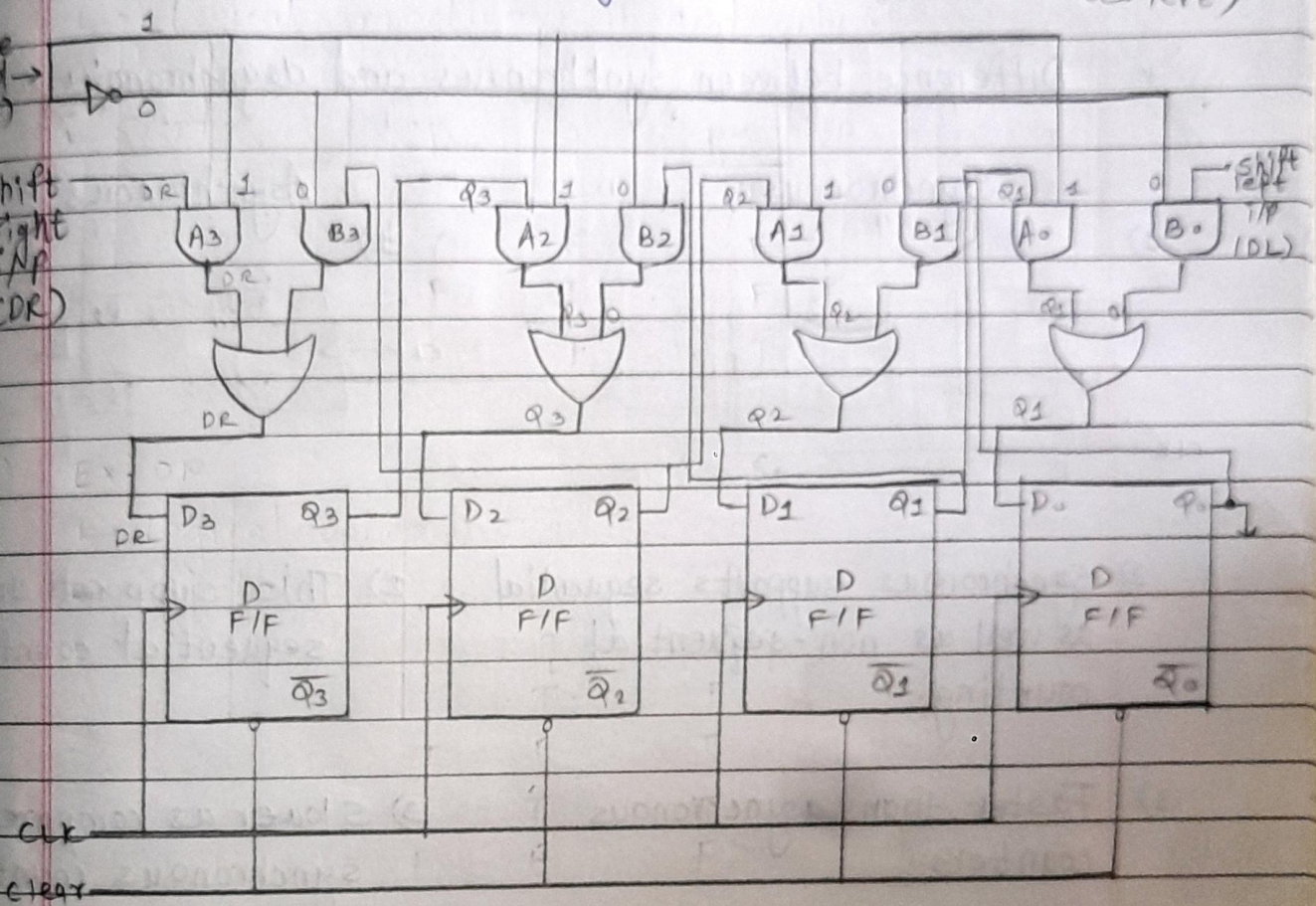
Clock pulse to load the data = 1
Clock pulse to read the data = N-1

④ PIPO



Clock pulse to load the data = 1
Clock pulse to read the data = 1

ii) Bidirectional shift register: (data can flow from R as well as left)



Note: if M = 0 shift left
if M = 1 shift right

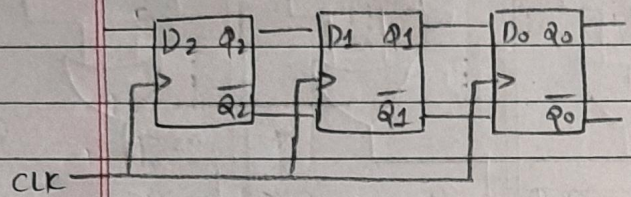
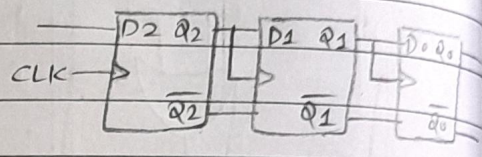
J-K = 7476

Page No.
Date

* Counters:

Unit 6 Imp

Difference between synchronous and asynchronous counters.

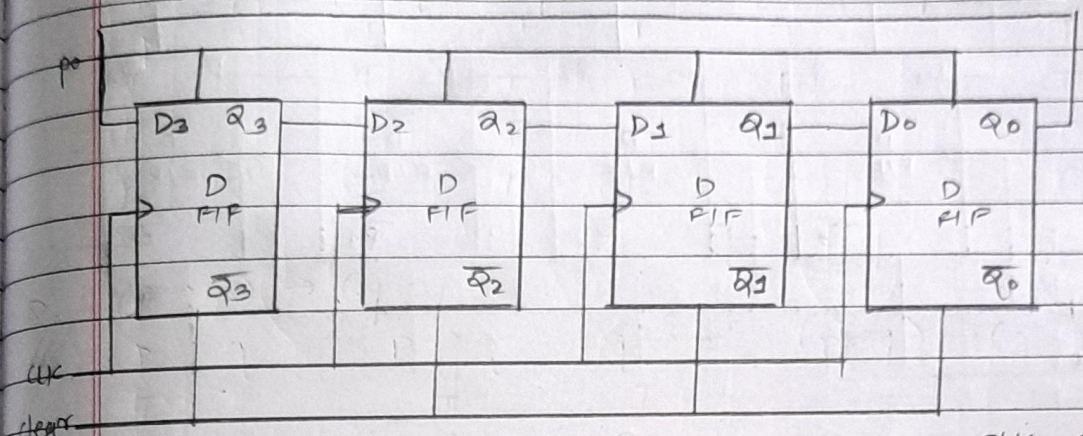
	Synchronous	Asynchronous
1)		
2)	Synchronous supports sequential as well as non-sequential counting.	This supports only sequential counting.
3)	Faster than asynchronous counters.	Slower as compare to synchronous counters.
4)	Complex in design, it requires gates also.	Simple in design, it doesn't require gates.
5)	Having higher cost.	Lower cost
6)	e.g Ring counter, twisted ring counter.	e.g Ripple counter.

3M for each counter (ring & twisted)

Page No.
Date

Synchronous counters:

Ring counter:



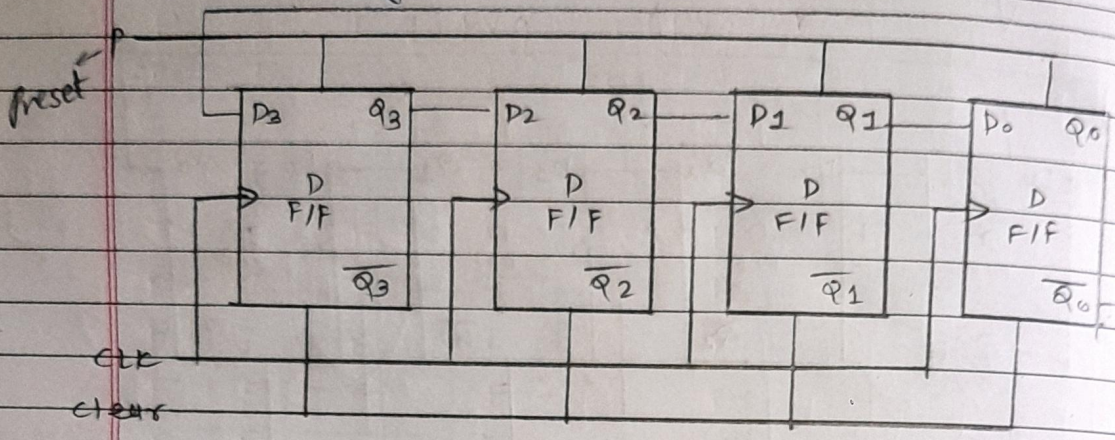
CLK $= 2^n = 2^4 = 16$ required.
∴ Here, $C_1 \cdots C_{16}$

$Q_3(n+1) = D_3 = Q_0(n)$
$Q_2(n+1) = D_2 = Q_3(n)$
$Q_1(n+1) = D_1 = Q_2(n)$
$Q_0(n+1) = D_0 = Q_1(n)$

	Q_3	Q_2	Q_1	Q_0
	1	0	0	0
CL1	0	1	0	0
CL2	0	0	1	0
CL3	0	0	0	1
CL4	1	0	0	0

Here, to return at original (1000), we required 4 CLK cycles.

* Twisted ring counter:



	Q_3	Q_2	Q_1	Q_0
	1	0	0	0
CL1	1	1	0	0
CL2	1	1	1	0
CL3	1	1	1	1
CL4	0	1	1	1
CL5	0	0	1	1
CL6	0	0	0	1
CL7	0	0	0	0
CL8	1	0	0	0

Here, we required 8 clock cycles to return at 1000