

Digital Electronics

- Boolean
- Logic Gate
- K-Map

Conversion

Bits < 0

1 byte = 8 bits

1 nibble = 4 bits

1 double byte = 16 bits

1 kilo byte = 1024 bits

* Number System

Set of symbols used to represent particular information to perform arithmetic operation.

→ Decimal

→ Binary

→ Octal

→ Hexadecimal

[0, 1, 2, ..., 9]

[0, 1]

[0, 1, 2, ..., 7]

[0, 1, 2, ..., 9, A, B, C, D, E, F]

[0, 1, 2, ..., 9, A, B]

Use

Decimal → It is our language, that we are using to perform calculation.

Binary → Language that is understandable by computer.

Octal & hexadecimal → Used for compact representation.

Conversion

Decimal → Base (8).

Rules :

Divide continuously integer by '8'. Noted down remainder in reverse order.

Multiply fraction part continuously by '8' and noted down carry in same order.

ex (123.25)₁₀ → ()₂ for fraction part carry

2	123	1
2	61	1
2	30	0
2	15	1
2	7	1
2	3	1
	1	

(11101101)₂

Here while calculating integer part, remainder part 0, 1 if base is 2.

2	123	1 (0, 1)
2	61	1 (0, 1)

ex. $(12.25)_{10} \rightarrow ()_2$

$(1100.01)_2$

$.25 \times 2 = .50$

$.50 \times 2 = 1.0$

carry
0

1

2	12	0
2	6	0
2	3	1
	1	

★ Any Base To decimal.

ex $(1100.01)_2$

$= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$

$= 8 + 4 + \frac{1}{4} = 12 + \frac{1}{4} = \frac{49}{4} = 12.25$

ex. $(23E.2D)_{16} \rightarrow ()_{10}$

$2 \times 16^2 + 3 \times 16^1 + 14 \times 16^0 + 2 \times 16^{-1} + 13 \times 16^{-2}$

$= 512 + 48 + 14 + \frac{2}{16} + 0.0507 = (574.175)_{10}$

★ for To convert any Base into any other Base

$()_m = ()_n$

ex. $(732)_8 \rightarrow ()_2$

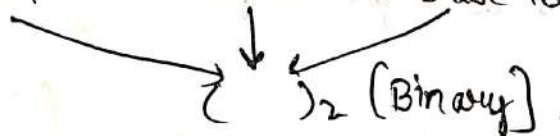
$\rightarrow ()_{10} \rightarrow$

$7 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 = (474)_{10} \rightarrow ()_{12}$

12	474	(6)
12	39	(3)
	3	

★ Alternate Method only for Base 2 power [for Binary]

ex. Base 4 Base 8 Base 16



Ques. $(321.33)_4 \rightarrow ()_2$

$= (111001.1111)_2$

$4 = 2^2$

$2^1 = 2$

$2^0 = 1$

Ques. $(632.15)_8 \rightarrow ()_2$

$= (110011010.001101)_2$

$8 = 2^3$

$2^2 = 4$

$2^1 = 2$

$2^0 = 1$

★ Now to convert (Binary)₂ to ()₄

ex $(01011011.01)_2 \rightarrow ()_4$ Ans $(1123.1)_4$

Grouping always start from decimal (LSB)

MSB = Most Significant Bit
LSB = Least Significant Bit

Binary Addition

1 carry = 1 Base

$$(01011 \cdot 0101)_2 + (1101 \cdot 0010)_2 = ()_2$$

→ firstly make pair if our base is 2 (that base in which we want answer). then make pair of 2. start from decimal, if any digit left unpaired then put zero before that number.

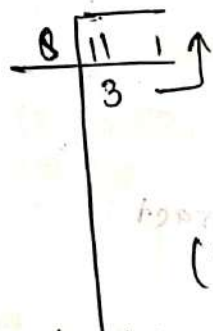
$$(00101011 \cdot 110)_2 + (023 \cdot 310)_4 = ()_8$$

$$(127.6)_8 + (023 \cdot 310)_4 = ()_8$$

$$2(2^4)2^3 = 8$$

$$(023 \cdot 310)_4 \rightarrow ()_{10}$$

$$0 \times 4^2 + 2 \times 4^1 + 3 \times 4^0 + 3 \times 4^{-1} + 1 \times 4^{-2} + 0 \times 4^{-3} = 11 + \frac{13}{16}$$



$$(31.64)_8$$

$$= 8 + \frac{31}{8} = 8.125 \times 8 = 65$$

$$(127.6)_8 + (31.64)_8 = ()_8$$

Subtract:

1 Borrow = 1 Base

$$\begin{pmatrix} 101 \\ - 11 \\ \hline 010 \end{pmatrix}_2$$

$$\begin{pmatrix} 1000 \\ - 0101 \\ \hline 0011 \end{pmatrix}_2$$

$$\begin{pmatrix} 400 \\ - 122 \\ \hline 256 \end{pmatrix}_8$$

$$\begin{pmatrix} 423 \\ - 256 \\ \hline 145 \end{pmatrix}_8$$

$$\begin{pmatrix} 10101101 \\ - 111012 \\ \hline 10010000 \end{pmatrix}_2$$

$$\begin{pmatrix} 100001001 \\ - 11101 \\ \hline 01101100 \end{pmatrix}_2$$

$$\begin{pmatrix} 1110010 \\ - 111101 \\ \hline 100110101 \end{pmatrix}_2$$

Complement

(Negative representation of No.)

r^1

$(r-1)^1$



but in binary we are studying



1's complement
2's complement

To find 1's compl.

put "0" in place of "1".
put "1" in place of "0".

ones

$$(10110011)$$

$$1's \text{ comp} \rightarrow 01001100$$

To find 2's Complement:

$$2's \text{ Comp.} = 1's \text{ Comp.} + 1$$

Ques $(111001011)_2 \rightarrow (00110100)_2$

$(00110101)_2$

→ Base = r ; total r^n ; $N = \text{number}$

(i) r 's comp:

$$(732)_8 \rightarrow \begin{array}{r} 045 \\ +1 \\ \hline 046 \end{array}$$

(ii) $(r-1)$'s comp:

$$(732)_8 \rightarrow \begin{array}{r} 777 \\ -732 \\ \hline 045 \end{array}$$

$$\begin{array}{r} 777 \\ 322 \\ \hline 555 \end{array}$$

→ Need of complement:-

We can perform subtraction with the same circuit

$$20 = 8 \times 2 + 4 \quad [5] + [-3] = 2$$

Steps: [subtraction using 2's complement]

- ① Make no. of digits equal in both number.
- ② find 2's complement the no. to be subtract.
- ③ Add Both of them.
- ④ check added value for carry.

→ if there is carry → then discard carry, remaining will be the answer.

→ if no carry is, there → take 2's complement of added value and put a -ve sign.

→ carry means in answer the digits comes greater than the no. of value.

ie $n \text{ bit} + n \text{ bit} \rightarrow n \text{ bit}$ (no carry will not come)
 $n \text{ bit} + n \text{ bit} \rightarrow (n+1) \text{ bit}$ (carry will come)

Ques: $(11011)_2 - (01110)_2$

$$(11011)_2 + (01110)_2$$

$$\begin{array}{r} 10001 \\ +1 \\ \hline 10010 \end{array}$$

$$\begin{array}{r} 11011 \\ + 10010 \\ \hline (01101)_2 \end{array} = 01101 \text{ Ans}$$

if carry not come

$$\begin{array}{r} 01101 \\ +1 \\ \hline (10011)_2 \end{array}$$

$(1101)_2 - (0111)_2$
 $(1101)_2 + (1001)_2 = \begin{array}{r} 1101 \\ 1001 \\ \hline (10110)_2 \end{array} = 0110 \text{ Ans}$
subtraction 1's Compl.

→ If carry comes then add to that no. again

* Logic Gates

Logic Gate is device which implement a boolean function. They are called gate b/c they control the flow of signal their output is neither 0 or 1, b/c they are the components of logic circuits.

* Two types of Logic Gates :-

→ Basic Gates [AND, OR, NOT]

→ Universal Gate [NOR, NAND]

→ Derived [EX-OR, EX-NOR]

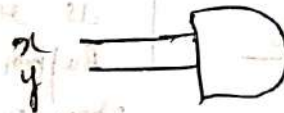
↳ Derived gate are derived from the Basic Gates that's why called derived gate. [error detection, error correction]

* Parity (error)

Whenever data is transmitted from one place to another then to calculate the error we use parity.

* Basic Gates :-

* AND Gate :-



2 or more input with single output

output will be high only when all input are high.

"." is called logical operator or Naming

It perform logical multiplication.

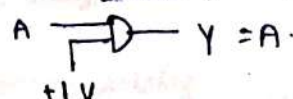
Truth Table (Tabular Representation of I/P / O/P)

I/P		O/P
x	y	z = x.y
0	0	0
0	1	0
1	0	0
1	1	1

"0" → disable, with O/P = 0.



"1" → enable / buffer



OR - Gate : ("+")



Truth Table

I/P		O/P
x	y	$z = x + y$
0	1	1
1	0	1
0	0	0
1	1	1

One or more than 2 input is high the output will be high

Two or more input with single output

enable $\rightarrow$ "0"

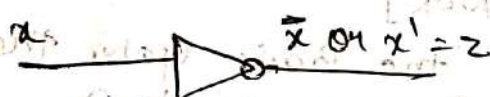


disable $\rightarrow$ "1"



NOT Gate [inversion]

operator "!" or "-"



single input, single output

output will be inverse of input

Truth table

I/P	O/P
0	1
1	0

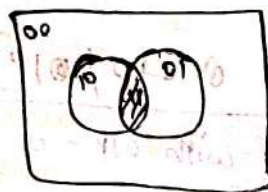
Buffer Gate



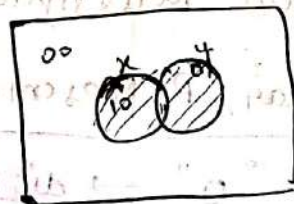
when the strength of signal is reduced then, we use Buffer gate to increase the strength of signal

Venn Diagram for AND, OR, NOT

AND



OR



NOT



when $x = 0$

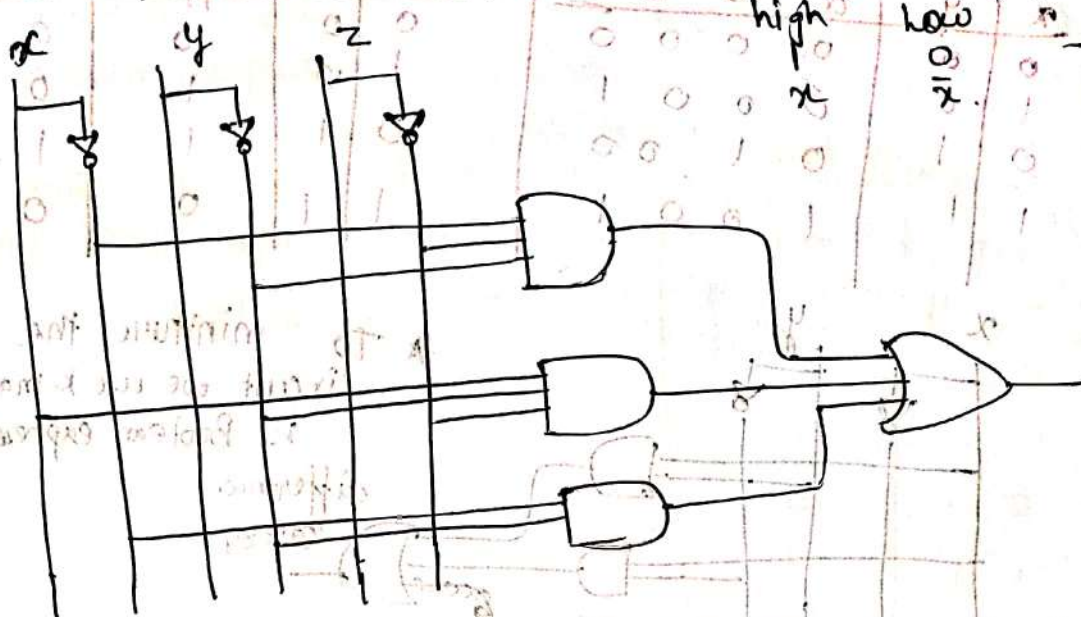


when $y = 0$

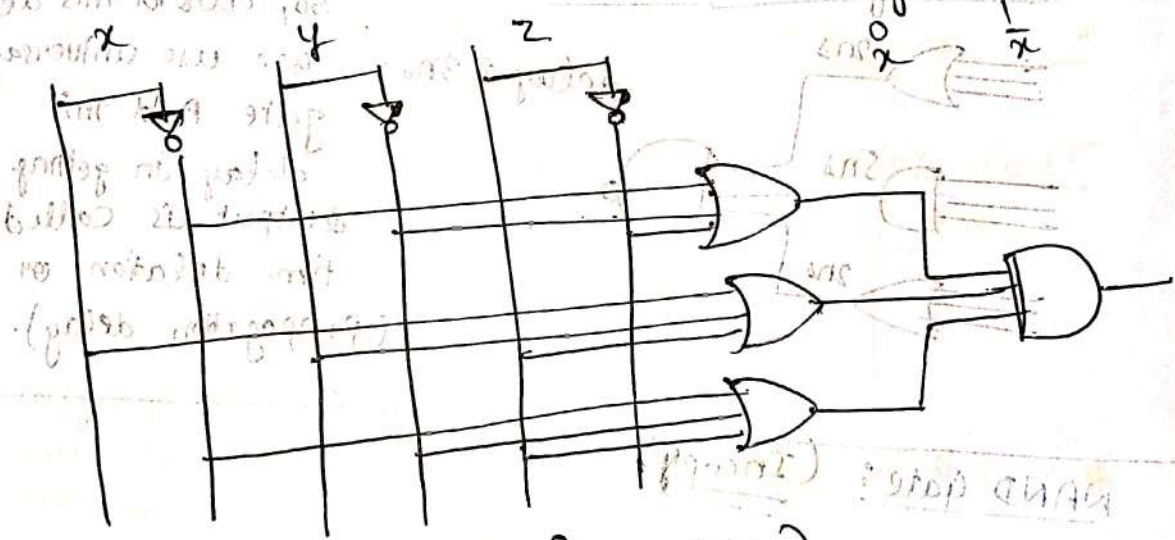
Draw a circuit Diagram

$$(xzy) + (x\bar{y}z) + (\bar{x}y\bar{z})$$

[SOP $\rightarrow$ positive logic]



(ii) $(\bar{x} + y + \bar{z}) \cdot (x + y + z) \cdot (\bar{x} + \bar{y} + z)$ [POS $\rightarrow$ negative logic]



A 2-bit Binary Adder [Half Adder]

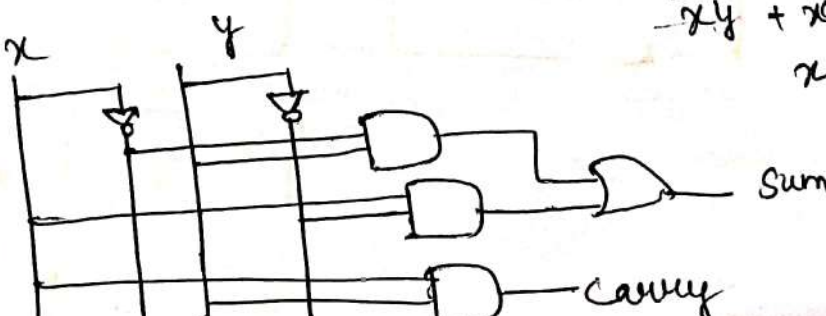
x	y	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



where $x = 1$
 $\bar{x} = 0$

$$\bar{x}y + x\bar{y} = \text{sum}$$

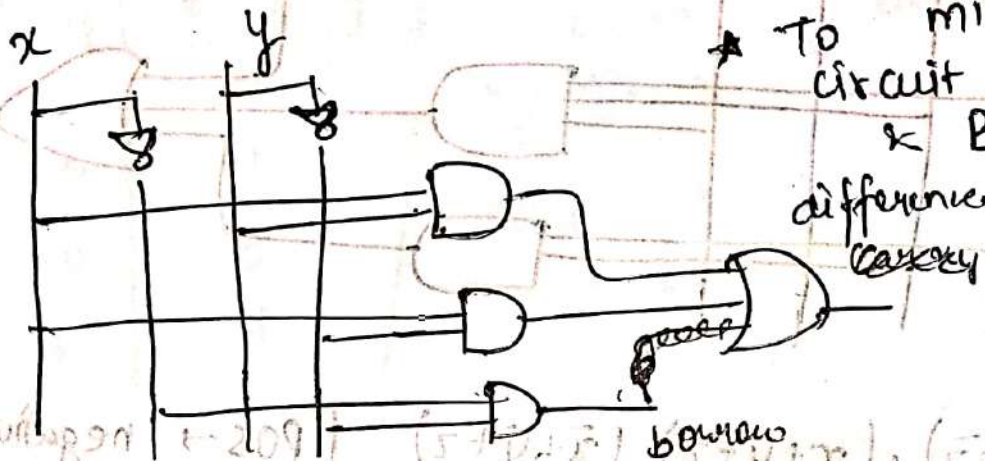
$$xy = \text{carry}$$



2-Bit Binary Squarer Generator

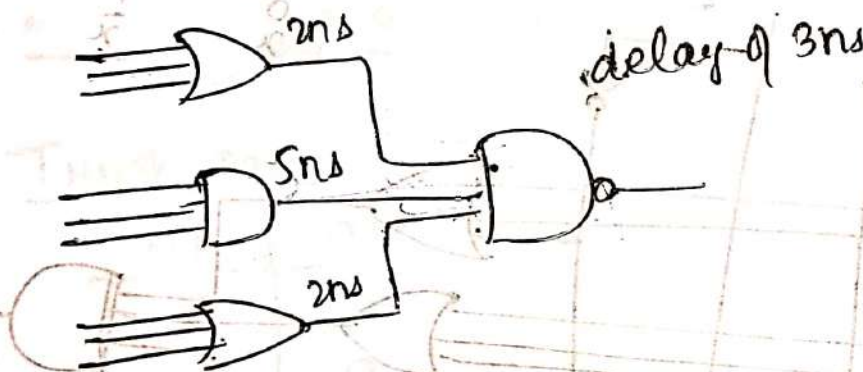
I/P		O/P			
x	y	a	b	c	d
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	1	0	0
1	1	1	0	0	1

I/P		O/P	
x	y	Difference	Borrow
0	0	0	0
1	0	1	0
0	1	1	1
1	1	0	0



* To minimize the circuit we use K-map & Boolean expression.

* Need of Universal Gate:



So, avoid this delay we use universal gate and this delay in getting output is called time delay or (propagation delay).

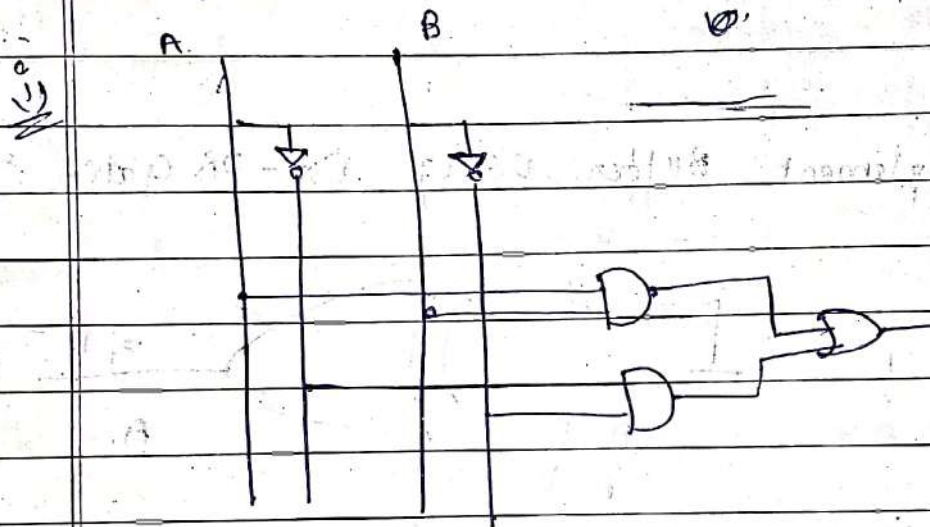
* Implement EX-NOR Gate Using

(i) Basic Gate

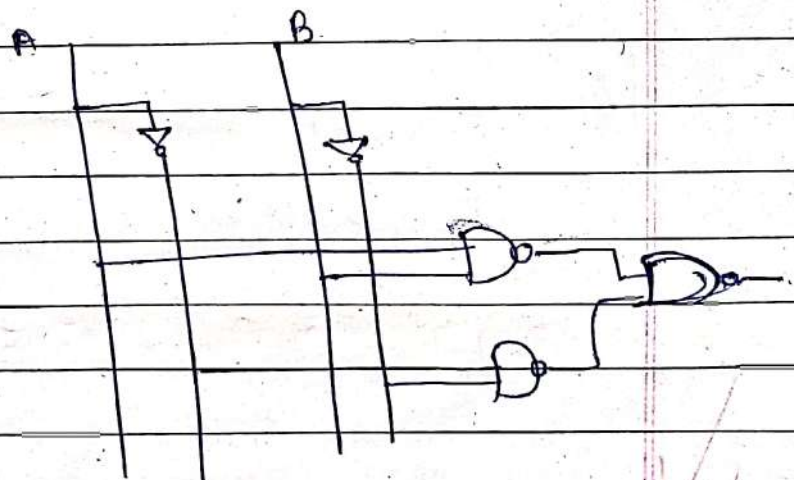
(ii) NAND Gate only

(iii) NOR Gate only

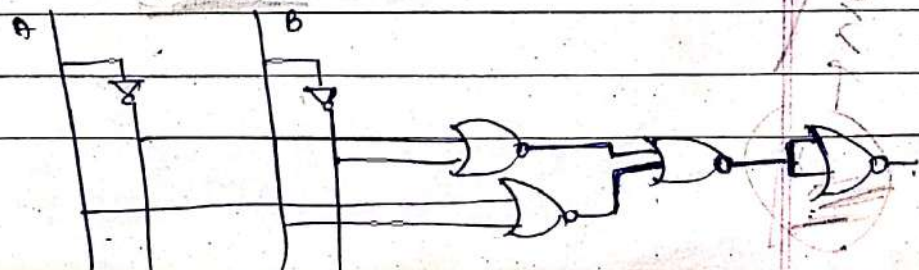
$$\text{EX-NOR} = AB + \bar{A}\bar{B}$$



(ii) NAND $AB + \bar{A}\bar{B}$



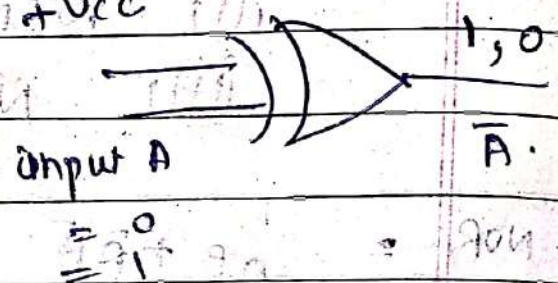
(iii) NOR $AB + \bar{A}\bar{B}$



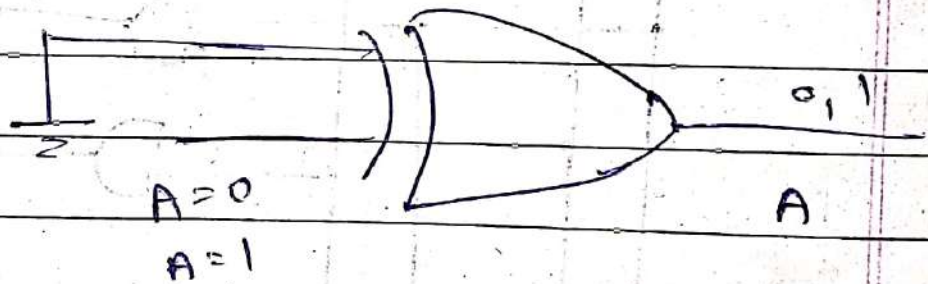
NOT Gate

* Implement ~~using EX-OR~~ using NOT EX-OR Gate

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

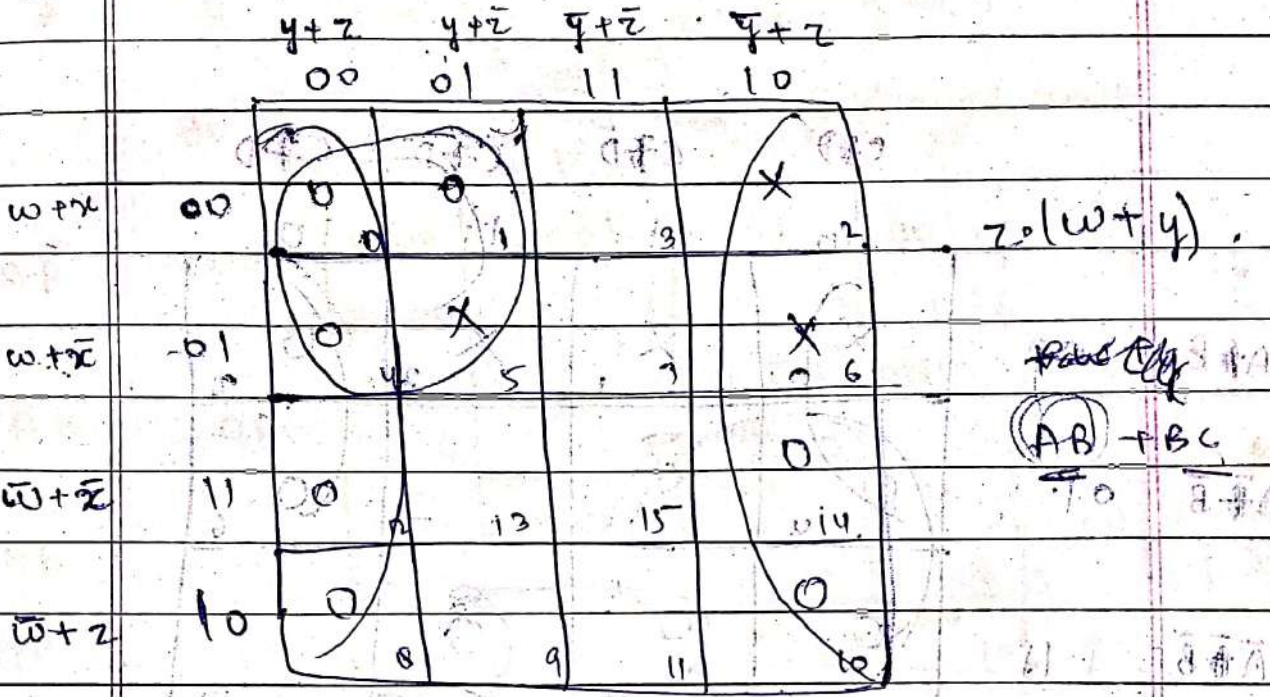


* Implement Buffer using EX-OR Gate

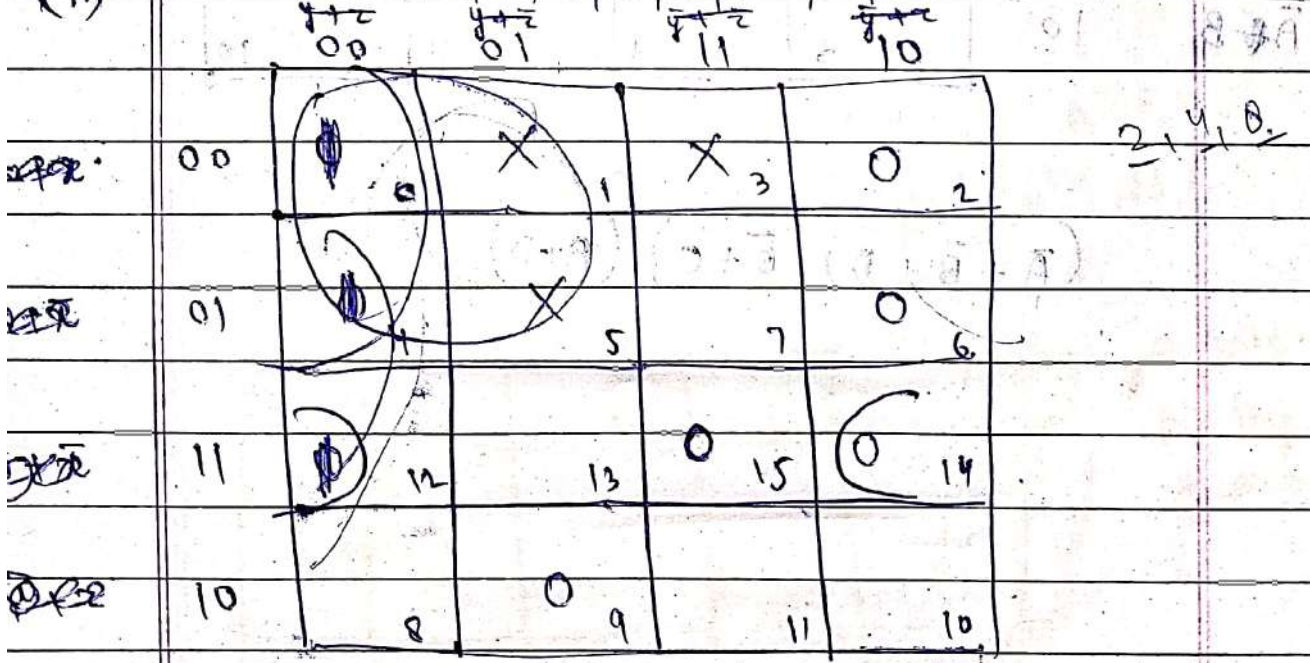


* POS

a) $f(w, x, y, z) = \sum m(0, 1, 4, 8, 10, 12, 14) + d(1, 2, 5, 6)$



ii) $\sum m(0, 2, 4, 6, 9, 12, 14, 15) + d(1, 3, 5)$



2-pair OR 2-quad,
 $\bar{B}\bar{C}D + ABC + \bar{A}\bar{D} + \bar{A}\bar{D}$

(iii) $\Sigma M(2, 4, 6, 12, 13, 15) \cdot d(0, 1, 3, 5)$

		$C \rightarrow D$ 00	$C \rightarrow \bar{D}$ 01	$\bar{C} \rightarrow \bar{D}$ 11	$\bar{C} \rightarrow D$ 10
$A \rightarrow B$ 00		X 0	X 1	X 3	0 2
$A \rightarrow \bar{B}$ 01		0 4	X 5		0 6
$\bar{A} \rightarrow \bar{B}$ 11		0 12	0 13	0 15	
$\bar{A} \rightarrow B$ 10					

$$(\bar{A} + \bar{B} + D)(\bar{B} + C)(A + D)$$

K-Map (K $\rightarrow$ Karnau)

* (4-variable) term $2^4 = 16$.

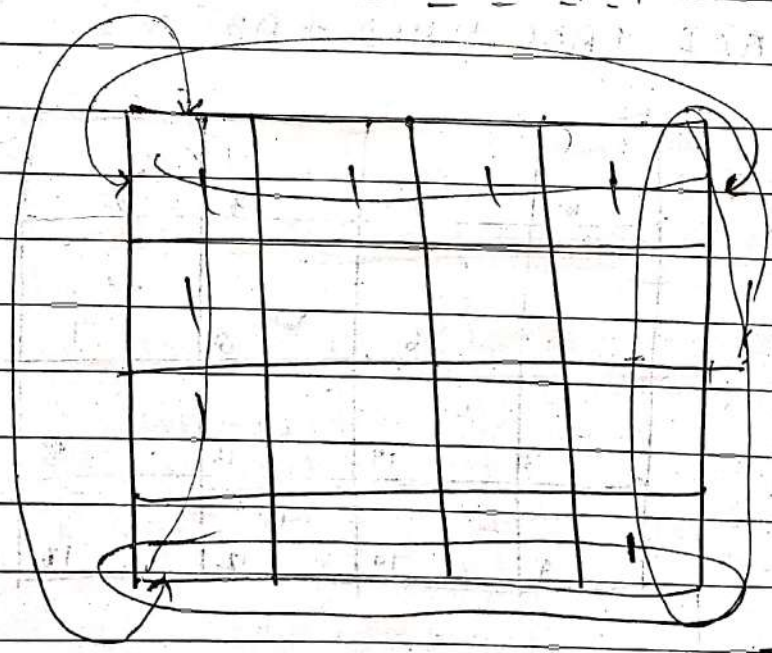
↓ AB/CD most significant (rows)

	CD	$\bar{C}\bar{D}$	$\bar{C}D$	$C\bar{D}$	CD
$\bar{A}\bar{B}$	00	0000	0001	0011	0010
$\bar{A}B$	01	0100	0101	0111	0110
$A\bar{B}$	11	1100	1101	1111	1110
AB	10	1000	1001	1011	1010

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

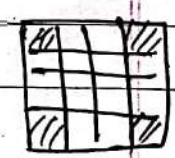
example!

$f = ABCD + AB\bar{C}D + A\bar{B}CD$
 $\Sigma m (0, 1, 2, 3, 4, 10, 14, 15)$

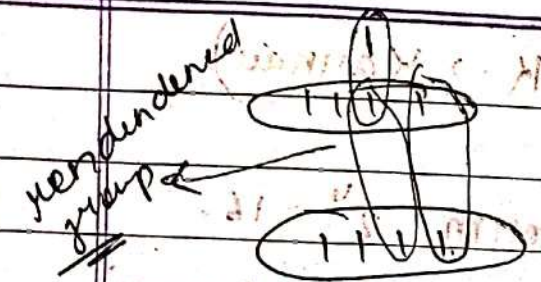


structure
filling
grouping

$\rightarrow$ all bits should be adjacent to each other.



$\rightarrow$ No. of bits in group is 1, 2, 4, 8, 16.



ABCD

$$y = \sum m(1, 5, 6, 7, 11, 12, 13, 15)$$

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

Handwritten annotations on the K-map:
 - Group 1: $\bar{A}\bar{B}B$ (cells 1, 5)
 - Group 2: $\bar{A}B$ (cells 5, 7)
 - Group 3: AB (cells 7, 15)
 - Group 4: $A\bar{B}C$ (cells 13, 15)
 - Group 5: $A\bar{B}D$ (cells 11, 13)
 - Group 6: $AB\bar{C}$ (cells 11, 15)
 - Group 7: $AB\bar{D}$ (cells 13, 15)

$$A\bar{C}D + \bar{A}BC + ACD + AB\bar{C}$$

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Handwritten checkmarks (✓) in the following cells: 1, 5, 7, 11, 13, 15.

$$A + \bar{B}\bar{C}D + AB\bar{C}D + \bar{C}\bar{D}$$

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{A}\bar{B}$	0	1	3	2
$\bar{A}B$	4	5	7	6
AB	12	13	15	14
$A\bar{B}$	8	9	11	10

$$\Sigma m(1, 3, 12, 13, 14, 15, 8, 9, 11, 10)$$

$$\Sigma m(0, 1, 4, 5, 9, 10, 11, 12, 13, 14, 15)$$

$$\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} +$$

(iii)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	3	2
$\bar{A}B$	4	5	7	6
AB	12	13	15	14
$A\bar{B}$	8	9	11	10

Don't care :-

Used for to minimise the circuit

1 → pair
1 → quad
2 → quad

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0 X	1 1	3 1	2 X
$\bar{A}B$	4 X	5 1	7 1	6 0
AB	12 0	13 0	15 1	14 0
$A\bar{B}$	8 0	9 0	11 1	10 0

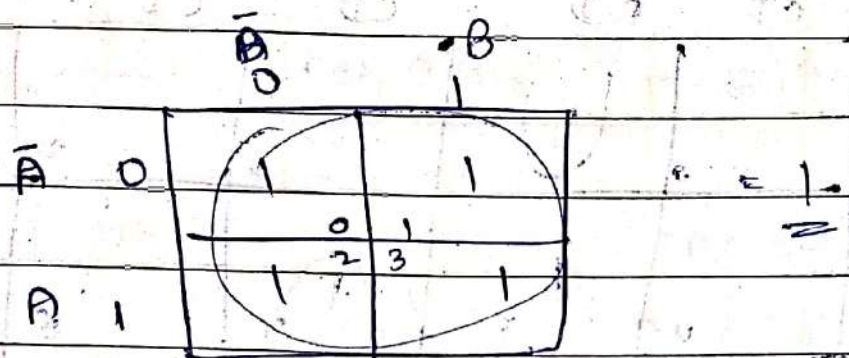
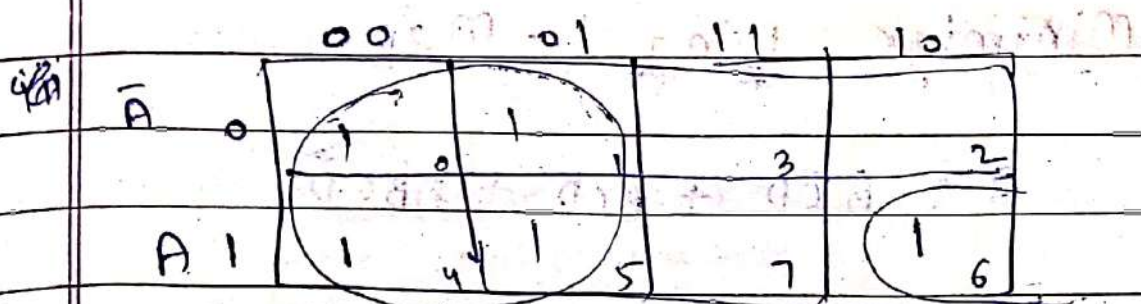
$$\Sigma m(1, 3, 7, 11, 15) + d(0, 2, 5)$$

$$\bar{A}D + CD$$

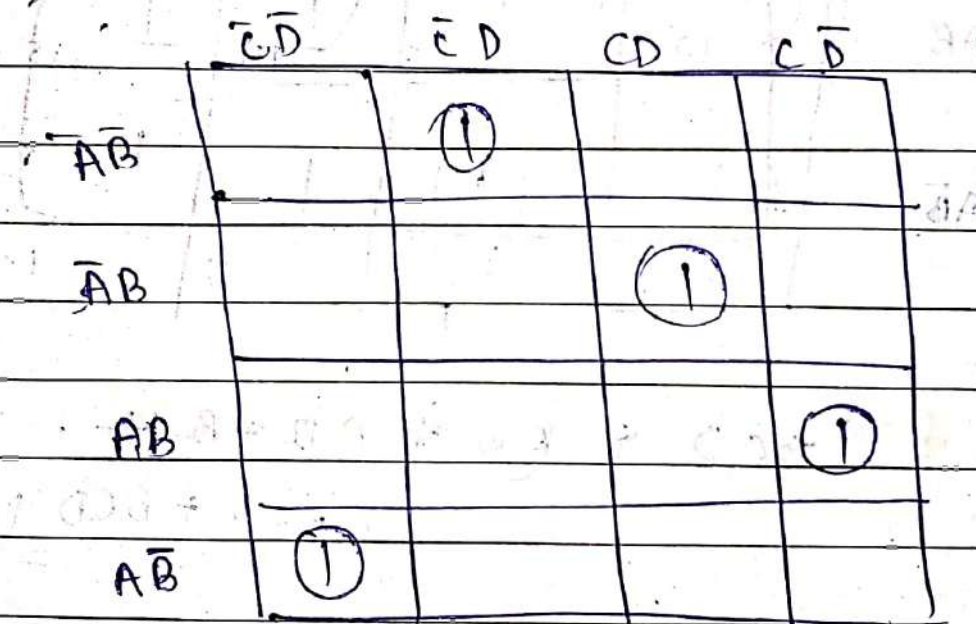
If there is No. is repetitive in don't care then exclude from don't care and include in prim term and max term

(2) $\Sigma m(1, 4, 8, 12, 13, 15) + d(3, 14)$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0 0	1 1	3 X	2 0
$\bar{A}B$	4 1	5 0	7 0	6 0
AB	12 1	13 1	15 1	14 X
$A\bar{B}$	8 1	9 0	11 0	10 0



Ques



*

* Minimize using K-Map

$$A + \bar{B}\bar{C}D + BCD + A\bar{B}\bar{C}D$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	3	2
$\bar{A}B$	4	5	7	6
AB	12	13	15	14
$A\bar{B}$	8	9	11	10

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

$$\bar{B}\bar{C}D + BCD + A$$

* Minimize Find min term and max term using the given exp. (always give expres

$$A + \bar{B}\bar{C}D + BCD + A\bar{B}\bar{C}D$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	3	2
$\bar{A}B$	4	5	7	6
AB	12	13	15	14
$A\bar{B}$	8	9	11	10

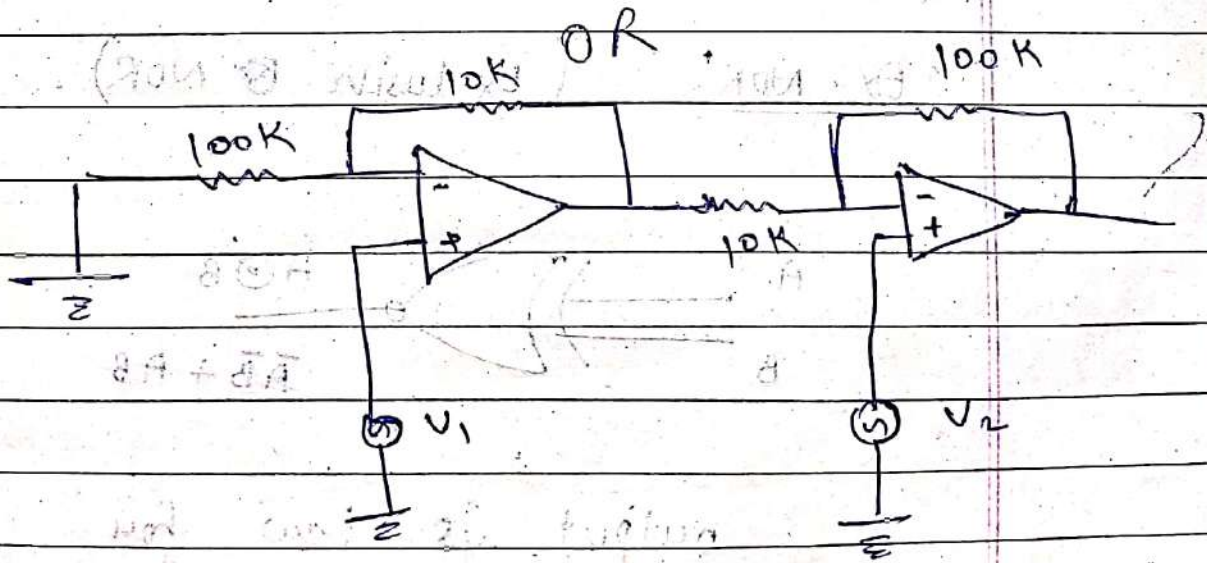
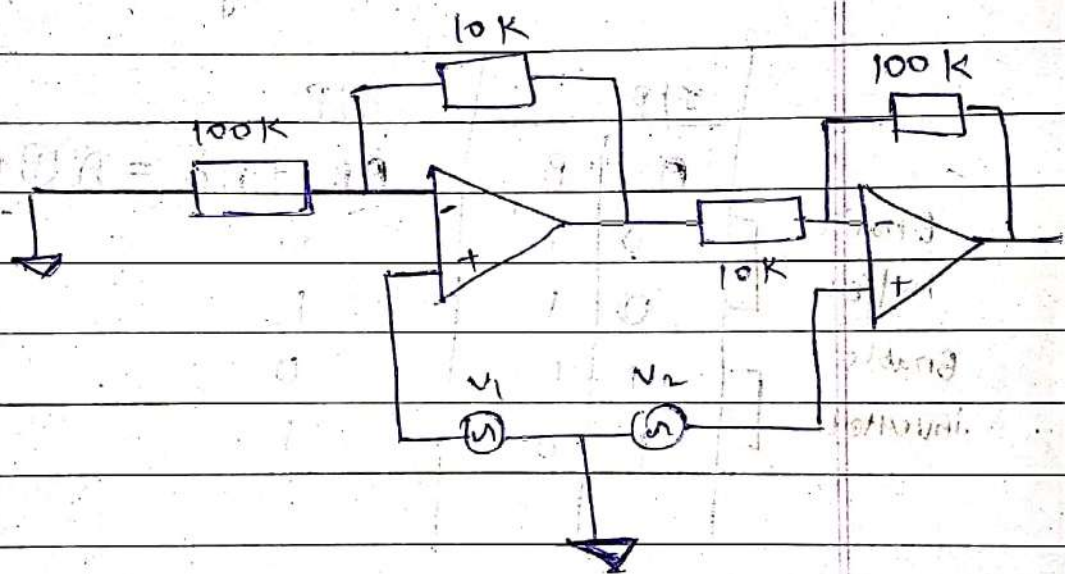
min

em (1, 7, 13, 10, 14, 15, 9, 11, 12)

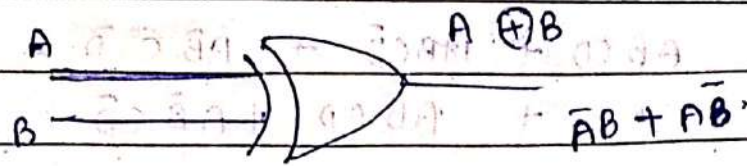
$$\bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + A\bar{B}\bar{C}D + A\bar{B}CD + AB\bar{C}D + ABC\bar{D} + AB\bar{C}\bar{D} + AB\bar{C}D + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + A\bar{B}\bar{C}D + ABC\bar{D}$$

max term :- $(0, 1, 2, 3, 4, 5, 6)$

$$(A+B+C+D)(A+B+\bar{C}+D)(A+B+\bar{C}+\bar{D})(A+\bar{B}+C+D)(A+\bar{B}+C+\bar{D})(A+\bar{B}+\bar{C}+D)$$



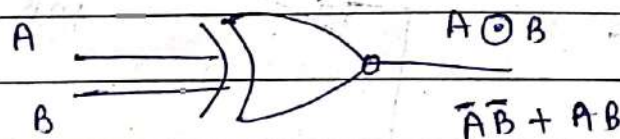
(X-OR OR) EX - OR (Exclusive - OR).



more than
we avoid two input. To check
whether it
is even or
odd
output is high for
odd no. of high input.

I/P		O/P	
A	B	$\bar{A}B + A\bar{B} = A \oplus B$	
Enable buffer	0	0	0
	1	1	1
Enable inverter	1	1	0
	0	0	1

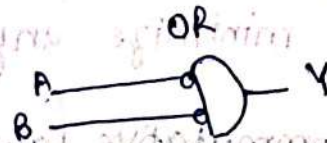
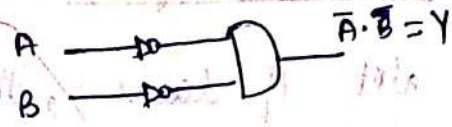
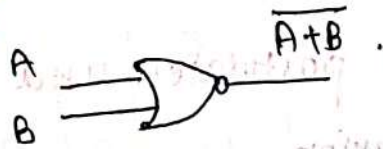
Ex - NOR (Exclusive NOR).



output is low for
odd no. of high input

I/P		O/P	
A	B	$\bar{A}\bar{B} + AB$	
Enable inverter	0	0	1
	1	1	0
Enable buffer	1	1	1
	0	0	0

NOR GATE (Bubbled AND) another way to represent NOR gate is $\overline{A+B} = \overline{A} \cdot \overline{B}$



Truth Table

I/P		O/P	
A	B	A.B	A+B
0	0	0	1
1	0	0	0
0	1	0	0
1	1	1	0

disable
o/p
= 0

"If any of the input is high then output is low"

enable
inverter:

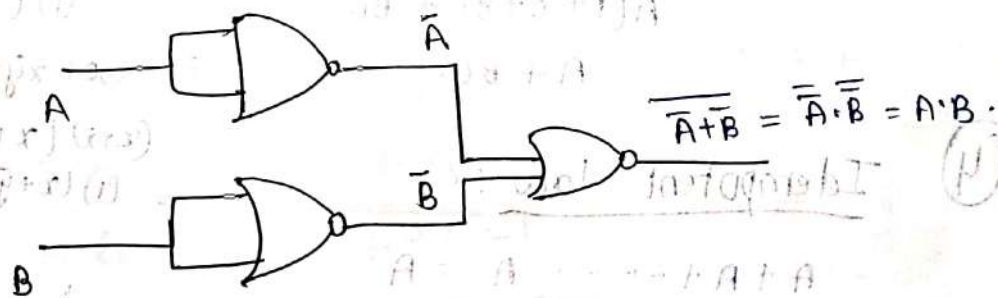
Associative law $\times$
Comm. law $\checkmark$

(Used I/P) : same as OR gate

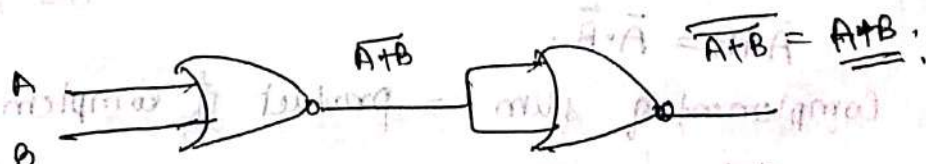
* Implement Not Gate using NOR Gate only



* Implement AND Gate using NOR Gate only



* Implement OR Gate using NOR GATE only



Boolean Algebra :-

Set of laws, theorem and postulates used to minimize any logic expression.

① Commutative law :-

$$A+B = B+A$$

$$A \cdot B = B \cdot A$$

② Associative law :-

$$A+(B+C) = (A+B)+C$$

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

③ Distributive Law :-

$$A(B+C) = AB+AC$$

$$\# \quad A(B+C) = AB+AC \quad (A+B)(A+C) = A+BC$$

Proof:

$$(A+B)(A+C)$$

$$A \cdot A + A \cdot C + B \cdot A + B \cdot C$$

$$A + A \cdot C + A \cdot B + B \cdot C$$

$$A(1+C+B) + BC$$

$$A + BC$$

Use:-

$$1. \quad x + \bar{x}y =$$

$$= (x + \bar{x})(x + y)$$

$$= (1)(x + y)$$

$$2. \quad (x + \bar{x}y)z =$$

$$(x + \bar{x})(x + y)z$$

$$= (1)(x + y)(x + z)$$

④ Idempotent law :-

$$A + A + \dots + A = A$$

$$A \cdot A \cdot \dots \cdot A = A$$

⑤ De Morgan's Theorem :-

$$\overline{A+B} = \bar{A} \cdot \bar{B}$$

Complement of sum = product of complement

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

Proof:-

				Proved 1st Theorem		Proved 2nd Theorem	
A	B	$\bar{A}$	$\bar{B}$	$A+B$	$\bar{A} \cdot \bar{B}$	$A \cdot B$	$\bar{A} + \bar{B}$
0	0	1	1	0	1	0	1
0	1	1	0	0	0	0	1
1	0	0	1	0	0	0	1
1	1	0	0	1	0	1	0

not applied to minimize

Duality Theorem:- To find dual of logical expressions Replace

AND $\rightarrow$ OR

OR $\rightarrow$ AND

0 $\rightarrow$ 1

1 $\rightarrow$ 0

AND law

$$0 \cdot 0 = 0$$

$$1 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 1 = 1$$

$$A \cdot 0 = 0$$

$$A \cdot 1 = A$$

$$A \cdot A = A$$

$$A \cdot \bar{A} = 0$$

$$A \cdot \bar{A} = 0$$

OR law

$$0+0=0$$

$$0+1=1$$

$$1+0=1$$

$$1+1=1$$

$$A+0=A$$

$$A+1=1$$

$$A+A=A$$

$$A+\bar{A}=1$$

NOT law

$$\bar{0}=1$$

$$\bar{1}=0$$

$$\bar{\bar{A}}=A$$

Use complement

$A + AB = A(1+B) = A \cdot 1$ [minimization of circuit will be like this]

Logic Expression

* sum of product

- Terms are formed by ANDing the variables.
- Expression is formed by ORing the terms

→ Uncomplement variable $A \rightarrow 1$

→ complemented variable $\bar{A} \rightarrow 0$

variable
↓ AND
terms

↓ OR
expression

* Product of Sum

→ Terms are formed by ORing the variables

→ Expressions are formed by ANDing the terms

→ uncomplement variable $A \rightarrow 0$

→ complement variable $\bar{A} \rightarrow 1$

variables

↓ OR

Terms

↓ AND
expression

Canonical SOP :-

Each term should contain all variable A
either complemented or uncomplemented

Canonical SOP :-

Each term of min term contain all variable
either complemented or uncomplemented

min term :- Each term of canonical SOP

is known as minterm. Denoted by "m".

example:- $\bar{A}\bar{B}C + A\bar{B}C + ABC = F(A, B, C)$

$m_1 \quad m_2 \quad m_3$

Ques. Find minterm for the given expression
OR

Convert it into canonical expression.

$$AB + A\bar{B}\bar{D} + ABD = F(A, B, C, D)$$

$$F(A, B, C, D) = AB + A\bar{B}\bar{D} + ABD$$

$$= AB(C + \bar{C}) + A\bar{B}\bar{D}(C + \bar{C}) + ABD(C + \bar{C})$$

$$= ABC + A\bar{B}\bar{C} + A\bar{B}C\bar{D} + A\bar{B}\bar{D}\bar{C} + ABCD + AB\bar{C}\bar{D}$$

$$= ABC(D + \bar{D}) + A\bar{B}\bar{C}(D + \bar{D}) + A\bar{B}C\bar{D} + A\bar{B}\bar{D}\bar{C} + ABCD + AB\bar{C}\bar{D}$$

$$= ABCD + ABC\bar{D} + A\bar{B}\bar{C}D + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + A\bar{B}\bar{D}\bar{C} + ABCD + AB\bar{C}\bar{D}$$

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

$$F(A, B, C, D) = \sum m(0, 9, 10, 11, 13, 15)$$

Canonical POS

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

$M_1 \quad M_2 \quad M_3$

→ Max term :- Each term of canonical POS is known as maxterm. Denoted by "M".

→ Each term of maxterm contain all variable either complemented or uncomplemented.

Ques. Find maxterm and minterm for the expression.

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

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

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

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

~~F(A, B, C)~~ 014

$$F(A, B, C) = \pi M(0, 1, 4)$$

$$F(A, B, C) = \sum m(2, 3, 5, 6, 7)$$

expression of POS:-

$$m_2 + m_3 + m_5 + m_6 + m_7$$

010 011 101 110 111

$$\boxed{\text{SOP} \rightarrow \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC}$$

$$\text{POS} \rightarrow M_0 + M_1 + M_4$$

$$\boxed{\text{POS} \rightarrow ABC + AB\bar{C} + \bar{A}B\bar{C}}$$

$$\text{POS} \rightarrow (A+B+\bar{C})(A+B+C)(\bar{A}+B+C)$$

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

Ques

Minimize given expression using Boolean algebra :-

① $AB + A(B+C) + B(B+C)$

$$AB + AB + AC + B(B+C)$$

$$AB + AB + AC + B + BC$$

$$AB + AB + AC + B + BC$$

$$AB + B + BC + AC$$

$$B(A+C+1) + AC$$

$$= B + AC$$

$$AB + AC + B + BC$$

$$B + AC$$

② $\bar{A} + \bar{B} + AB\bar{C}$

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

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

$$\bar{A} + \bar{B}\bar{C} + \bar{B}$$

$$\bar{A} + \bar{B}\bar{C} + \bar{B}$$

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

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

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

③ $(A+B)'(A'+B)'$

$$(\overline{A+B})(\overline{A'+B})$$

$$= (\bar{A} \cdot \bar{B})(\bar{A} \cdot B)$$

$$= \bar{A} \cdot A \cdot B \cdot \bar{B}$$

$$= 0$$

④

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

$$\bar{A}B\bar{D} + \bar{A}B\bar{C}D + BA + \bar{A}B + B\bar{A}CD$$

$$\bar{A}B(\bar{D} + \bar{C}D + CD) + BA$$

$$\bar{A}B(\bar{D} + C(\bar{D} + D) + CD) + BA = \bar{A}B(\bar{D} + C + CD) + BA$$

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

$$\bar{A}B(1+\bar{D}) + AB$$

$$\bar{A}B + AB = B(A + \bar{A}) = \underline{B}$$

ques. [Concept]. $(1+0)B + (0+0)A + 0A$

4 → variable (A, B, C, D) $0A + 0A + 0A$

if (A, B, D) then C is absent (all the alphabets are in continuation.)

$$2^4 = 16$$

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

$$B = (0 + \bar{A} + A)B + (0 + 0 + \bar{A} + A)\bar{B}$$

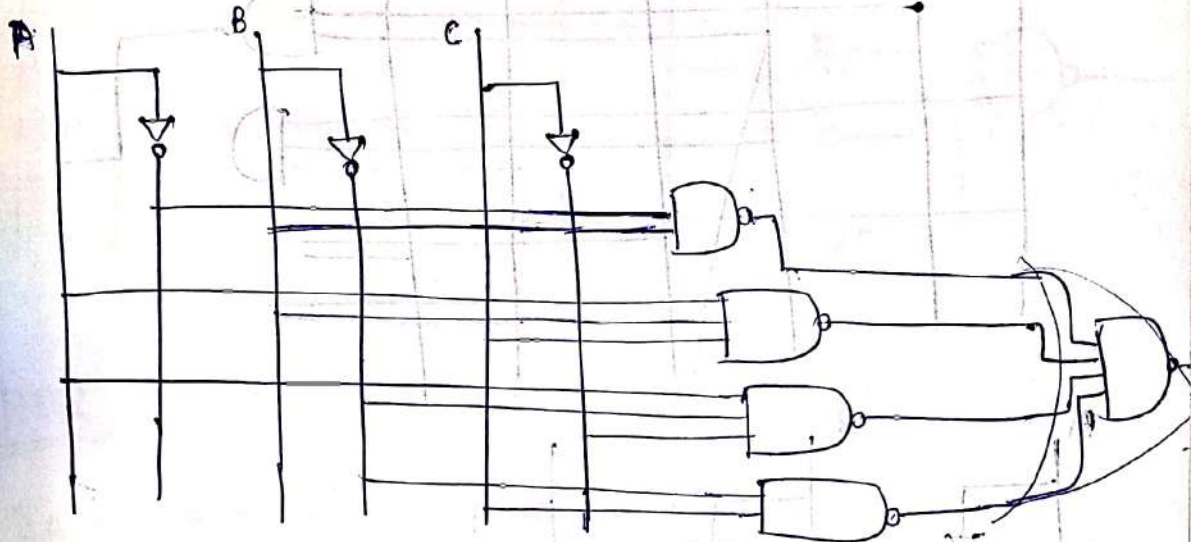
SQ 222

NAND $\rightarrow$ SOP

NOR $\rightarrow$ POS

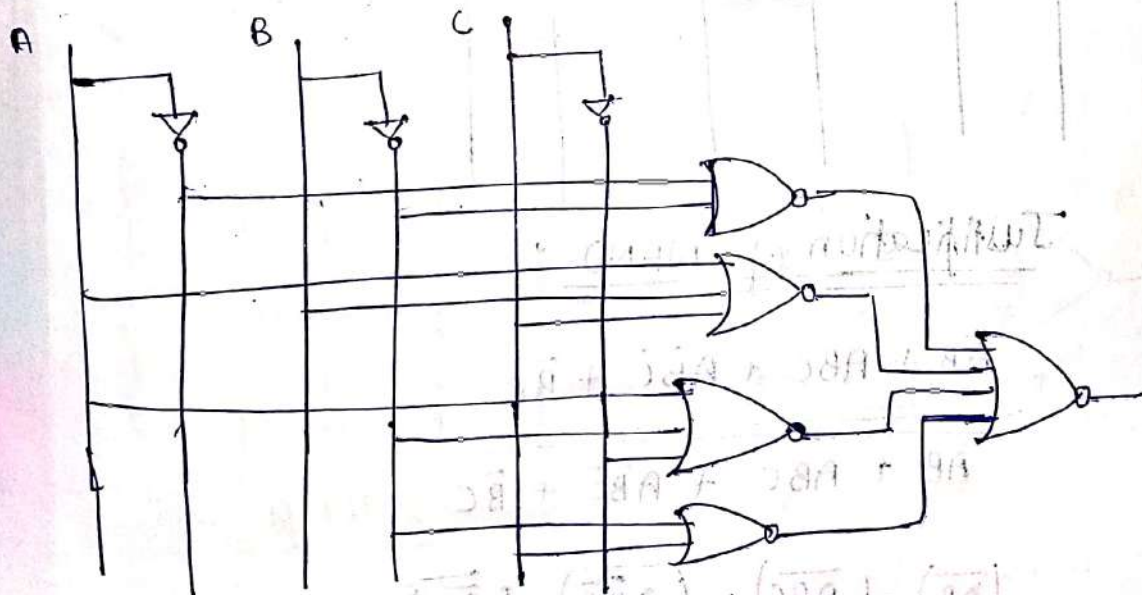
NAND

$$\bar{A}B + ABC + A\bar{B}\bar{C} + \bar{B}C = \text{SOP}$$



~~NAND~~ NOR $(\bar{A}+B)(A+B+C)(A+\bar{B}+\bar{C})(\bar{B}+C) = \text{POS}$

~~$\bar{A}B + ABC + A\bar{B}\bar{C} + \bar{B}C = \text{POS}$~~



Justification of NAND :-

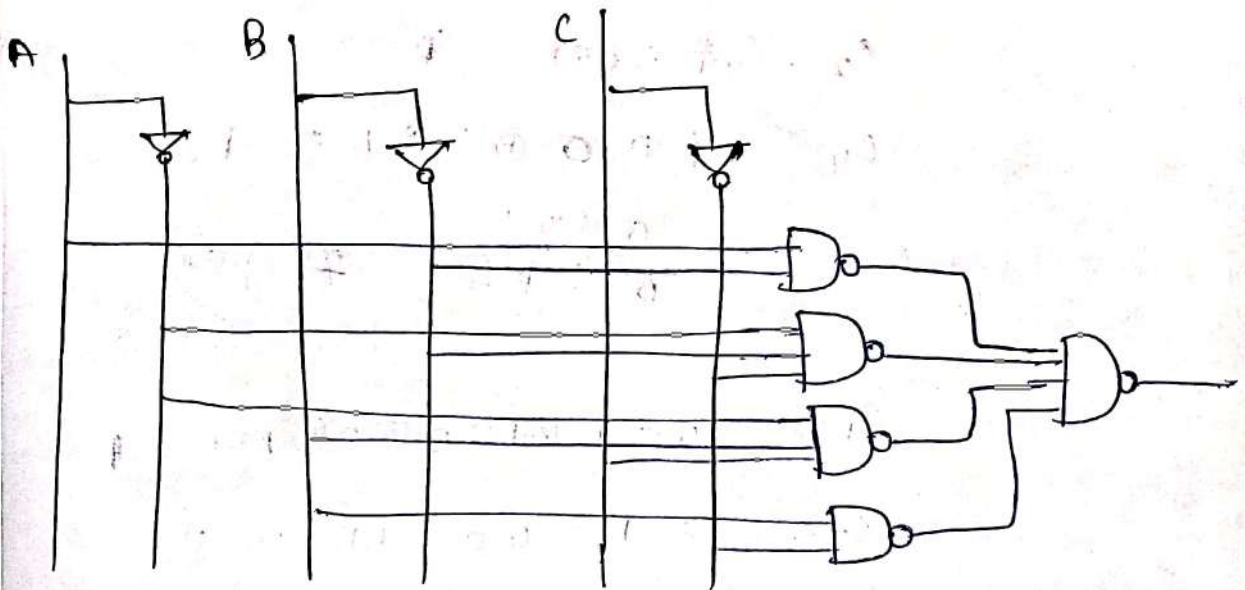
$$\bar{A}B + ABC + A\bar{B}\bar{C} + \bar{B}C$$

$$\bar{A}B + ABC + A\bar{B}\bar{C} + \bar{B}C$$

$$(\bar{A}B) \cdot (ABC) \cdot (A\bar{B}\bar{C}) \cdot (\bar{B}C)$$

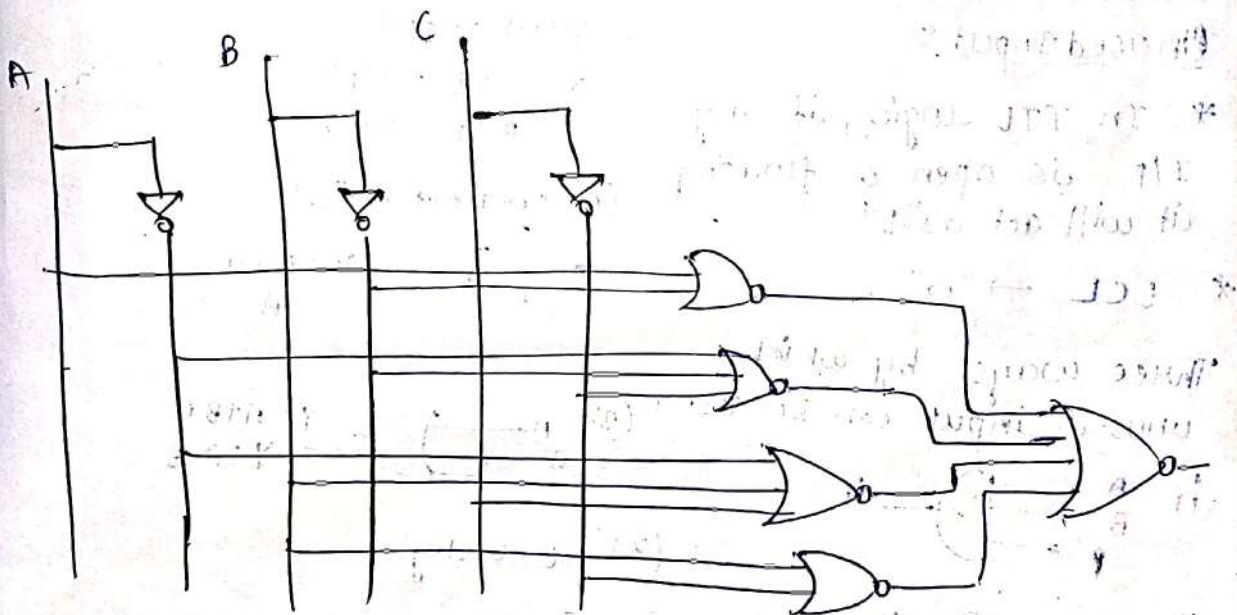
NAND

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



NOR

$$\bar{A}B + ABC + A\bar{B}\bar{C} + \bar{B}C = \text{SOP}$$



Justification of NOR

$$\bar{A}B + ABC + A\bar{B}\bar{C} + \bar{B}C$$

$$\overline{\overline{\bar{A}B}} + \overline{\overline{ABC}} + \overline{\overline{A\bar{B}\bar{C}}} + \overline{\overline{\bar{B}C}}$$

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

1 0 1 0 1
7 2 3 4 5 6 7

$201 = (1 \oplus 1) \oplus 1 = 1$

$C_1 = 1 \oplus 1 \oplus 1 = 1$

$C_2 = 0 \oplus 1 \oplus 0 \oplus 1 = 0$

$C_4 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$

8 4 2 1
0 1 0 1

$C_1 = 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 1$

$C_2 = 0 1 0 0 1 1 = 0$

$C_4 = 1 1 0 0 0 1 = 0$

$C_8 = 1 1 0 0 0 1 = 0$

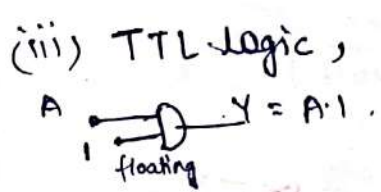
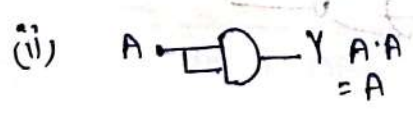
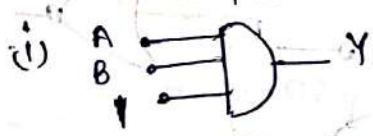
AND Gate

Unused Input:

* In TTL logic, if any I/p is open or floating it will act as "1"

* ECL $\rightarrow$ "0"

Three ways by which unused input can be used



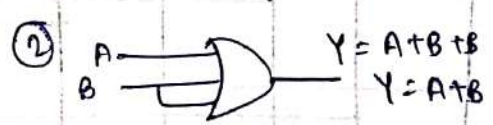
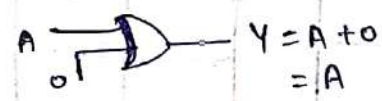
ECL logic not applied b/c it is disabled when it is 0.

OR Gate

Used input

* TTL $\rightarrow$ "1"
ECL $\rightarrow$ "0"

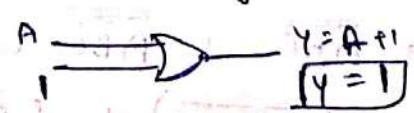
① connect to "0"



③ ECL logic



TTL logic



Hence answer is not floating so, not used.

Introduction To Boolean Algebra :-

word comes from George Boole

★ There are some set of rules to minimize the boolean expression without changing its functionality.

AND

OR

$$A \cdot A = A$$

$$A + A = A$$

$$A \cdot 0 = 0$$

$$A + 0 = A$$

$$A \cdot 1 = A$$

$$A + 1 = 1$$

$$A \cdot A' = 0$$

$$A + A' = 1$$

Distributive law :-

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

$$A + BC = (A + B)(A + C)$$

Commutative law :-

$$A + B = B + A$$

$$A \cdot B = B \cdot A$$

Associative law :-

$$(A \cdot B) \cdot C = A \cdot (B \cdot C)$$

V.V-Imp

Priority

NOT

AND

OR

Algebra will be solve in this priority manner.

De Morgan's law :-

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

Redundancy TheoremRules

①

$$Y = AB + A'C + BC$$

$$Y = AB + A'C$$

$$Y = A \cdot B + A'C + B \cdot C \cdot 1$$

$$Y = A \cdot B + A'C + B \cdot C(A + A')$$

$$Y = A \cdot B + A'C + A \cdot B \cdot C + A'BC$$

$$Y = AB[1 + C] + A'C[1 + B]$$

$$Y = AB + A'C$$

(i) Three variables

(ii) Each variable is repeated twice one variable is complimented

(iv) Take the complimented variable

②

$$f = \bar{A}\bar{B} + A\bar{C} + \bar{B}\bar{C}$$

$$f = \bar{A}\bar{B} + A\bar{C}$$

③ $f = (A+B)(\bar{A}+C)(B+C)$

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

Sum of Product :- $\bar{A} \rightarrow 0$ (Min terms) $A \rightarrow 1$ (b/c of its size)
($A \cdot B \cdot C$)Total no. of comb. = $2^n = 2^3 = 8$

$$f = \bar{A} \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot \bar{C} + A \cdot \bar{B} \cdot C + A \cdot B \cdot \bar{C} + A \cdot B \cdot C \quad \text{[sol form]}$$

$$\begin{aligned} f &= \bar{A}B\bar{C} + A \\ &= (\bar{A} + A)(A + B\bar{C}) \\ &= A + B\bar{C} \end{aligned}$$

Product of Sum $A \rightarrow 0$ (Max terms) $\bar{A} \rightarrow 1$ (b/c of its size)

$$F = (A+B+C) \cdot (A+B+\bar{C}) \cdot (A+\bar{B}+\bar{C}) \quad \text{(standard form)} \quad (A+B+C)$$

$$Y = A$$

(minimal form)

SOP $\rightarrow$ has more AND GatePOS $\rightarrow$ has more OR GateQues. for the given Truth Table minimize the POS expression.

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

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

$$Y = (A + \bar{B}) \cdot (\bar{A} + \bar{B})$$

$$= \bar{B} + 0$$

$$Y = \bar{B} \quad \text{(minimal form)}$$

$$Y = \sum m(1, 3) \text{ or } \sum m(m_1, m_3)$$

$$Y = \sum m(0, 2)$$

SOP form

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$Y = \sum m(0, 2, 3, 6, 7)$$

$$Y = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C} + ABC \quad \left[\begin{array}{l} \text{canonical} \\ \text{form} \end{array} \right]$$

$$Y = \bar{A}\bar{B}\bar{C} + \bar{A}B(\bar{C} + C) + AB\bar{C} + ABC$$

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

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

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

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

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

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

POS form

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

Ex. for $n=4$, what is the total no. of logical expression.
 ↳ no. of variables.

formula = 2^{2^n} where $n = \text{no. of variables}$

$$\therefore 2^{2^4} = 2^{16} = 65536$$

Positive logic

(High) 1 volt $\rightarrow 1$
 (Low) 0 volt $\rightarrow 0$

logic 0 $\rightarrow -5V$
 logic 1 $\rightarrow 0V$ } Positive logic

Negative logic

(Low) 0 volt $\rightarrow 1$
 (High) 5 volt $\rightarrow 0$

0 $>$ -5
 1 $>$ 0 (positive logic)
 0 $<$ 1 (negative logic)

+ logic AND Gate = - logic OR Gate
 - logic AND Gate = + logic OR Gate

$\rightarrow$ Dual form:-

means to convert it into negative to positive and vice-versa.

$A \cdot B \xrightarrow{\text{Dual}} A + B$
 (+)

$A + B \xrightarrow{\text{Dual}} A \cdot B$

Not complemented so, this is difference b/w complement & Dual form

Self Dual

On one time dual if it give same expression then self Dual,

$F = A \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot C + A \cdot \bar{B} \cdot C$ $F' = (A+B+\bar{C}) (\bar{A}+B+C) \cdot (A+\bar{B}+C)$ $F'' = A \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot C + A \cdot \bar{B} \cdot C$	$F = A \cdot B + B \cdot C + A \cdot C$ $F' = (A+B) \cdot (B+C) \cdot (A+C)$ $= (B+AC) \cdot (A+C)$ $= A \cdot B + A \cdot A \cdot C + B \cdot C + A \cdot C \cdot C$ $= A \cdot B + A \cdot C + B \cdot C + A \cdot C$ $= A \cdot B + A \cdot C + B \cdot C$
--	---

** If we have n variables we have to find out ^{how many} self dual are possible then,

$$n \text{ variables} = 2^{2^n - 1}$$

$$n = 2$$

$$\text{No. of self dual} = 2^{2^2 - 1} = 2^3 = 8$$



$$A \rightarrow A \quad B \rightarrow \bar{B}$$

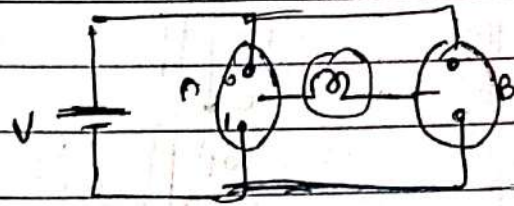
$$\bar{A} \rightarrow \bar{A} \quad \bar{B} \rightarrow B$$

XOR is complement of XNOR

Switching circuit

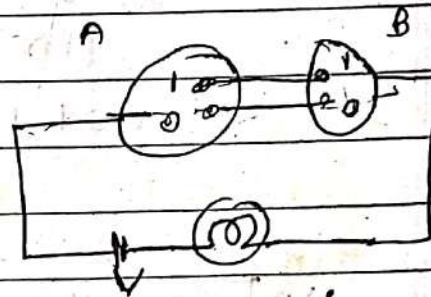
Ex OR

A	B	$A'B + B'A = Y$
0	0	0
1	0	1
0	1	1
1	1	0



Ex NOR

A	B	Y
0	0	1
1	0	0
0	1	0
1	1	0



Number system

Weighted

ex. Decimal
Binary
Octal
BCD etc

Unweighted

ex Gray code
Excess-3

* Decimal to Binary
(22.07)₁₀ →

(010 010.000)₂
111

* Hexadecimal to binary and then octal

Binary Division

$$\begin{array}{r}
 110 \overline{) 101010 \ 0001} \\
 \underline{10} \\
 0 \\
 \underline{101} \\
 0 \\
 \underline{1010} \\
 - 110 \\
 \hline
 1001 \\
 \underline{110} \\
 0011
 \end{array}$$

$$1 < 110$$

$$10 < 110$$

$$101 < 110$$

$$1010 > 110$$

$$\begin{array}{r}
 110 \overline{) 101010 \ 000111} \\
 \underline{10} \\
 0 \\
 \underline{101} \\
 0 \\
 \underline{1010} \\
 - 110 \\
 \hline
 1001 \\
 \underline{110} \\
 00110 \\
 \underline{110} \\
 0
 \end{array}$$

$$1 < 110$$

$$10 < 110$$

$$101 < 110$$

$$1010 > 110$$

$$1001 > 110$$

$$110 = 110$$

Ans

111

Octal Addition

$$\begin{array}{r}
 567 \\
 + 243 \\
 \hline
 1032 \quad \text{Ans}
 \end{array}$$

Octal Subtraction

$$\begin{array}{r}
 624 \\
 - 265 \\
 \hline
 337
 \end{array}$$

Octal Multiplication

$$\begin{array}{r}
 72 \\
 \times 12 \\
 \hline
 164 \\
 72 \times \\
 \hline
 1104 \quad \text{Ans}
 \end{array}$$

(12)

$$\begin{array}{r}
 357 \\
 \times 23 \\
 \hline
 1314 \\
 736 \times \\
 \hline
 10675 \quad \text{Ans}
 \end{array}$$

Hexadecimal Addition :-

$$\begin{array}{r}
 5689 \\
 + 4574 \\
 \hline
 9BF D
 \end{array}$$

$$\begin{array}{r}
 A D D \\
 + D A D \\
 \hline
 188A
 \end{array}$$

Hexadecimal Subtraction :-

$$\begin{array}{r}
 9654 \\
 - 5321 \\
 \hline
 4333
 \end{array}$$

(12)

$$\begin{array}{r}
 974B \\
 - 587C \\
 \hline
 3ECF
 \end{array}$$

Hexadecimal Multiplication

$$\begin{array}{r} \times 23 \\ A B C \end{array}$$

$$2034$$

$$1578X$$

$$177B4$$

R's complement

$$r^n - N$$

① (101) find r's complement

$$r=2$$

$$n=4$$

$$N=1101$$

$$= 2^4 - 1101$$

$$= (16)_{10} - 1101$$

$$= 10000 - 1101$$

$$= \underline{\underline{0001}}$$

② (r-1)'s complement

$$(r-1)'s \text{ complement} \Rightarrow r^n - N = 1$$

$$= (r's \text{ complement}) - 1$$

8's complement of 5674

7's complement of 5674 = 7777

5674

2103

+1

2404

8's complement

Short cut of 2's complement

(101100)₂
↓ 2's comp
(010100)₂

- ① write down the given no.
- ② starting from LSB, copy all the zeroes till the first 1.
- ③ Copy the first 1.
- ④ complement all the remaining bits.

Data Representation

signed → 110
unsigned → not represent

signed
unsigned

+ve 6 = 0110

-ve 6 = 1110

+ve 13 → 01101

-ve 13 → 11101

1's complement subtraction

① (1100)₂ - (0101)₂

1100
0000

10110
+1

1111 Ans

② (0101)₂ - (1100)₂

0101 0101
+0001 0011

0110 0000

Ans (-0111) Ans

* Binary Sub. using 2's complement

$$\begin{array}{r} \textcircled{1} \quad 1001 \\ + 1100 \\ \hline 10101 \end{array}$$

$$= (0101)_2 \text{ Ans}$$

$$\begin{array}{r} \textcircled{2} \quad 0110 \\ 0101 \\ \hline 1011 \end{array}$$

$$\text{Ans } (-0100)_2 \text{ Ans}$$

* Classification of codes

① Weighted codes

ex. Binary,
8421, 2421 etc

② Non-weighted codes

ex. excess-3
0

③ Reflective code
(Self comp. code)

ex 2421, excess-3

④ Sequential code

ex. 8421,
excess-3 etc

⑤ Alphanumeric codes

ex. ASCII

American

Standard code for

Information Interchange

⑥ Error detecting & correcting codes

ex. Hamming code

Binary
10 $\rightarrow$ 1010

BCD

0001-0000

* BCD is less efficient than Binary b/c no. of bits is more and hence it takes more time to perform calculations.

Binary coded Decimal

In this code each decimal digit is represented by 4-bit binary number.

$2^4 \rightarrow 16$ Out of 16 only 10 are valid.

0 0000

1 0001

2 0010

3 0011

4 0100

5 0101

6 0110

7 0111

8 1000

9 1001

10 Invalid

11 Invalid

12 Invalid

13 Invalid

14 Invalid

15 Invalid

BCD to Decimal

10100

as BCD is four digit bit binary number.
So, make pair of bits

 $(0001, 0100)_2$ $(14)_{10}$ BCD AdditionCase Iex1. $(2)_{10} + (6)_{10}$ BCD Addition:

0010

+ 0110

 $(1000)_2 = (8)_{10}$

condition is :- $\text{sum} \leq 9$, final carry = 0.
then our answer will be correct.

Necessary Conditions

- ① $\text{sum} \leq 9$, final carry = 0.
- ② $\text{sum} \leq 9$, final carry = 1.
- ③ $\text{sum} \geq 9$, final carry = 0

if $\text{sum} \geq 9$ then we have to add 6 to the sum to make it a valid BCD number.

↓

B/c

0

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

we add 6

15

15 + 6 = 21

21

21

21

21

21

21

∴ $15 - 9 = 6$
To make it valid add 6 to the number.

Case III:- $(3)_{10} + (7)_{10}$

0011

+ 0111

$(1010)_2 = (10)_{10}$ which is incorrect in BCD

so, we add 6 to the number. (sum > 9)

1010

+ 0110

$(10000)_2$ in BCD $(00010000)_2 = (10)_{10}$

New incorrect answer.

Case II:

$(8)_{10} + (9)_{10}$

1000

+ 1001

10001

carry = 1 sum < 9. incorrect

So, we add +6.

10001

+ 0110

$(10111)_2 = (17)_{10}$ which is correct in BCD.

Imp Ex:

$(57)_{10} + (26)_{10} = (83)_{10}$

01010111

+ 00100110

01111101

sum < 9 sum > 9

answer is incorrect

So, we add +6.

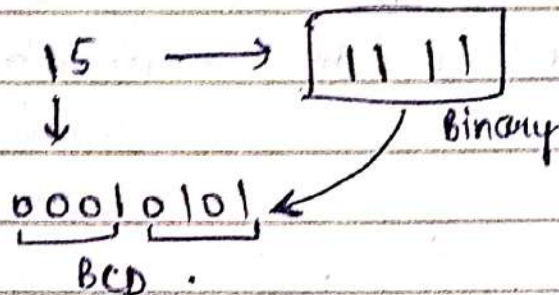
01111101

+ 0110

$(10000011)_2 = (83)_{10}$ then our answer is correct

* Shift Add - 3 Method.

This method is use to convert Binary to BCD directly. (No need to convert it in decimal)



for ex,

operations	Tens	ones	Decimal	
original no.			1111 (Binary)	
Shift		1	111	} shift till one no. is greater than 4.
Shift		11	11	
Shift		111	1	
Add-3		+ 011		
Shift	1	0101	1	
	1	5	10101 (BCD)	

2421 code:-

Like BCD if each bit decimal is encoded in 4-bit number (2421 code)

2421 code has a self complementing code (also called reflective code).

So, the code is given by

	2	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	0	1
8	1	1	1	0
9	1	1	1	1

8421 code is a BCD code

if w_3, w_2, w_1, w_0 are there like 2421, 8421, ...

and if $w_3 + w_2 + w_1 + w_0 = 9$ then it is self complementing

ex. $2-4-2-1 \Rightarrow 2+4+2+1=9$

Hence, self complement code

! $8-4-2-1=15 \neq 9$ then

it is not self complementing code.

Some other code are given by:-

Decimal	7421	5421	3321	84-2-1	74-2-1
0	0000	0000	0000	0000	0000
1	0001	0001	0001	0111	0111
2	0010	0010	0010	0110	0110
3	0011	0011	0011	0101	0101
4	0100	0100	0101	0100	0100
5	0101	0101	1010	1011	1010
6	0110	0110	1100	1010	1001
7	1000	0111	1001	1001	1000
8	1001	1011	1110	1000	1111
9	1010	1100	1111	1111	1110

Excess-3 code (self completing code) ~~ex~~ (It's 4 bit code)

→ excess-3 code is only unweighted code which is self complementing.
 b/c we don't have positional weight

Decimal	BCD	XS-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	10110
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

ex.

24
↓ ↓

0010	0100
+ 0110	+ 0110
0101	0110
5	7

Excess-3 coded AdditionEx-1

$$(2)_{10} + (5)_{10} \rightarrow (7)_{10}$$

0010

0101

+ 0011

+ 0011

0101

1000

0101

← excess of 3

+ 1000

← excess of 3

1101

← excess of 6

↓

This answer is wrong

So, we have to sub 3.

1101

- 0011

1010

Ans. $(10)_{10}$ Ex-2

$$(27)_{10} + (39)_{10}$$

 $(27)_{10}$

00100111

~~00100111~~ $(39)_{10}$

00111001

~~00111001~~

01101000

★ If final carry = 1 we will add 0011

If final carry = 0 we will sub 0011

~~0011~~

BCD

Excess-3

27

 $(27)_{10}$

00100111

→

01011010

39

 $(39)_{10}$

00111001

→

01101100

66

33

99

01011010

+ 00011100

11100110

01011010

+ 01011100

10110110

01011010

+ 01101100

11000110

- 00110011

10011001

99

Introduction to Gray code (Reflected Binary code / Unit distance code / Minimum error code / Cyclic code)

- Unweighted code.
- Two successive values differ in only 1 bit
- Binary code is converted to Gray code to reduce switching operation.
- Gray code are use to facilitates error correction in digital comm. such as cable T.V system.

Decimal	Binary	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000
16	10000	10000

Binary To Gray code:-

1011 $\longrightarrow$

↓

1110

Gray code

Gray code To Binary

1110

↓

1011

What is Parity?

It detect error (but only single bit).

→ Parity bit is an extra bit that is added with the message.

Parity is of two types

① Even Parity

0100

01001

② Odd Parity

0100

01000

Now we find our data in even parity and due to noise, one 0 gets converted into 1 then the our msg. gets converted into odd parity. So, we know that there is an error in our msg.

How To detect error :-

→ what is Hamming code and how to detect it?

Hamming code = msg bit + Parity bit.

$$2^K - 1 \geq m + K$$

where K = No. of parity bit

m = No. of bit in our msg

Ex.

Encode data bit 0101 having even parity hamming code.

Hamming code = msg + Parity bit

$$= 4 + 3$$

$$= 7$$

$$2^3 - 1 \geq 4 + 3$$

$$7 \geq 7$$

∴ our parity bit = 3

Now, To know the value of parity bit.

1 2 3 4 5 6 7

P_1 P_2 0 P_4 1 0 1

$$P_1 = 0 \oplus 1 \oplus 1 = 0$$

$$P_2 = 0 \oplus 0 \oplus 1 = 1$$

$$P_4 = 1 \oplus 0 \oplus 1 = 0$$

∴ our Hamming code = 0101001

→ If not given assuming even parity

ex. $P_4 = 1 = 1$. [in order to make it even]

[as there is no one for

[and if it is odd parity to ex-or operation, then we will write 0].

check.

* In order to, the error:- [whether it is present or not]

ex. Hamming code. Transmitted 0100101 → Received 1100101.

$C_1 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$

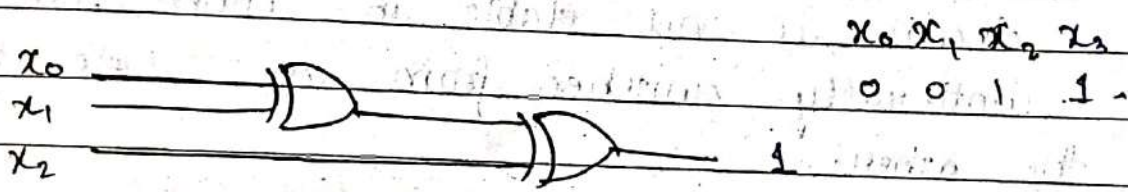
$C_2 = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 0$

$C_4 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0$

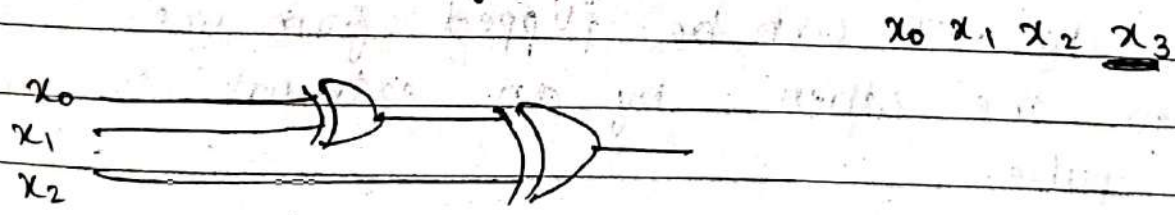
$C_4 \ C_2 \ C_1$
0 0 1

So, in decimal error is present at 1st position.

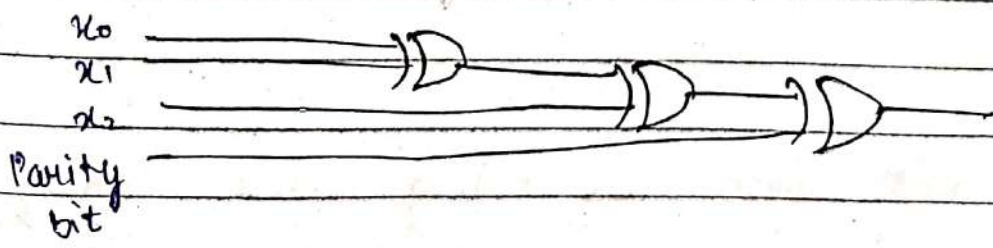
* 3-bit even Parity Generator.



* 3-bit odd Parity Generator.



* 3-bit even Parity Checker.



If comes
0 → No error
1 → Error

Same for odd parity

Logic Gates :-

It is a physical device which performs logic operation on one or more logical inputs and produces a single logical output.

Logic operation :- Inversion, Logic Multiplication, Logic sum etc

→ Basic Gates :- NOT, AND and OR.

→ Universal Gates :- NAND and NOR.

→ Arithmetic / Derived Gates :- X-OR / X-NOR Gate.

* Astable Multivibrator :- Astable, in which the circuit is not stable in either state, it continually switches from one state to the other.

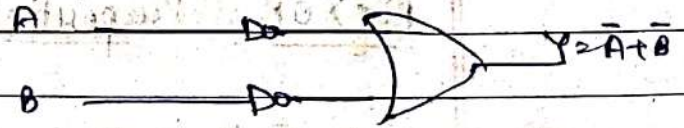
* Bistable Multivibrator :- Bistable, in which the circuit is stable in either of the two states. It can be flipped from one state to the other by an external trigger pulse.

* Formula for Time Period :-

$$T = 2 \times \text{no. of Not Gates} \times t_{pd}$$

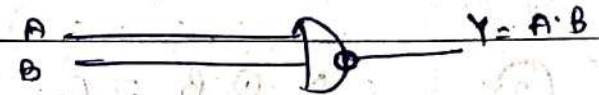
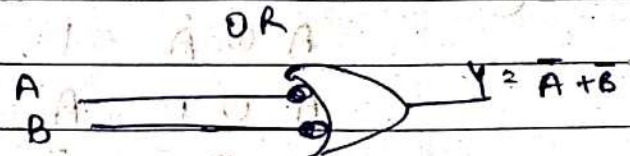
NAND Gate (Bubbled OR)

$$(\overline{A \cdot B}) \text{ OR } (\overline{A} + \overline{B})$$



Truth Table

	A	B	Y
disable OR=1	0	0	1
	0	1	1
enable inverter	1	0	1
	1	1	0



Associative law ✗

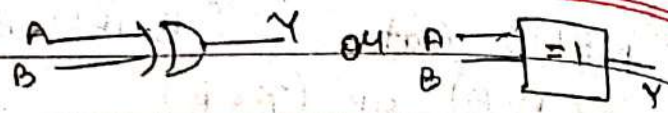
Commutative law ✓

Used I/P

same as AND Gate

E

XOR Properties



IEEE symbol

①

$$A \oplus A = 0$$

$$A \oplus \bar{A} = 1$$

$$A \oplus 0 = A$$

$$A \oplus 1 = \bar{A}$$

②

$$A \oplus B = C, B \oplus C = A, A \oplus C = B$$

$$\therefore A \oplus B \oplus C = 0$$

③

$$A \oplus A = 0$$

$$A \oplus A \oplus A$$

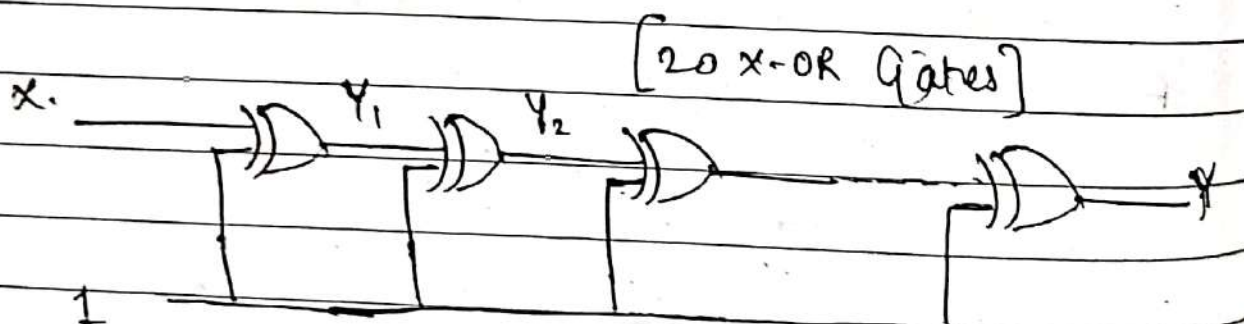
$$\underbrace{0 \oplus A}_{0 \oplus A} = A$$

④

$$A \oplus A \oplus A \dots n = A, n = \text{odd}$$

$$= 0, n = \text{even}$$

Problem



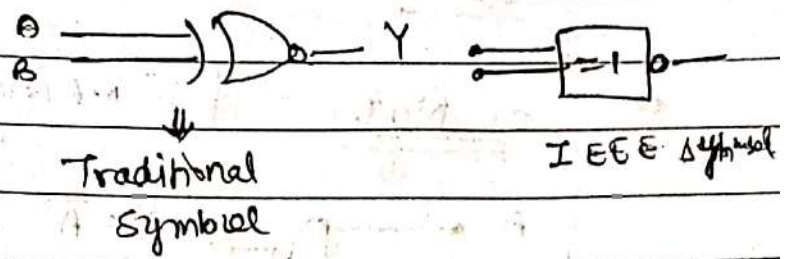
$$Y_1 = X \oplus 1$$
$$= \bar{X}$$

$$Y_2 = \bar{X} \oplus 1$$
$$= \bar{\bar{X}}$$
$$= X$$

So, after 20

$$Y = \underline{\underline{X}}$$

EX-NOR Properties



$$① A \odot A = 1$$

$$② A \odot \bar{A} = 0$$

$$③ A \odot 0 = \bar{A}$$

$$④ A \odot 1 = A$$

$$⑤ A \odot A \odot A = A$$

$$⑥ A \odot A \odot A \odot A \dots n = A, n = \text{odd}$$

$$= 1, n = \text{even}$$

⑦ XOR and X-NOR are complement of each other
 $\rightarrow$ if even I/P

EX-OR and EX-NOR are same if odd I/P are there

$$\text{ex } A \oplus B \oplus C = A \odot B \odot C \quad \text{--- ①}$$

$$A \oplus B \oplus C \oplus D = \overline{A \odot B \odot C \odot D} \quad \text{--- ②}$$

Proof:

$$A \odot B \odot C = A \odot B = Y$$

$$= Y \odot C$$

$$= YC + \bar{Y}\bar{C}$$

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

from eqn ②

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

$$\text{Let } A \odot B = Y = \bar{Y} \cdot C + Y \cdot \bar{C}$$

$$= Y \oplus C$$

$$= A \oplus B \oplus C$$

NAND

*

as NOT.

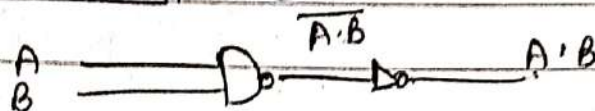
[1 NAND GATE]



*

as AND :-

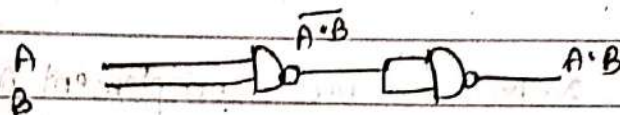
[2 NAND GATE]



$\Downarrow$

*

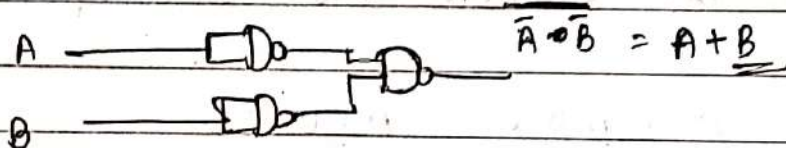
Now only using NAND only



*

as OR.

NAND: $Y = \overline{A \cdot B} = \bar{A} + \bar{B}$ [3 NAND GATE]

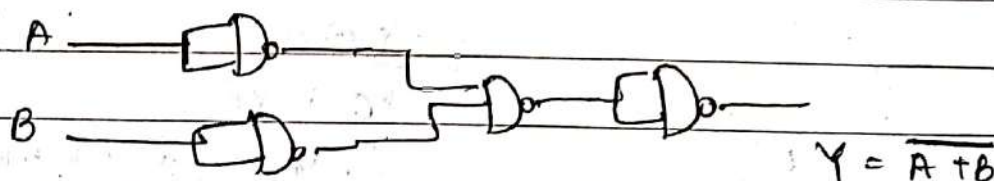


*

as NOR.

$(\bar{A} + \bar{B})$

[4 NAND GATE]

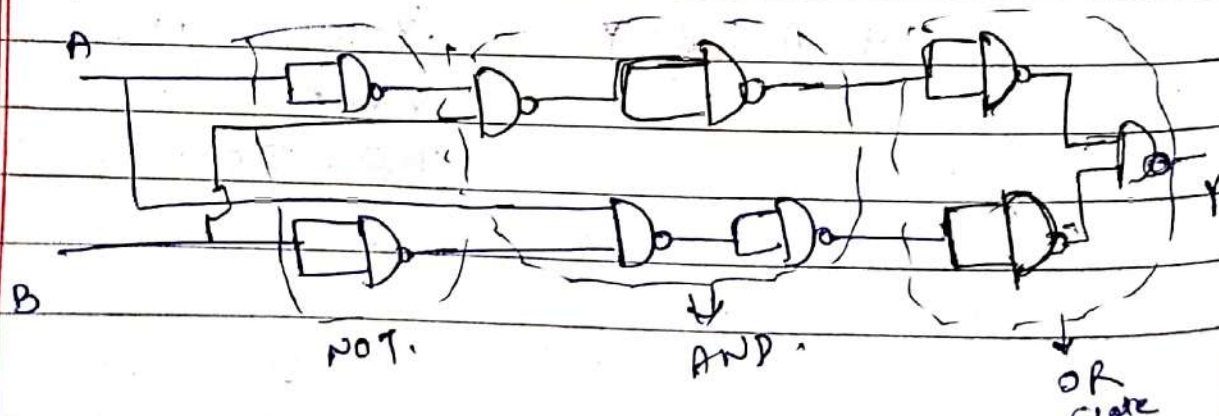


*

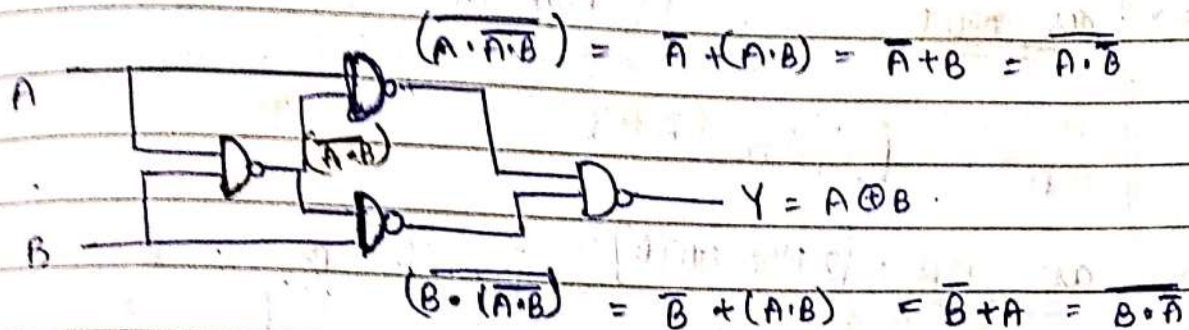
as EXOR

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

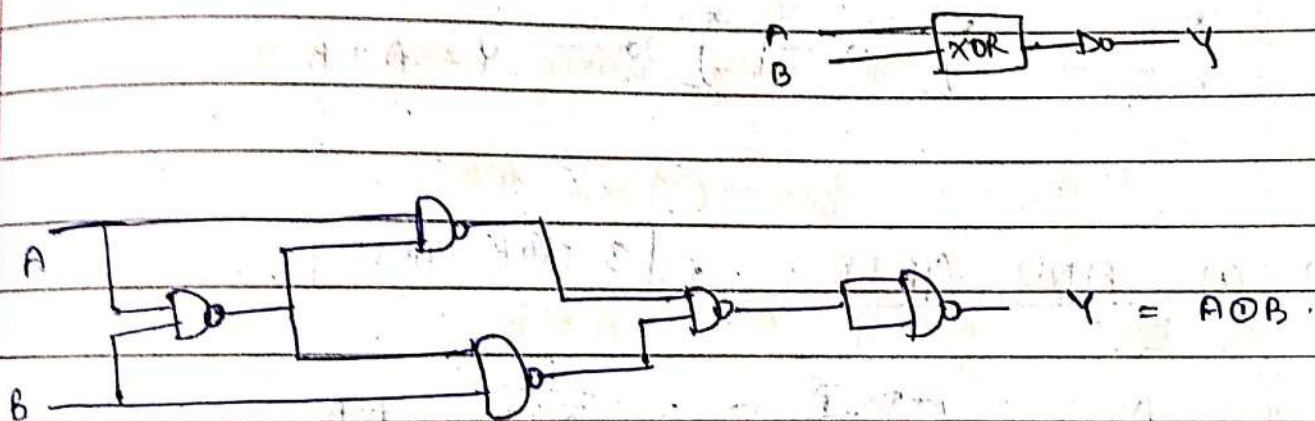
[9 NAND GATE]



This can also be prepared by 4 NAND GATE..

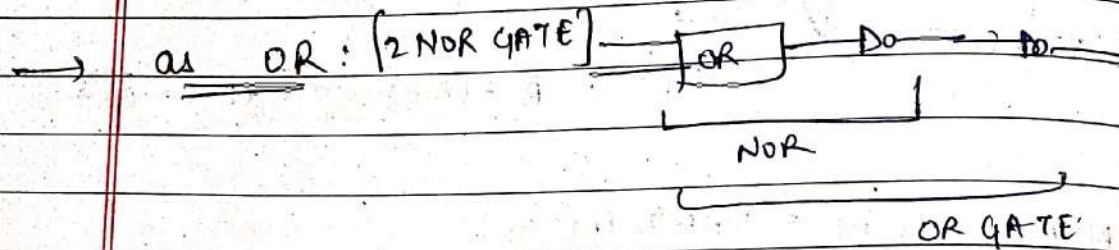
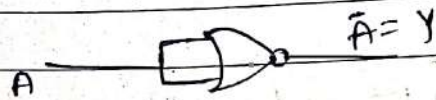


an EXNDR (by 5 NAND gates)

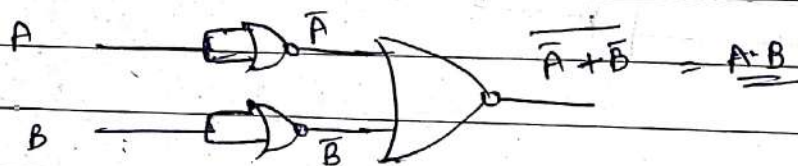


NOR GATE

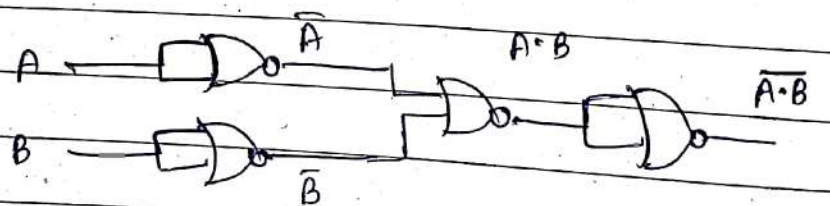
→ as NOT :- [1 NOR GATE]



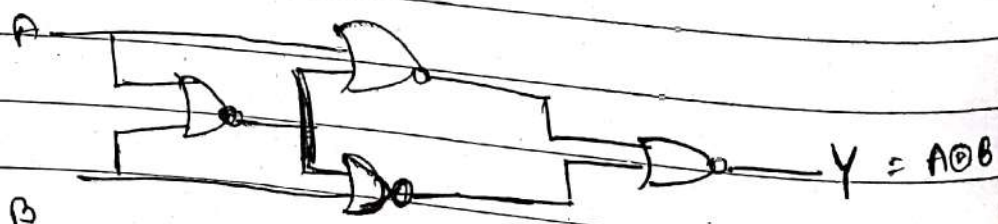
→ as AND GATE [3 NOR GATE]



→ as NAND Gate [4 NOR GATE]



→ as EXNOR Gate :-



check board configuration

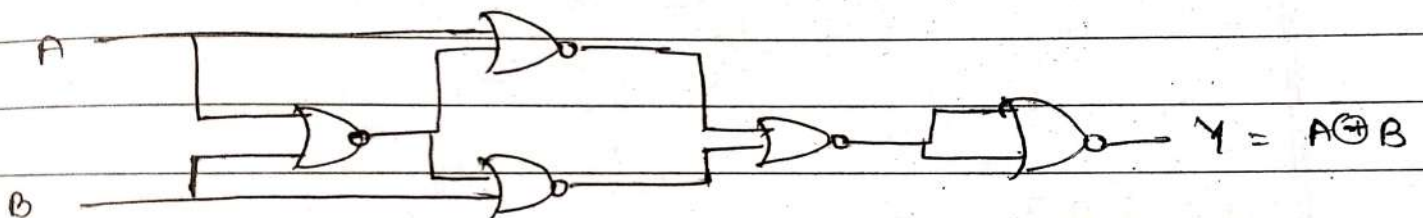
$b_0 \oplus b_1 \oplus b_2 \oplus b_3$

Page No.

your writing partner

Date / /

→ as X-OR Gate :-

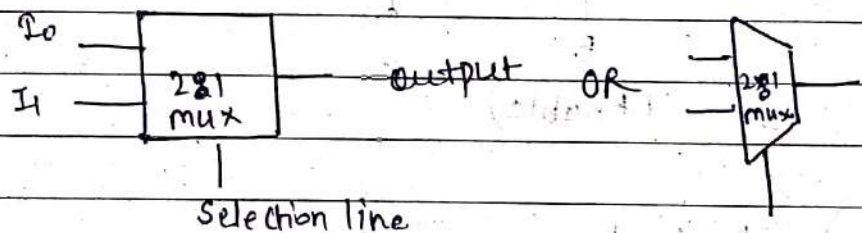


Multiplexer: (Mux)

- It is combinational circuit that selects binary information from many input lines and directs it to single output lines.
- It is also called data selector because it is selecting our data.
- Selection line:- These are the input lines to select one ^{binary} information from many of the input lines.

Now, for m selection lines there will be n number of inputs are there in Multiplexer.
ie $m = \log_2 n$. $n = 2^m$

Representation:-



Application:-

Multiplexer are used to increase the amount of data that can be sent over the network within the amount of time and bandwidth.

Advantages:-

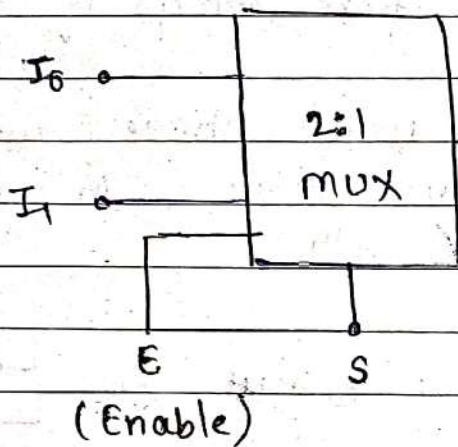
- Reduces ^{the usage} ~~loss~~ no. of wires
- (As wires and gates are reduced) Reduces circuit complexity and cost
- Implementation of various circuit using mux.

Types

Selection lines

- | | | |
|---|----------|---|
| ① | 2:1 mux | 1 |
| ② | 4:1 mux | 2 |
| ③ | 8:1 mux | 3 |
| ④ | 16:1 mux | 4 |
| ⑤ | 32:1 mux | 5 |

2:1 MUX:-

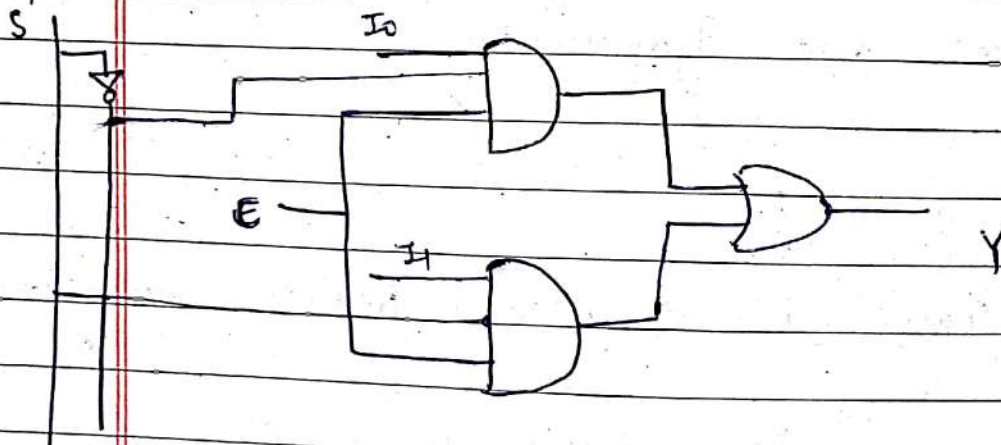


E	S	Y
0	X	0
1	0	I ₀
1	1	I ₁

logical expression :-

$$Y = E \cdot \bar{S} \cdot I_0 + E \cdot S \cdot I_1$$

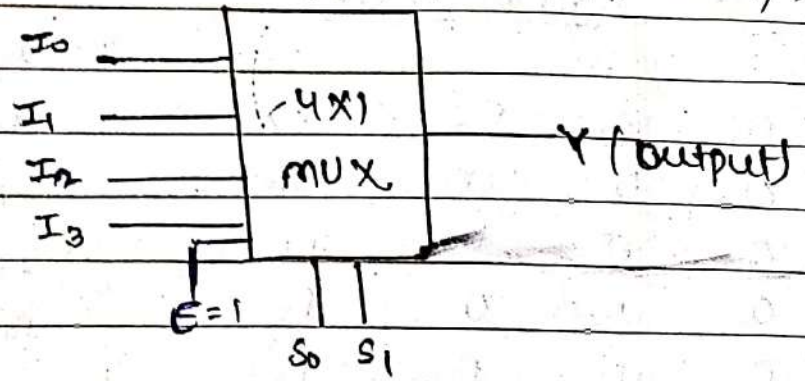
Implementation:-



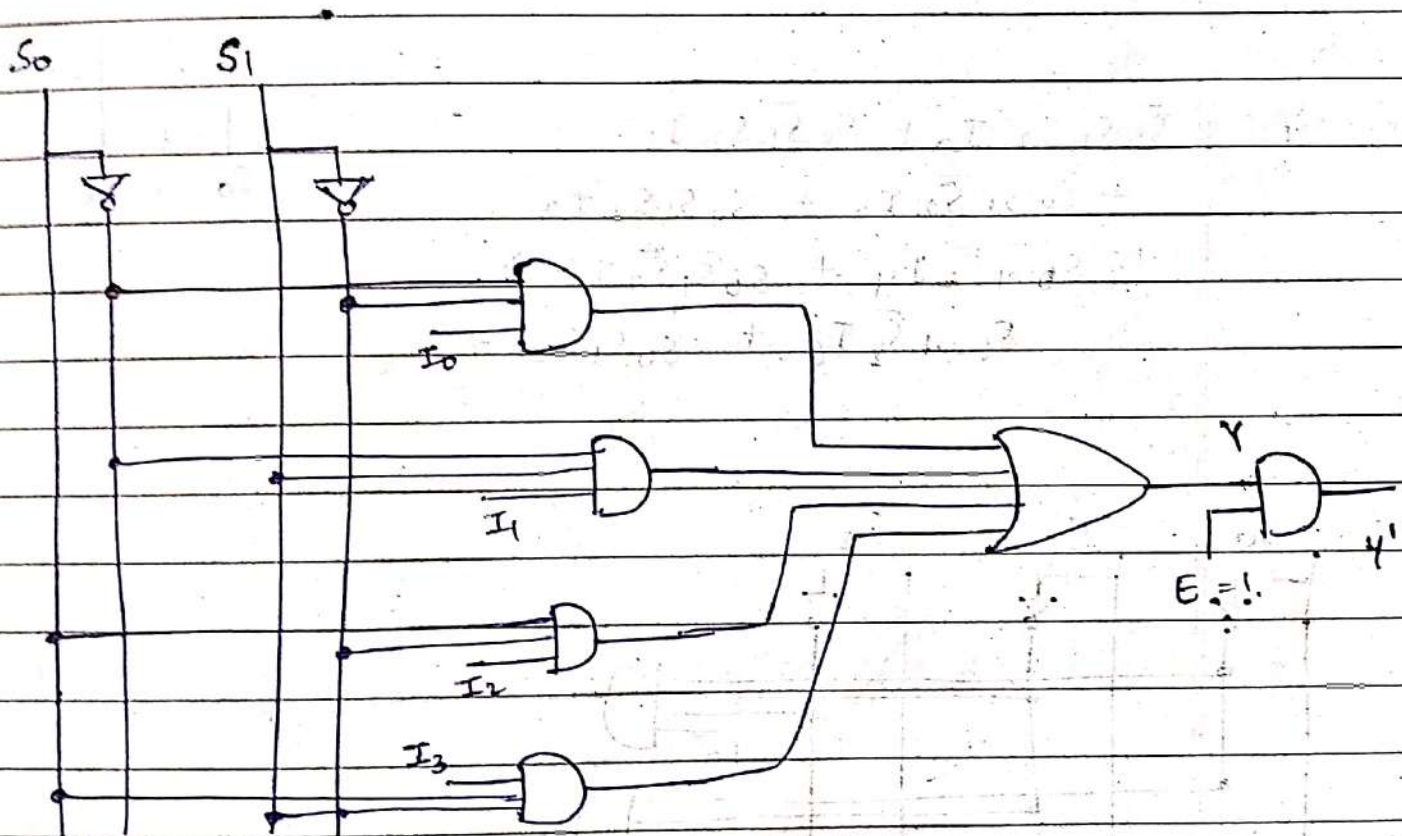
(We are not using Enable Not to make our circuit complicated).

4:1 MUX

S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

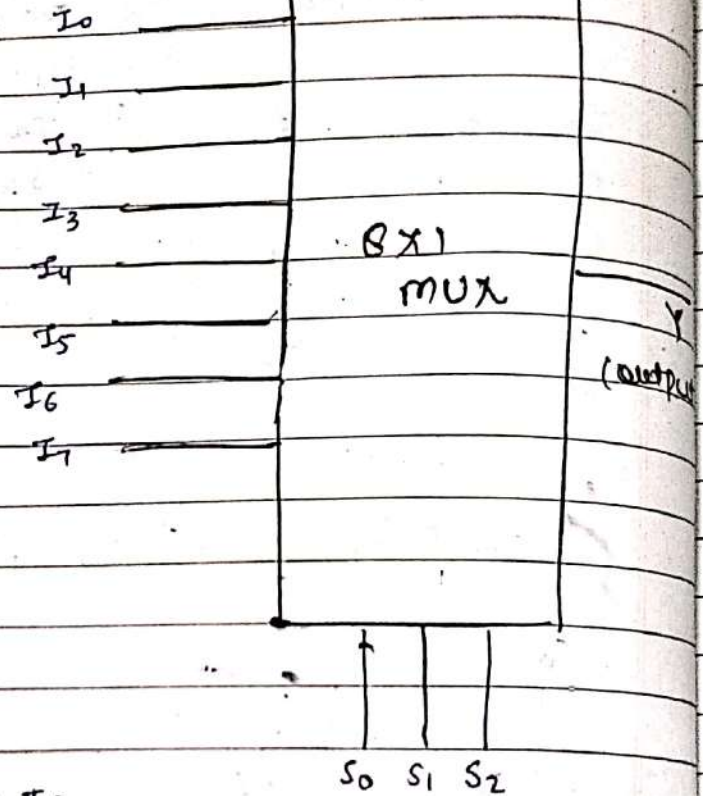


$$Y = \bar{S}_0 \bar{S}_1 I_0 + \bar{S}_0 S_1 I_1 + S_0 \bar{S}_1 I_2 + S_0 S_1 I_3$$

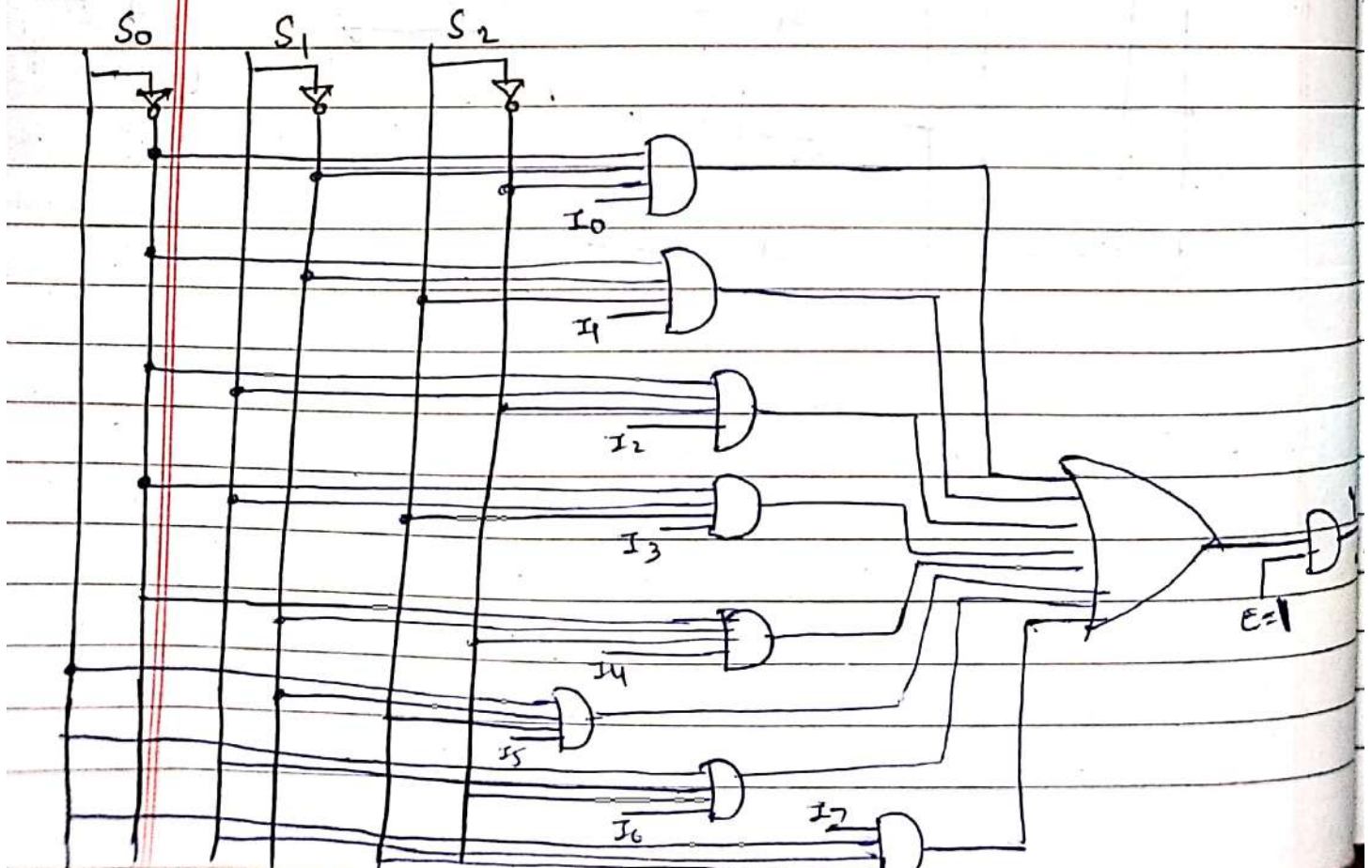


8x1 MUX

S_0	S_1	S_2	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



$$\begin{aligned}
 Y = & \bar{S}_0 \bar{S}_1 \bar{S}_2 I_0 + \bar{S}_0 \bar{S}_1 S_2 I_1 \\
 & + \bar{S}_0 S_1 \bar{S}_2 I_2 + \bar{S}_0 S_1 S_2 I_3 \\
 & + S_0 \bar{S}_1 \bar{S}_2 I_4 + S_0 \bar{S}_1 S_2 I_5 + \\
 & S_0 S_1 \bar{S}_2 I_6 + S_0 S_1 S_2 I_7
 \end{aligned}$$



MUX TREE BASIC (obtaining higher order MUX using lower order MUX)

Implement 4x1 MUX using 2x1 MUX.

Roughly

$$\# 4 = n$$

$$\frac{4}{2} = 2$$

$$+ \textcircled{3}$$

$$\frac{\textcircled{2}}{2} = 1$$

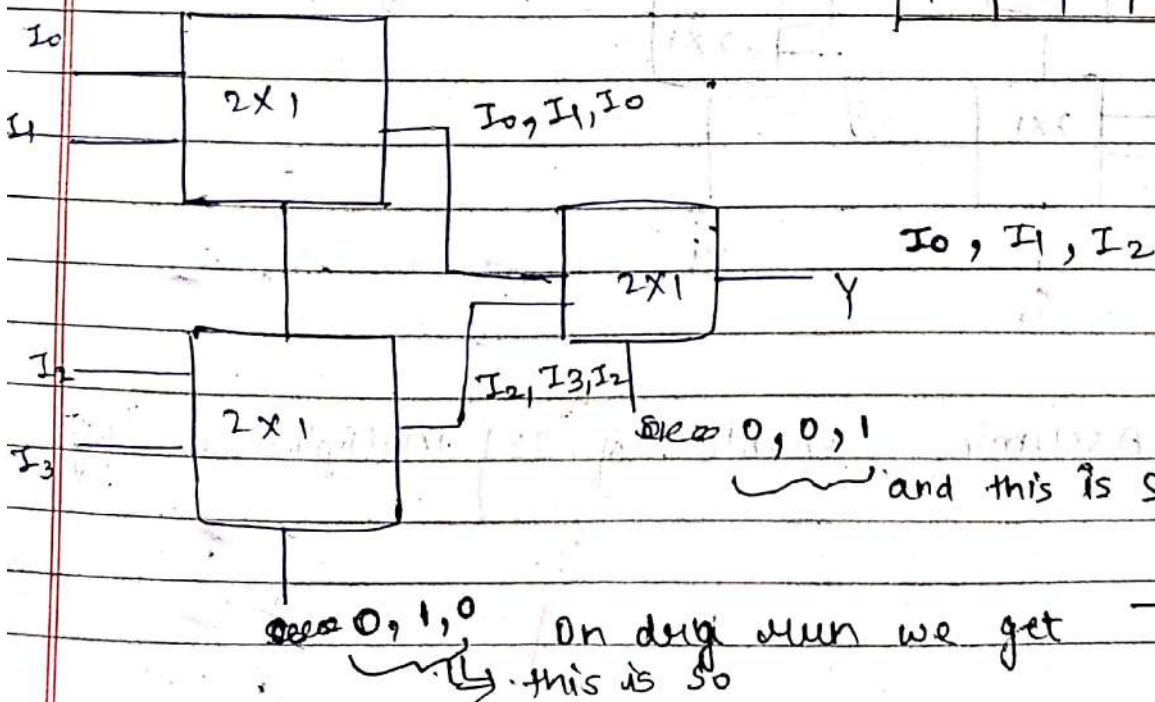
(untill we get 1).

So, 3 ~~2x1~~ $\rightarrow$ 2x1 MUX required to obtain 4x1 MUX

[in the same sequen

Justification.

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



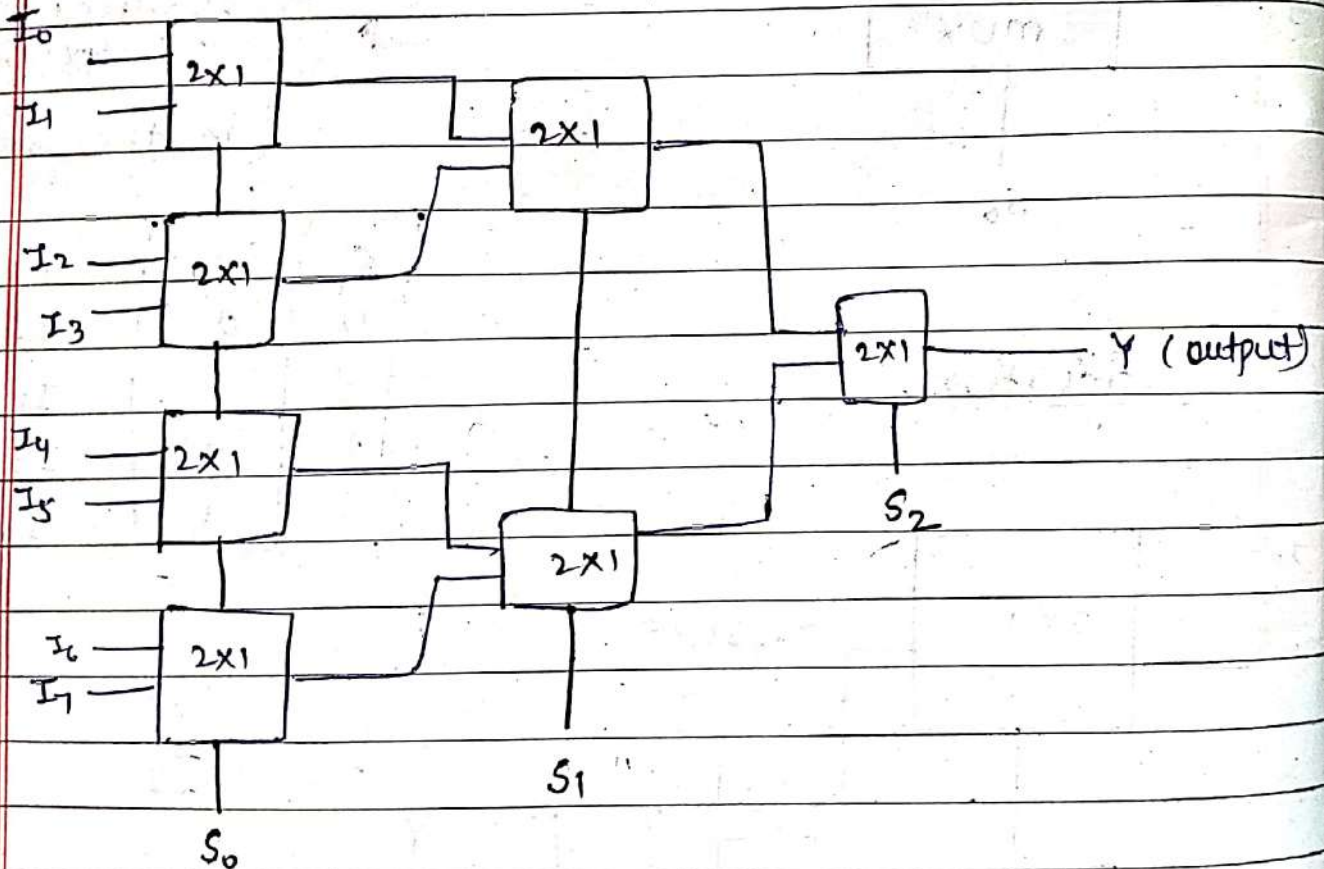
8x1 MUX Using 2x1 MUX

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Roughly

$$\begin{aligned}
 n_1 &= 8 & n_2 &= 2 \text{ available} \\
 \frac{8}{2} &= 4 & & \\
 &+ & & \\
 \frac{4}{2} &= 2 & & \\
 &+ & & \\
 \frac{2}{2} &= 1 & & \text{in same sequence}
 \end{aligned}$$

7 (2x1) MUX



Assuming enable of all 2x1 Multiplier are high.

8x1 MUX Using 4x1 MUX: (Special Case)

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

Roughly

$$\frac{8}{4} = 2$$

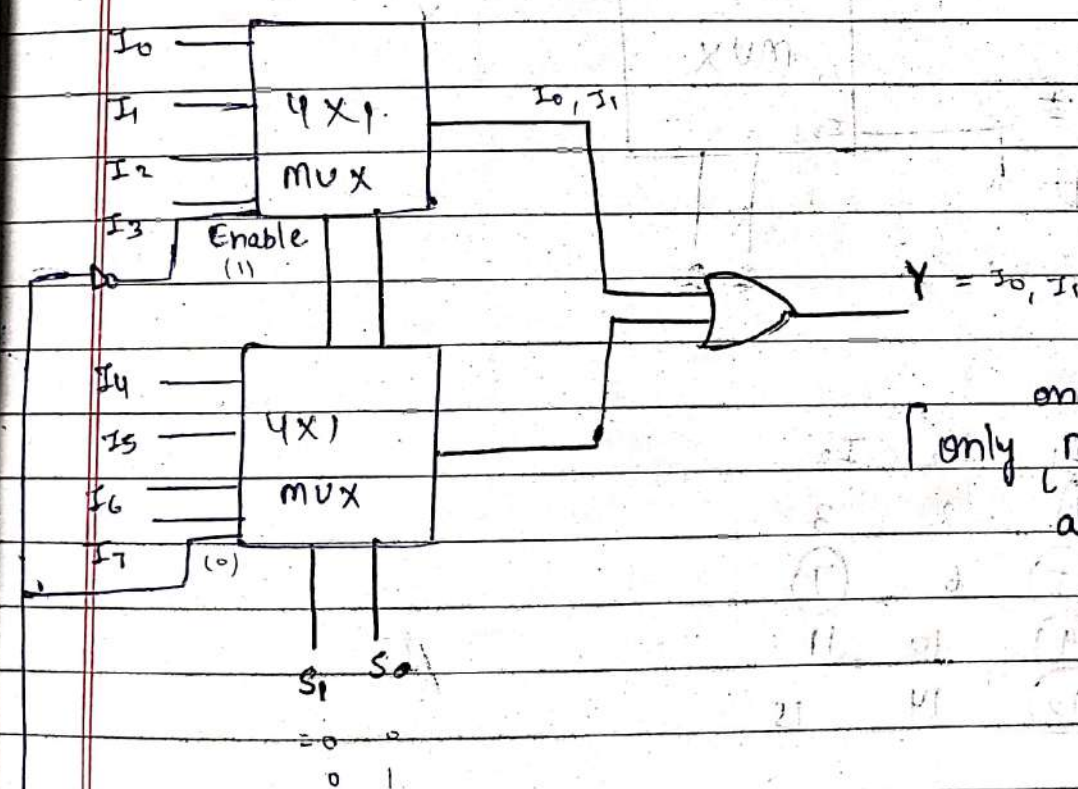
$$\frac{2}{4} = 0.5$$

$$\frac{2}{4} = 0.5$$

$$\frac{2}{4} = 0.5$$

So, we cannot use the previous method

So, we can implement this by using varying the enable



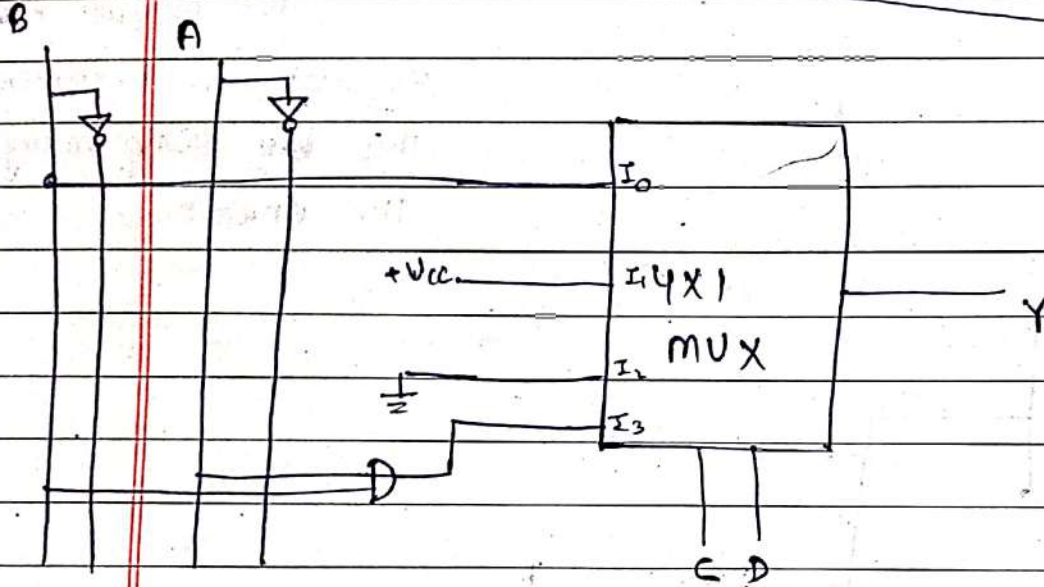
only one MUX is working at a time

$S_2 = 0$

Implementation of Boolean function Using Multiplexer :-

Ques $F(A, B, C, D) = \sum m(1, 4, 5, 7, 9, 12, 13)$ (using 4×1 mux)
[AB as input and CD as selection line]

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



	I_0	I_1	I_2	I_3
$\bar{A}\bar{B}$	0	(1)	2	3
$\bar{A}B$	(4)	(5)	6	(7)
$A\bar{B}$	8	(9)	10	11
AB	(12)	(13)	14	15

$$I_0 = \bar{A}\bar{B} + \bar{A}B + A\bar{B} + AB$$

$$I_2 = 0$$

$$I_0 = \bar{A}B + A\bar{B} + AB = B(A + \bar{A}) = B$$

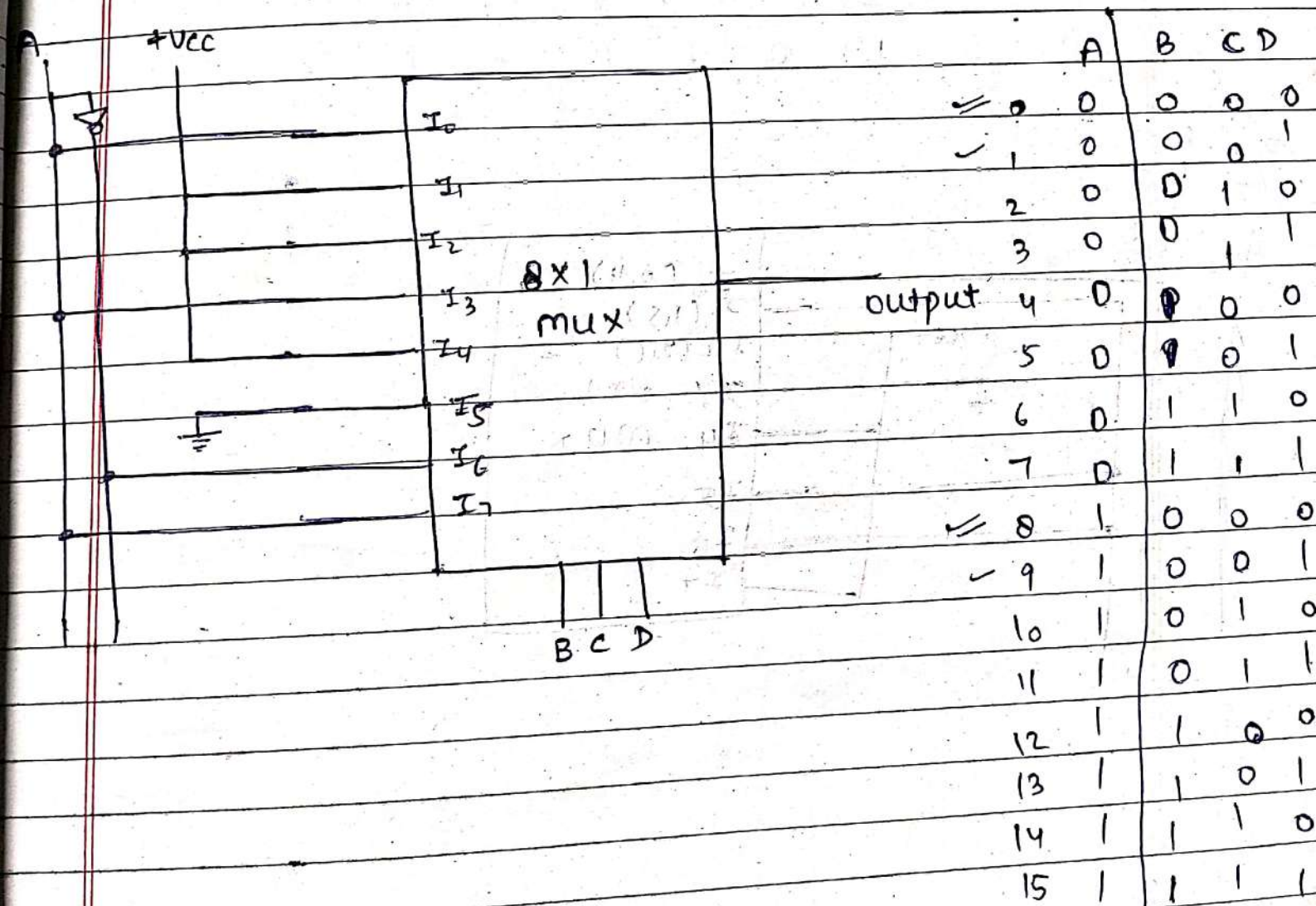
$$I_3 = AB$$

$$I_1 = \bar{A}\bar{B} + \bar{A}B + A\bar{B} + AB$$

$$= 1$$

Ques $F(A, B, C, D) = \sum(1, 2, 4, 6, 8, 9, 10, 11, 12, 15)$. using 8×1 mux
 [Taking A as input, B, C, D as selection line].

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



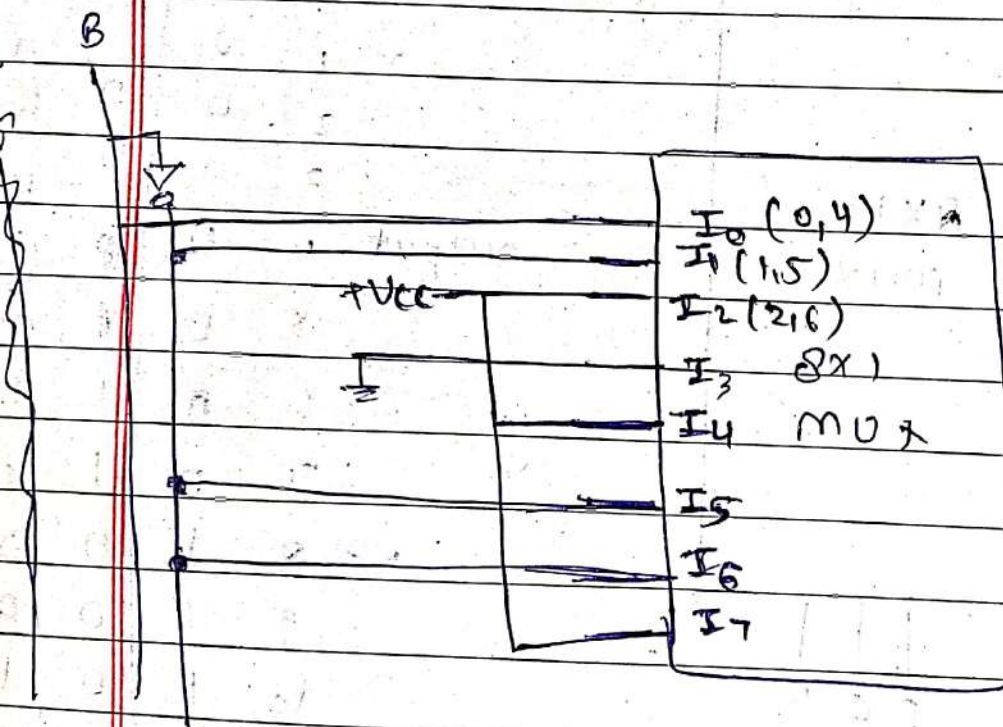
Ques $F(A, B, C, D) = \sum(1, 2, 4, 6, 8, 9, 10, 11, 12, 15)$ using 8x1 multiplexer
 [Taking B as input and ACD as a selection line]

$\overline{A}\overline{C}\overline{D}$ $\overline{A}\overline{C}D$ $\overline{A}C\overline{D}$ $\overline{A}CD$ $A\overline{C}\overline{D}$ $A\overline{C}D$ $AC\overline{D}$ ACD

I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7

$\overline{B}$ 0 ① ② 3 ⑧ ⑨ ⑩ ⑪
 B ④ 5 ⑥ 7 ⑫ 13 14 ⑮

B $\overline{B}$ 1 0 1 $\overline{B}$ $\overline{B}$ 1



1-Bit full Adder Using Multiplexer

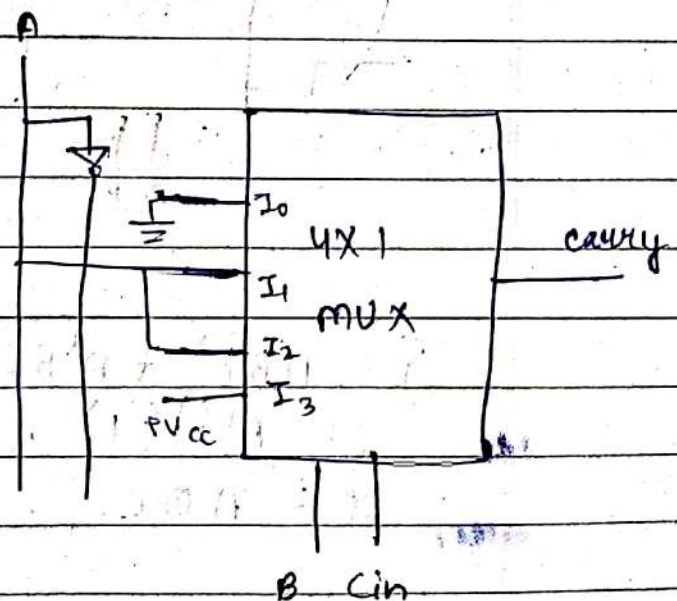
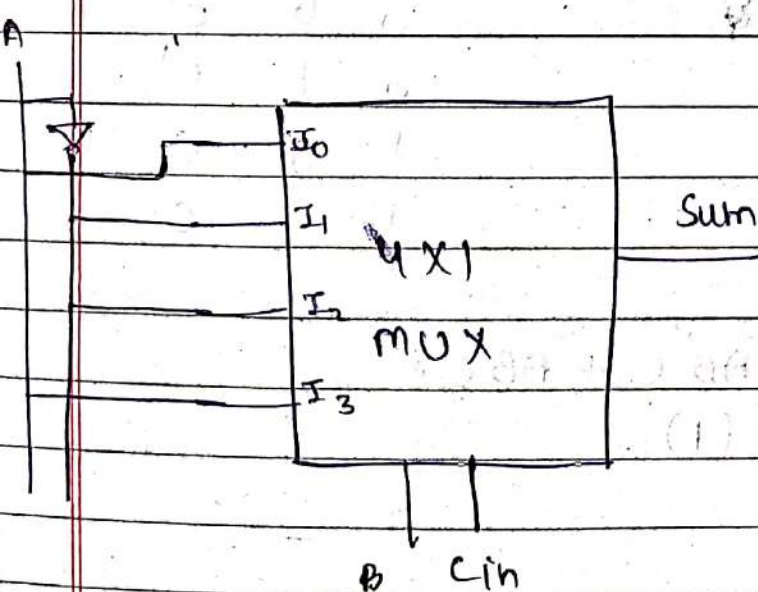
	A	B	C _{in}	S	C _o
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

$$S = \sum m(1, 2, 4, 7)$$

$$C_o = \sum m(3, 5, 6, 7)$$

	I ₀	I ₁	I ₂	I ₃
$\bar{A}$	0	①	②	3
A	④	5	6	⑦
	$\bar{A}$	$\bar{A}$	$\bar{A}$	$\bar{A}$

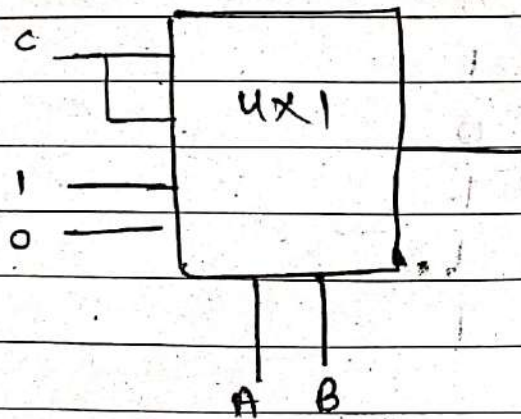
	I ₀	I ₁	I ₂	I ₃
A	0	1	2	③
A	4	⑤	⑥	⑦
	0	A	A	1



logical Expression from MUX.

let we have given: following MUX,

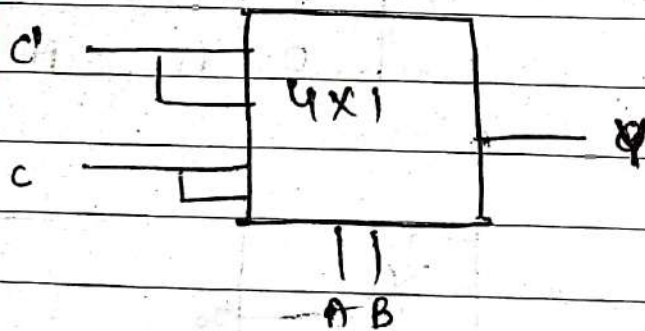
①



A	B	Y
0	0	C
0	1	C
1	0	1
1	1	0

$$\begin{aligned}
 Y &= A'B'C + A'B \cdot C + AB'(1) + AB \cdot (0) \\
 &= A'C(1) + AB' \\
 Y &= A'C + AB'
 \end{aligned}$$

②



A	B	Y
0	0	C'
0	1	C'
1	0	C
1	1	C

$$\begin{aligned}
 Y &= A'B'C' + A'BC' + AB'C + ABC \\
 &= A'C'(1) + AC(1) \\
 Y &= A \oplus C
 \end{aligned}$$

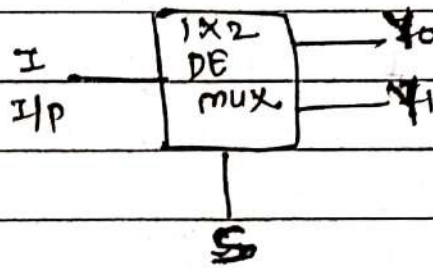
Demultiplexer (Demux) [Data Distributor]

A demultiplexer is a device that takes a single input line and routes it several output lines.

→ Reverse operation of multiplexer.

→ $n = 2^m$ $m \rightarrow$ selection line
 $n \rightarrow$ o/p lines.

→ Representation :-



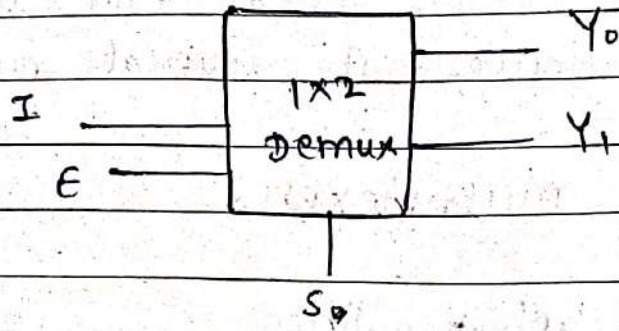
→ Application :-

→ Used in communication system to carry out the process of data transmission.

Advantages :

same as multiplexer. ~?

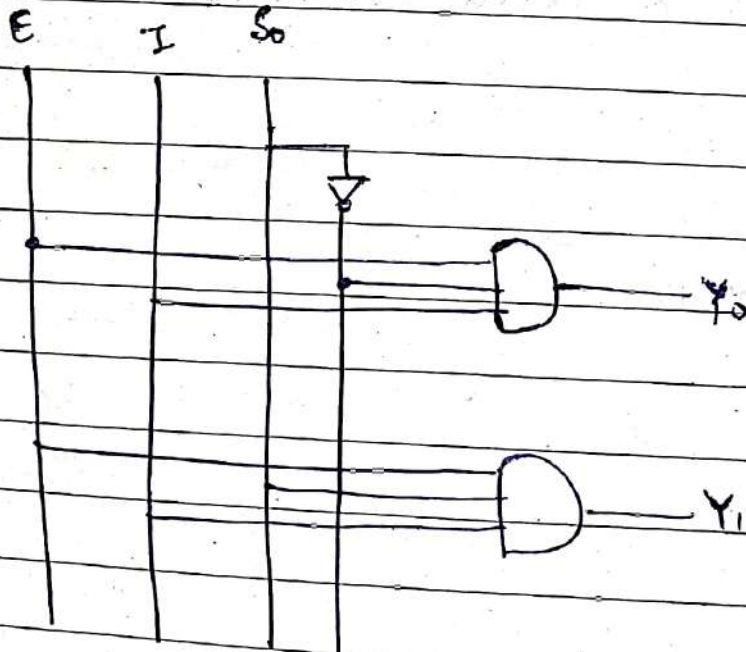
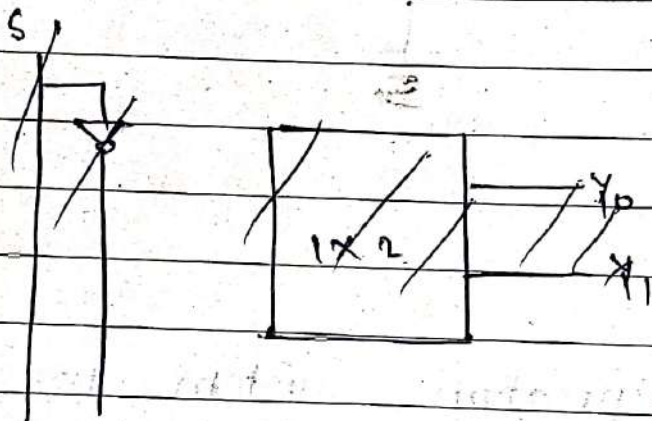
1x2 Demux



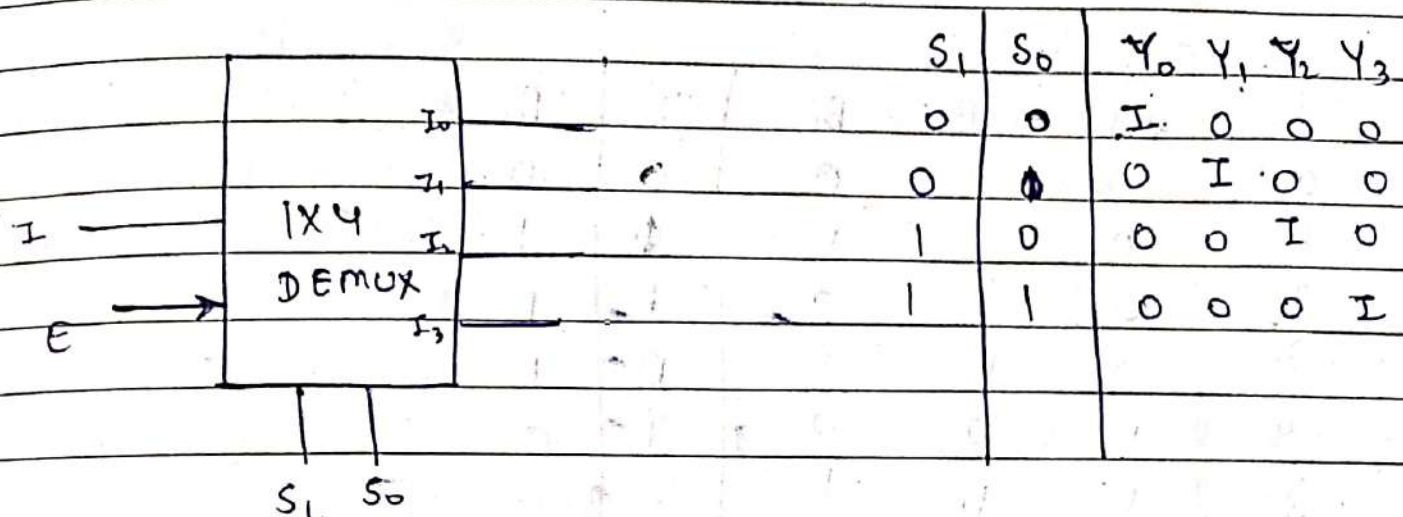
E	S	Y ₀	Y ₁
0	x	0	0
1	0	I	0
1	1	0	I

$$Y_0 = E \cdot \bar{S} \cdot I$$

$$Y_1 = E \cdot S \cdot I$$



1X4 Demultiplexer

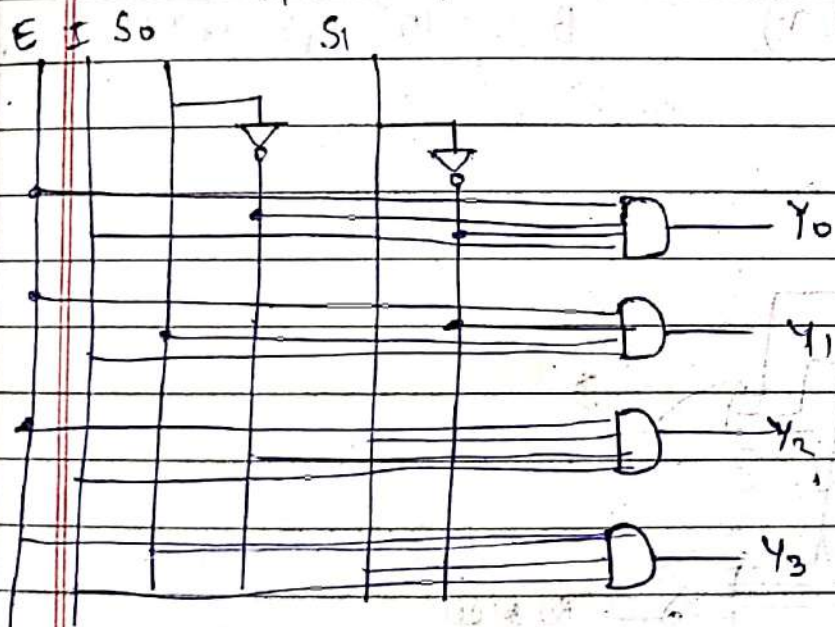


$$Y_0 = E \bar{S}_1 \bar{S}_0 I$$

$$Y_2 = E \cdot S_1 \bar{S}_0 I$$

$$Y_1 = E \bar{S}_1 S_0 I$$

$$Y_3 = E \cdot S_1 S_0 I$$

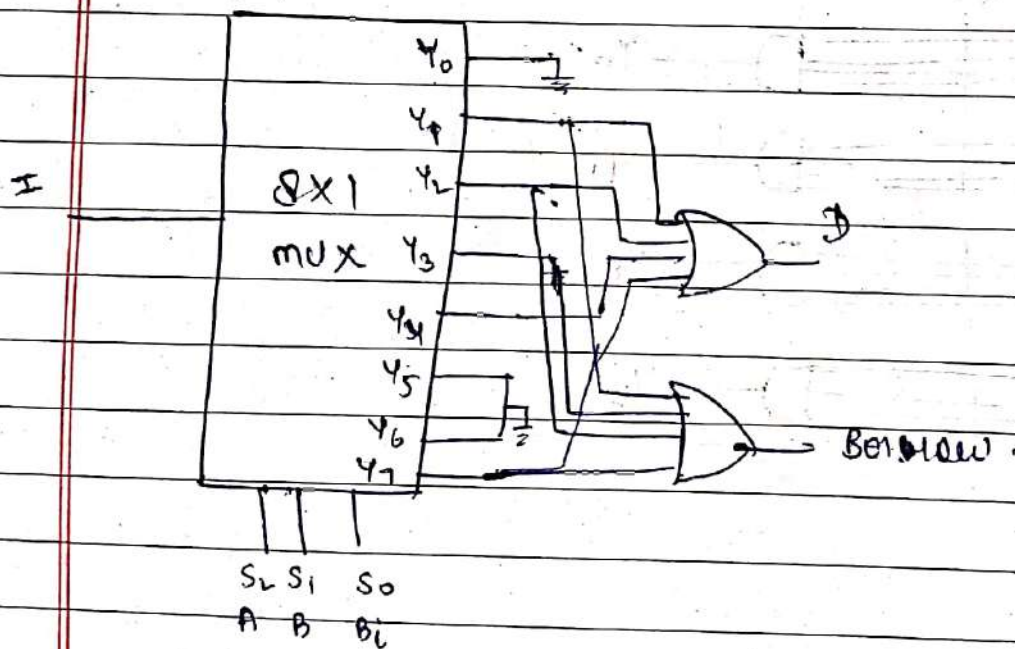


Full Subtractor Using 8x1 Demux

	A	B	B'	D	B ₀
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

$$D = \sum m(1, 2, 4, 7)$$

$$B = \sum m(1, 2, 3, 7)$$

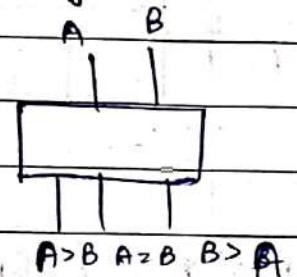


Magnitude Comparator OR 2-bit Comparator

A magnitude comparator is a combinational circuit that compares two numbers A and B and determine their relative magnitude i.e. (whether $A > B$, $A = B$, $A < B$).

~~2-bit~~ As there are

4 bit So Total $2^4 = 16$



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

for $B > A$

for $A \leq B$

	$\bar{B}_1\bar{B}_0$	$\bar{B}_1B_0$	$B_1\bar{B}_0$	B_1B_0
$\bar{A}_1\bar{A}_0$	0	1	3	2
$\bar{A}_1A_0$	4	5	7	6
$A_1\bar{A}_0$	12	13	15	14
A_1A_0	8	9	11	10

	$\bar{B}_1\bar{B}_0$	$\bar{B}_1B_0$	$B_1\bar{B}_0$	B_1B_0
$\bar{A}_1\bar{A}_0$	1			
$\bar{A}_1A_0$		1		
$A_1\bar{A}_0$			1	
A_1A_0				1

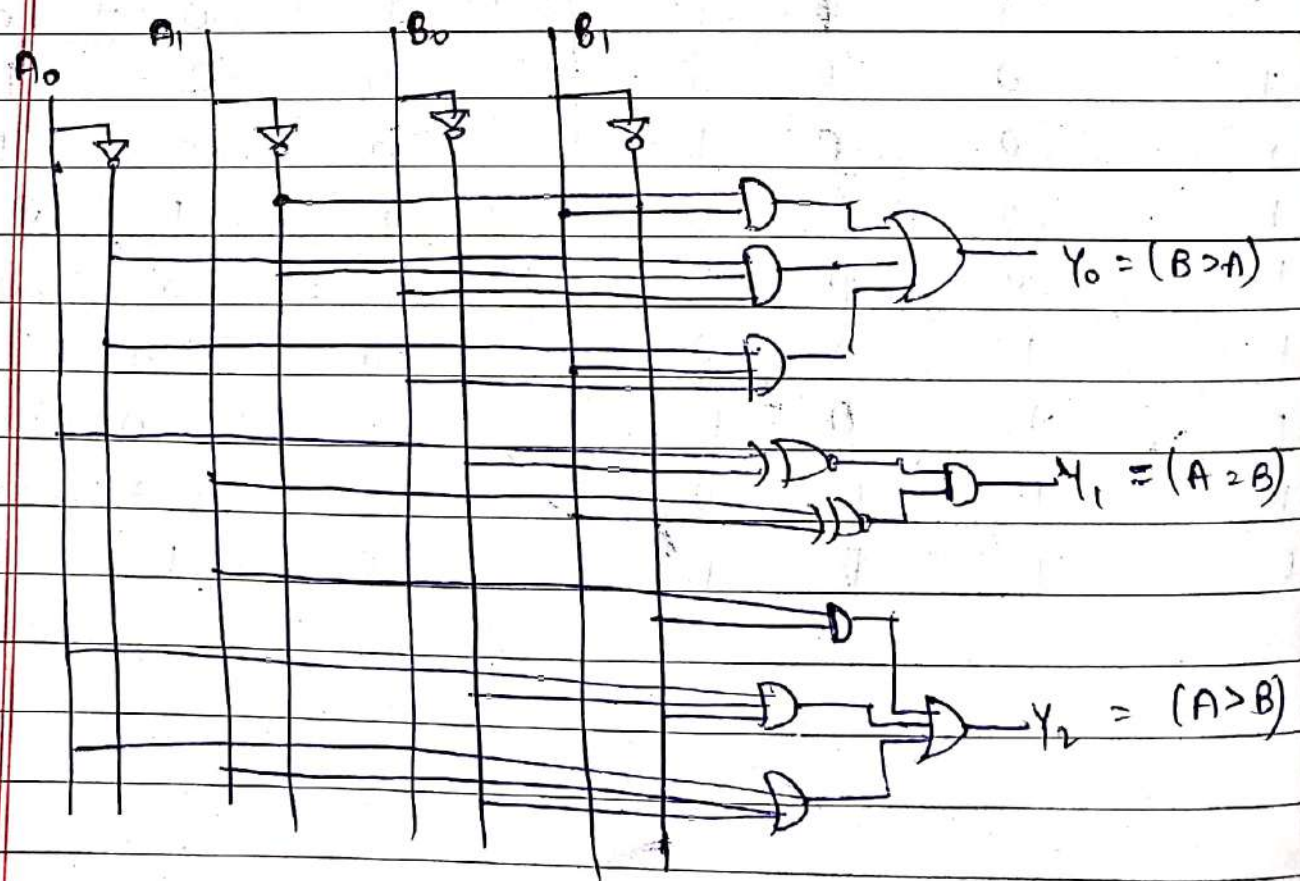
$$B > A = Y_0 = \bar{A}_1B_1 + \bar{A}_1\bar{A}_0B_0 + \bar{A}_0B_1B_0$$

$$Y_1 = (A_0 \odot B_0)(A_1 \odot B_1)$$

for $A \geq B$

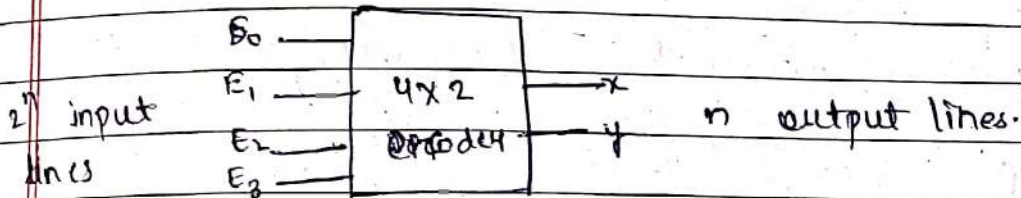
	$\bar{B}_1\bar{B}_0$	$\bar{B}_1B_0$	$B_1\bar{B}_0$	B_1B_0
$\bar{A}_1\bar{A}_0$	0	1	3	2
$\bar{A}_1A_0$	1	4	5	6
$A_1\bar{A}_0$	1	2	15	14
A_1A_0	1	1	10	11

$$Y_2 = A_1\bar{B}_1 + A_0\bar{B}_1\bar{B}_0 + A_1A_0\bar{B}_0$$



Introduction To Encoder And Decoder

Encoder: It is a combinational circuit that performs a reverse operation of a decoder. It has 2^n input and n output.

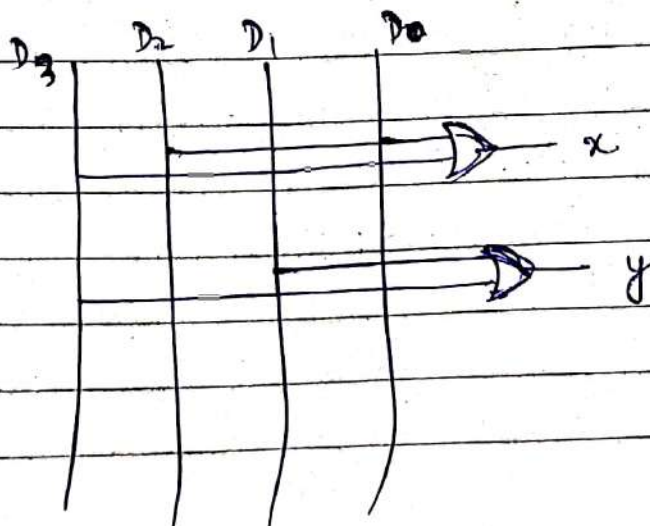


Types

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

Design Decimal To Binary Encoder.

I/p				O/p	
D ₀	D ₁	D ₂	D ₃	x	y
0	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1



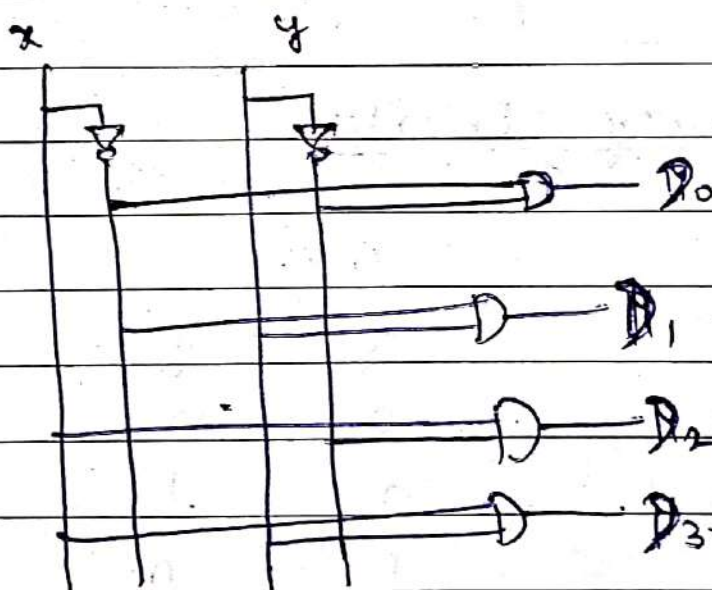
Decoder :-

It is a combinational circuit that convert Binary information from n input line to a maximum of 2^n output lines.
 $n \rightarrow 2^n$

Binary To decimal Decoder.

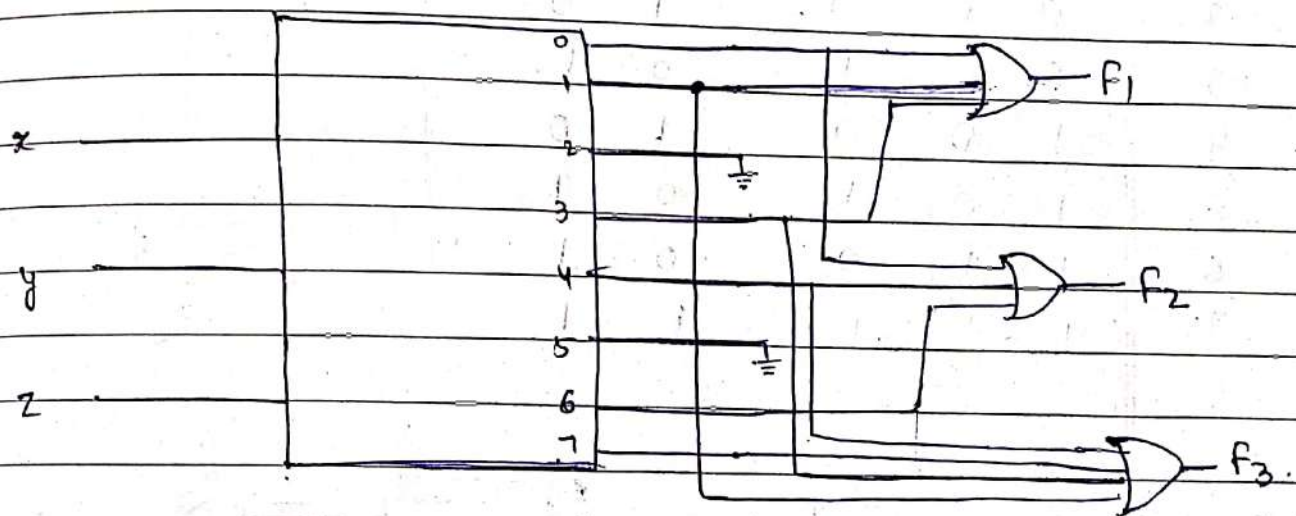
I/P

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



Ques Design a combinational circuit using 3×8 decoder having output's.

$$F_1 = \Sigma(0, 1, 3) \quad F_2 = \Sigma(0, 4, 6) \quad F_3 = \Sigma(1, 3, 4, 7)$$



Ques. $F(A, B, C) = F_1 = \bar{x}y + y$
 $F_2 = xyz + \bar{y}z$

$$F_1 = \bar{x}yz + \bar{x}y\bar{z} + xy + \bar{x}y$$

$$= \bar{x}yz + \bar{x}y\bar{z} + xyz + xy\bar{z} + \bar{x}yz + \bar{x}y\bar{z}$$

$$= \Sigma m(3, 2, 7, 6)$$

$$F_1 = \Sigma m(3, 2, 7, 6)$$

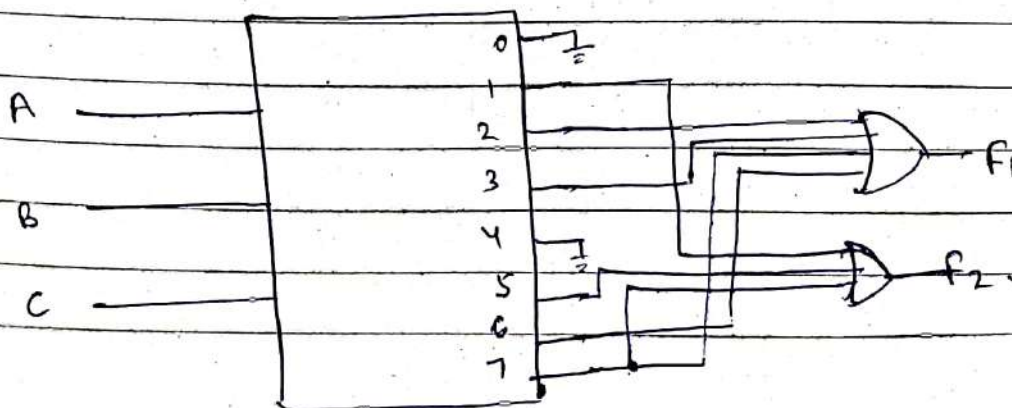
OR

	$\bar{y}z$	$y\bar{z}$	yz	$y\bar{z}$
$\bar{x}$	0	1	✓	✓
x	4	5	✓	✓

$$F_2 = xyz + x\bar{y}z + \bar{x}\bar{y}z$$

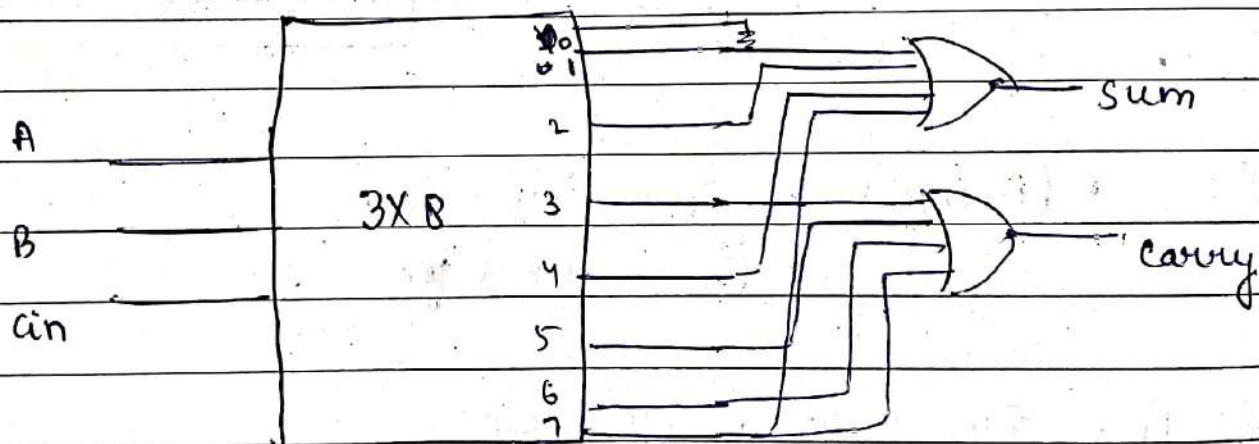
$$= \Sigma m(7, 5, 0)$$

$$F_2 = \Sigma m(7, 5, 0)$$



Full Adder Implementation Using Decoder

	A	B	C _{in}	S	C _o
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



Implement Boolean function using 4x1 mux
considered A and C as selection lines B & D as
input

$$f(A, B, C, D) = \sum m(1, 3, 4, 11, 12, 13, 14, 15)$$

	$\bar{A}\bar{C}$	$\bar{A}C$	$A\bar{C}$	AC		A	B	C	D	
	I_0	I_1	I_2	I_3						
$\bar{B}\bar{D}$	0	2	8	10		0	0	0	0	$\rightarrow 0$
$\bar{B}D$	(1)	(3)	9	(11)		0	0	0	1	1
$B\bar{D}$	(4)	6	(12)	(14)		0	0	1	0	2
BD	5	7	(13)	(15)		0	0	1	1	3
						0	1	0	0	4
						0	1	0	1	5
						0	1	1	0	6
						0	1	1	1	7
						1	0	0	0	8
						1	0	0	1	9
						1	0	1	0	10
						1	0	1	1	11
						1	1	0	0	12
						1	1	0	1	13
						1	1	1	0	14
						1	1	1	1	15

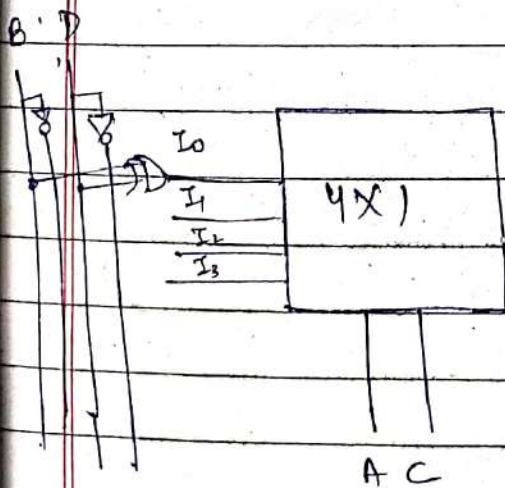
$$I_0 = \bar{B}D + B\bar{D}$$

$$I_0 = B \oplus D$$

$$I_1 = \bar{B}D$$

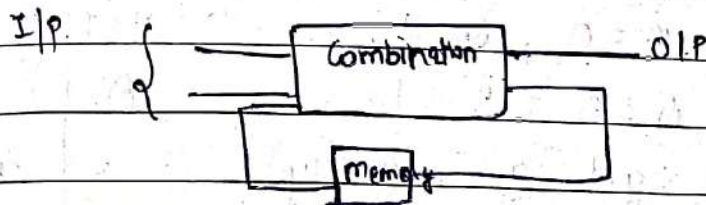
$$I_2 = B\bar{D} + BD$$

$$I_3 = \bar{B}D + B\bar{D} + BD$$



Sequential circuits :-

Are the logic circuit whose output depends on present input and previous output.



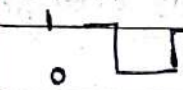
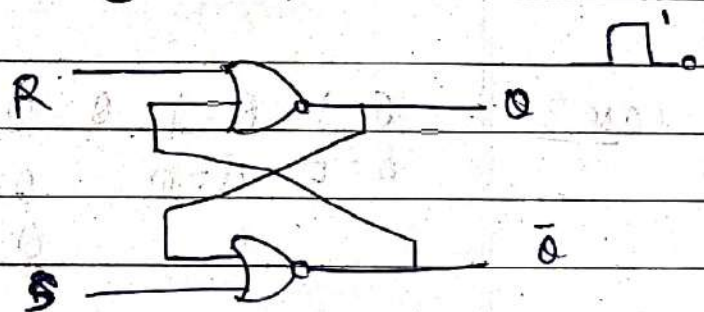
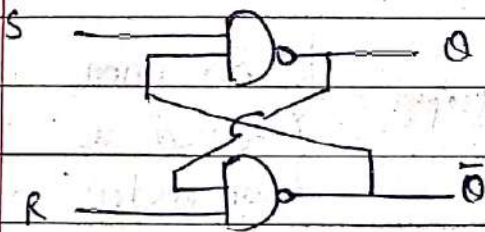
SR latch

The basic storage element is called latch.

It is of two types

① NAND LATCH

② NOR LATCH



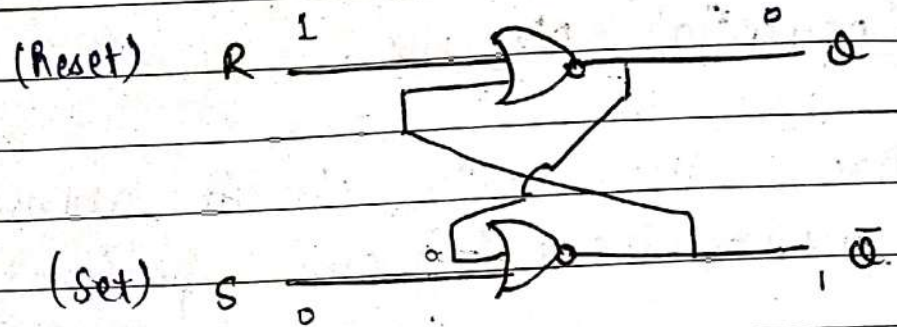
work on negative logic



→ work on (+) logic

NOR LATCH

NOR LATCH



Case 1: $S=0, R=1, Q=0, \bar{Q}=1$
 $S=0, R=0, Q=0, \bar{Q}=1$
 (So, Here we get a memory)

T.T for NOR gate

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

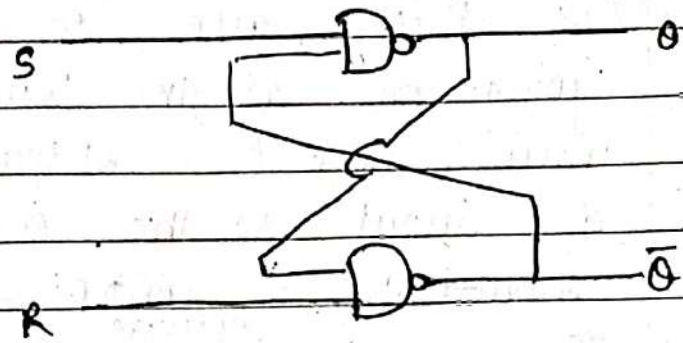
Case 2: $S=1, R=0, \bar{Q}=0, Q=1$
 $S=0, R=0, \bar{Q}=0, Q=1$
 (So, again Here we get a memory)

Case 3: $S=1, R=1, Q=0, \bar{Q}=0$ Not acceptable } So, not
 $S=0, R=0, Q=0, \bar{Q}=1$ used to
 $S=0, R=0, Q=0, \bar{Q}=0$ in order to
 store the data
 b/c it is changing

Summary

S	R	Q	$\bar{Q}$
0	0	memory (before)	
0	1	0	1
1	0	1	0
1	1	(Not used).	

NAND LATCH



Case I: $S=0$ $R=1$ $Q=1$ $\bar{Q}=0$ TT. for NAND GATE

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Case 2: $S=1$ $R=0$ $Q=0$ $\bar{Q}=1$

Case 3: $S=0$ $R=0$ $Q=1$ $\bar{Q}=1$ } not acceptable

$S=1$ $R=1$ $Q=0$ $\bar{Q}=1$

$Q=1$ $\bar{Q}=0$

→ Here this case we will not use b/c it is not storing our data

Summary

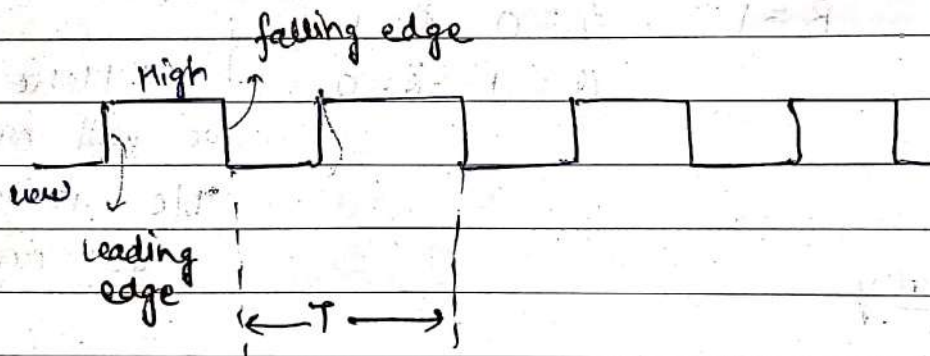
S	R	Q	$\bar{Q}$
0	0	(Not used)	
0	1	1	0
1	0	0	1
1	1	(memory) (before)	

What is clock :-

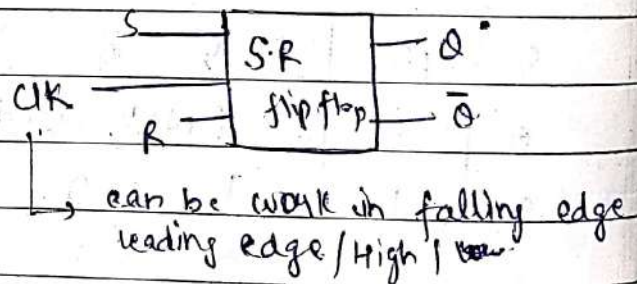
As there are so many processes are going on at the same time in sequential circuit then As output will be act as a input to the other so, there should be a control over the processes we ~~it~~ cannot ^{operate} ~~process~~ it randomly.

So, need clocking to time circuit In other words we need sequence of the circuit.

We are calling it clock b/c it select the time of input



$$\uparrow f = \frac{1}{T} \downarrow$$



Duty cycle :- Ratio of time.

$$= \frac{\text{Signal (or time at which is in working state)}}{\text{Total time}}$$

$$= \frac{t/2}{t} = \frac{1}{2} = 50\%$$

Triggering Methods IN FLIP FLOP



To change the memory (already stored the data)

Types of Triggering Methods

① Level Triggering

→ Whenever our clock level is high there is change in LATCH/FLIP FLOP

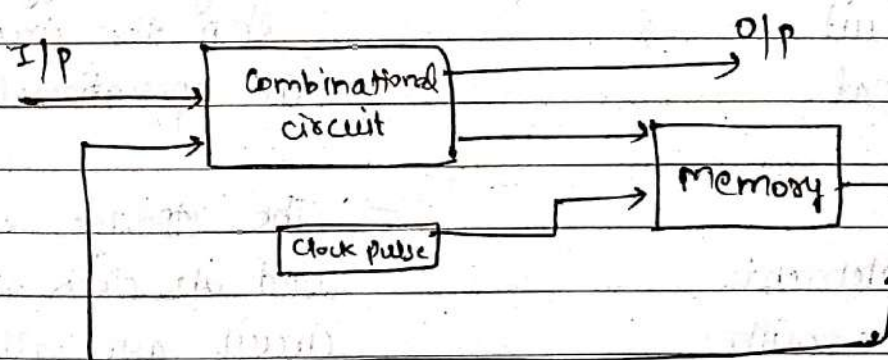
② Edge Triggering

→ ① +ve edge "

② -ve edge "

(+ve) edge triggering when clock is move from Low $\rightarrow$ High then there will a change in memory.

-ve edge triggering when clock is move from High $\rightarrow$ Low then there will a change in memory.

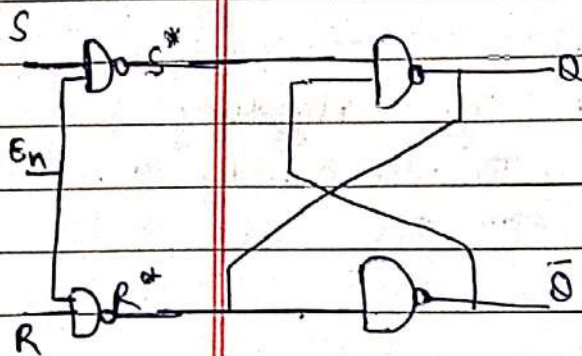


Four Types of Flip FLOP

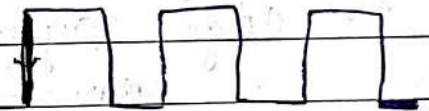
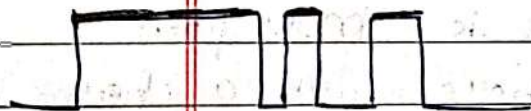
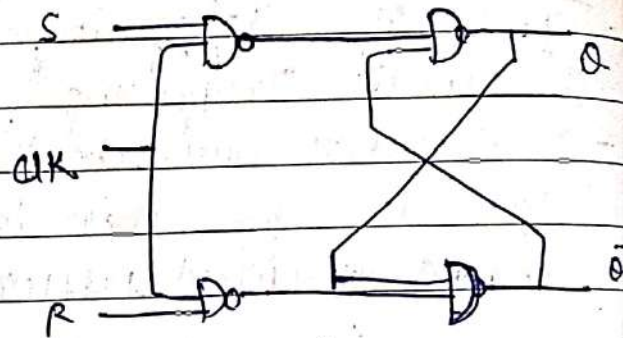
- ① S-R flip flop
- ② J-K flip flop
- ③ D - flip flop
- ④ T - flip flop

Difference between LATCH AND FLIP FLOP

LATCH



FLIP FLOP



It will work when enable is high

It will work when clock is changing either $L \rightarrow H$ or $H \rightarrow L$.

→ Level Triggered then the circuit is operational

→ edge Triggered then the circuit is operational

from left

→ storage element that operate with signal level.

→ The storage element used in clock sequential circuit are called flip flop.

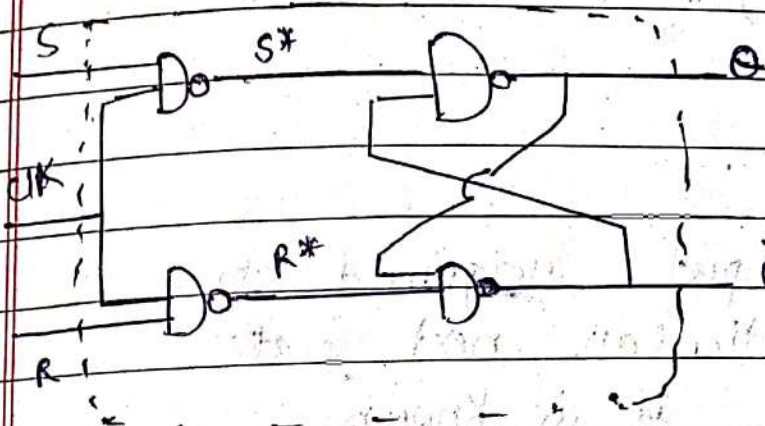
Definition: It is binary storage device capable of storing 1 bit of information.

→ It maintain a binary state as long as power is derived to the circuit.

Introduction To SR - FLIP FLOP.

Using NAND GATE

SR LATCH WITH NAND



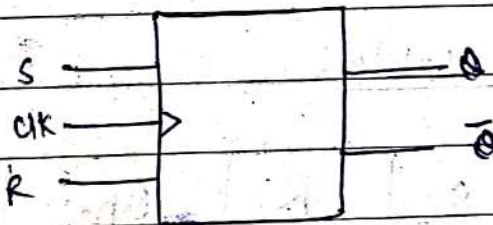
S*	R*	Q	Q̄
0	0	Not used	
0	1	1	0
1	0	0	1
1	1	memory	

CLK @ S R

$$S^* = (\overline{S \cdot CLK}) = \overline{S} + \overline{CLK}$$

$$R^* = (\overline{R \cdot CLK}) = \overline{R} + \overline{CLK}$$

CLK	S	R	Q	Q̄
0	*	x	memory	
1	0	0	memory	
1	0	1	0	1
1	1	0	1	0
1	1	1	Not used	



S=0	R=0	S=0	R=1
S*=1	R*=1	S*=1	R*=0

S=1	R=0	S=1	R=1
S*=0	R*=1	S*=0	R*=0

Characteristic Table :-

It defines the next state ^{as} of a function of the input and present state (used to analyse any circuit).

Excitable Table :-

It defines the input required to generate any particular next state and ^{when} present state ~~if~~ is known.

→ Truth Table, Characteristic Table, Excitation Table for S-R flip flop.

Truth table :

clk	S	R	Q_{n+1} (next state)
0	X	X	Q_n (Present state)
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Indetd Indetermined

Characteristic Table (Assuming clock at High)

	Q_n	S	R	Q_{n+1}
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	Indetermined
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	Indetermined

Excitation Table

	Q_n	Q_{n+1}	S	R
0	0	0	0	X
1	0	1	1	0
2	1	0	0	1
3	1	1	X	0

Characteristic Equation of S-R flip flop

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

$$= (S + \bar{R})(S + Q_n)$$

	$\bar{S}\bar{R}$	$\bar{S}R$	SR	$S\bar{R}$
$\bar{Q}_n$	0	1	1	2
Q_n	4	5	7	6

$$= S\bar{Q}_n + Q_n\bar{R}$$

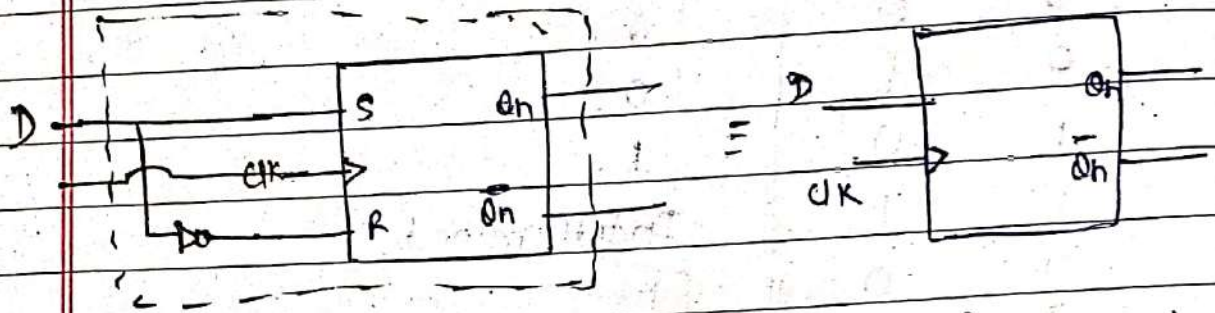
$$Q_{n+1} = \bar{Q}_n R$$

	$\bar{S}\bar{R}$	$\bar{S}R$	SR	$S\bar{R}$
$\bar{Q}_n$	0	1	X	2
Q_n	4	5	X	6

$$S\bar{R} + \bar{R}Q_n$$

$$= \bar{R}(S + Q_n)$$

Introduction To D-Flip flop



T.T for D flip flop

clk	D	Q_{n+1}
0	x	Q_n
1	0	0
1	1	1

for understanding only
T.T for S-R flip

clk	S	R	Q_{n+1}
0	x	x	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Indeterminate

characteristic Table

Assuming clock is high

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

in expression form $Q_{n+1} = D$

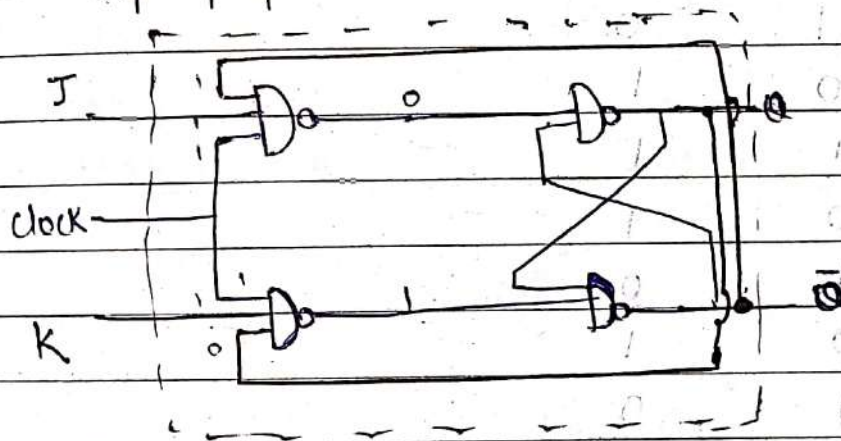
Excitation Table

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

Introduction To J-K flip flop

(intermined state)

→ In order to remove the disadvantage of S-R flip flop we make J-K flip flop.



clock = 0 memory

clock = 1, $J = 1$, $K = 0$ it will act same as S-R where $Q = 1$, $\bar{Q} = 0$

clock = 1, $J = 0$, $K = 1$, $Q = 0$, $\bar{Q} = 1$

clock = 1, $J = 1$, $K = 1$

assuming $Q = 0$, $\bar{Q} = 1$

→ $Q = 1$, $\bar{Q} = 0$

$Q = 0$, $\bar{Q} = 1$

$Q = 1$, $\bar{Q} = 0$

always complement next state

Truth Table

CLK	J	K	Q_{n+1}
0	x	x	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	$\overline{Q_n}$

Characteristic Table

	Q_n	J	K	Q_{n+1}
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Excitation Table

	Q_n	Q_{n+1}	J	K
0	0	0	0	x
1	0	1	1	x
2	1	0	x	1
3	1	1	x	0

for J

	Q_{n+1}	$\bar{Q}_{n+1}$
$\bar{Q}_n$		1
Q_n	X	X

for K

	Q_{n+1}	$\bar{Q}_{n+1}$
$\bar{Q}_n$	X	X
Q_n	1	0

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

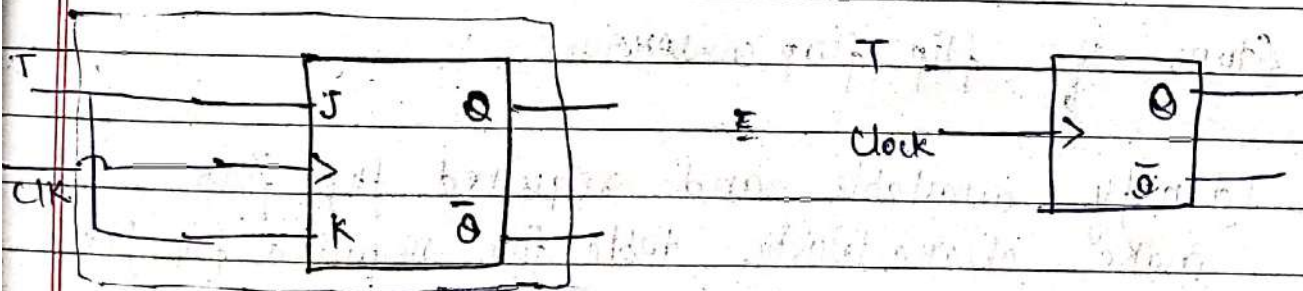
$$K = \bar{Q}_{n+1}$$

$2^3 = 8$ cell K-map

	$\bar{J}\bar{K}$	$\bar{J}K$	JK	$J\bar{K}$
$\bar{Q}_n$			1	1
Q_n	1			1

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

Introduction To T flip flop
 ↓
 Toggle



Truth Table for T flip flop

Clock	T	Q_{n+1}
0	X	Q_n
1	0	Q_n
1	1	$\bar{Q}_n$

Characteristic Table

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

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

Excitation Table

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

Steps of flip flop conversion

- ① Identify available and required flip flop
- ② make characteristic table for required flip flop
- ③ make excitation table for available flip flop
- ④ write Boolean expression for available flip flop
- ⑤ Draw the circuit

flip flop conversion

① JK to D flip flop conversion.

available flip flop = J.K
required flip flop = D.

characteristic Table of D

	Q_n	D	Q_{n+1}	J	K		Q_n	Q_{n+1}	J	K
0	0	0	0	0	X	0	0	0	0	X
1	0	1	1	1	X	0	1	1	1	X
2	1	0	0	X	1	1	0	X	1	0
3	1	1	1	X	0	1	1	X	0	0

for J.
 $\bar{D}$ D

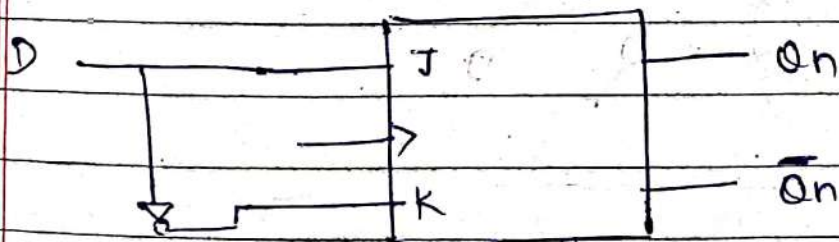
$\bar{Q}_n$	0	1
Q_n	X	X

for K.
 $\bar{D}$ D

$\bar{Q}_n$	X	X
Q_n	1	0

$$J = D$$

$$K = \bar{D}$$



② T to D flip flop

available flip flop = T

required flip flop = D

Characteristic Table of D flip flop

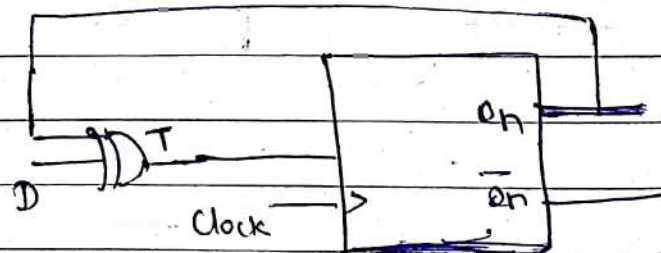
	Q_n	D	Q_{n+1}	T
0	0	0	0	0
1	0	1	1	1
2	1	0	0	1
3	1	1	1	0

Excitation table of T flip flop

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

for ~~D~~ T

	$\bar{D}$	D
$\bar{Q}_n$	0 1	1
Q_n	1 2	3



$$T = \bar{Q}_n D + Q_n \bar{D}$$

$$T = Q_n \oplus D$$

③ SR to JK flip flop

available flip flop = SR
required flip flop = JK

characteristic table
of JK flip flop

	Q_n	J	K	Q_{n+1}	S	R
0	0	0	0	0	0	X
1	0	0	1	0	0	X
2	0	1	0	1	1	0
3	0	1	1	1	1	0
4	1	0	0	1	X	0
5	1	0	1	0	0	1
6	1	1	0	1	X	0
7	1	1	1	0	0	1

for S

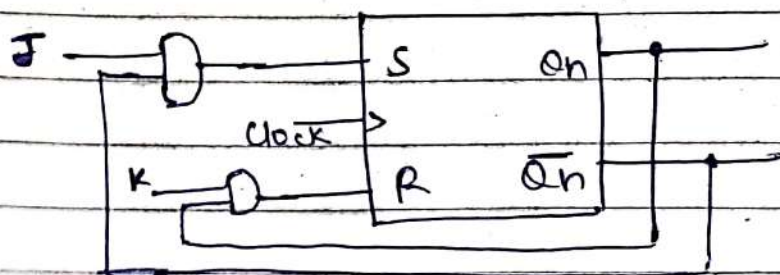
	$\bar{J}\bar{K}$	$\bar{J}K$	$J\bar{K}$	JK
$\bar{Q}_n$	0	1	1	1
Q_n	X	4	5	7

for R

	$\bar{J}\bar{K}$	$\bar{J}K$	$J\bar{K}$	JK
$\bar{Q}_n$	X	0	X	1
Q_n	4	1	5	7

$$S = \bar{Q}_n J$$

$$R = Q_n K$$



④

SR to T flip flop

Available = SR

Required = T

Characteristic Table

of T

	Q_n	T	Q_{n+1}	S	R
0	0	0	0	0	X
1	0	1	1	1	0
2	1	0	1	X	0
3	1	1	0	0	1

for S

Q_n	$\overline{Q_n}$	T
0	1	1
1	0	0

for R

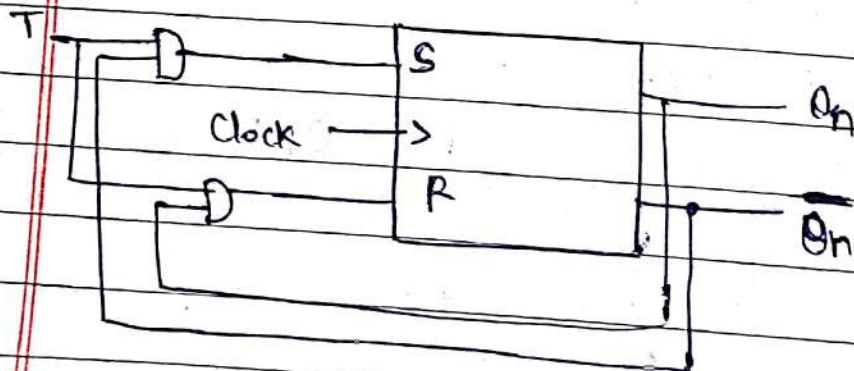
Q_n	$\overline{Q_n}$	T
0	1	0
1	0	1

$$S = \overline{Q_n} T$$

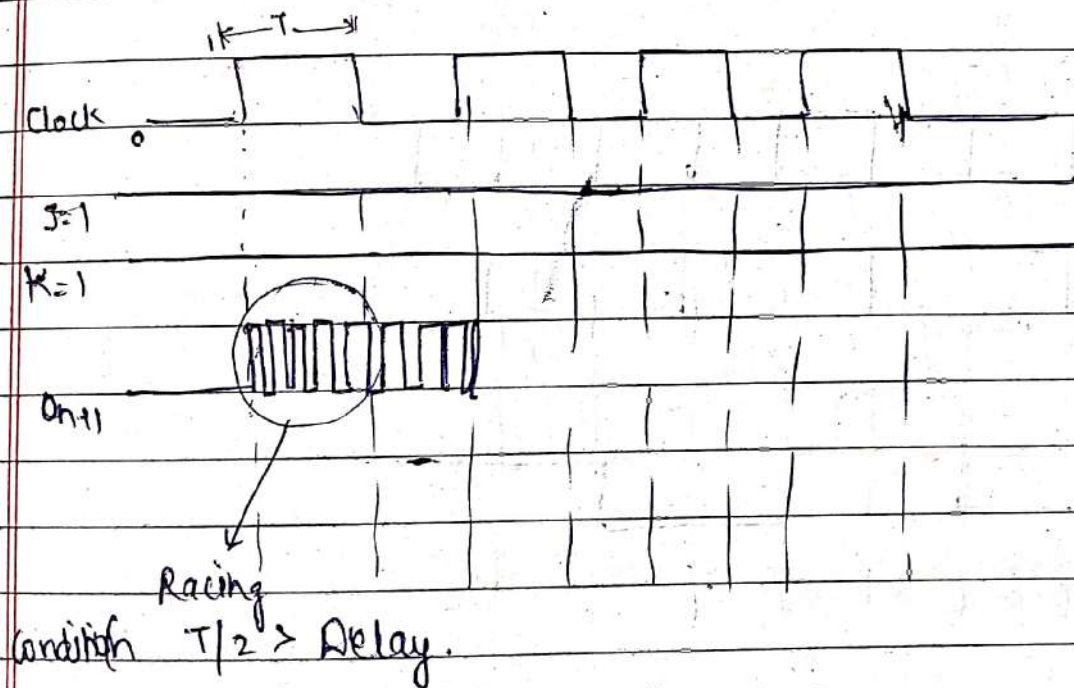
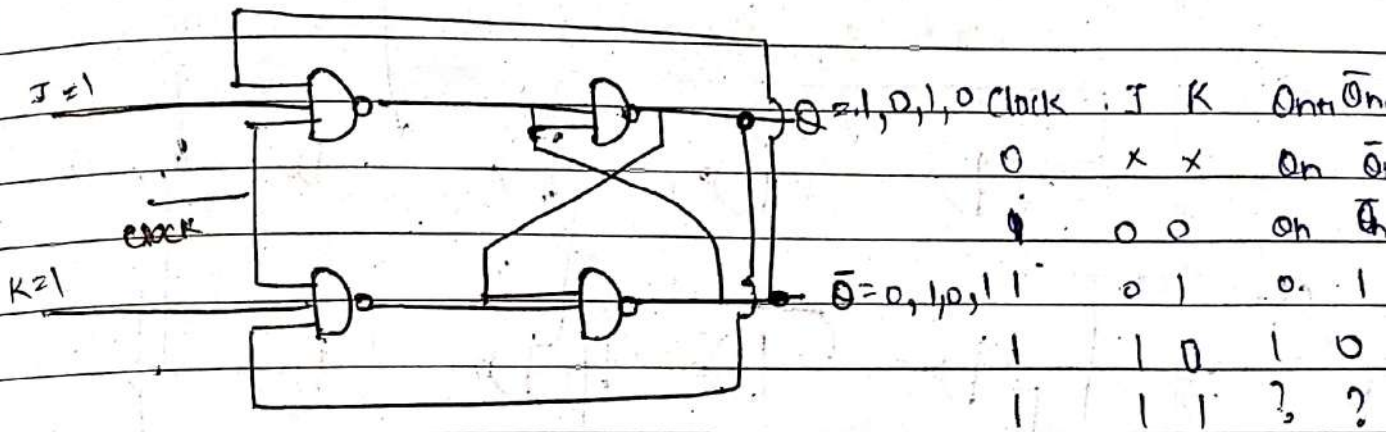
$$R = Q_n T$$

$$S = \overline{Q_n} T$$

$$R = Q_n T$$



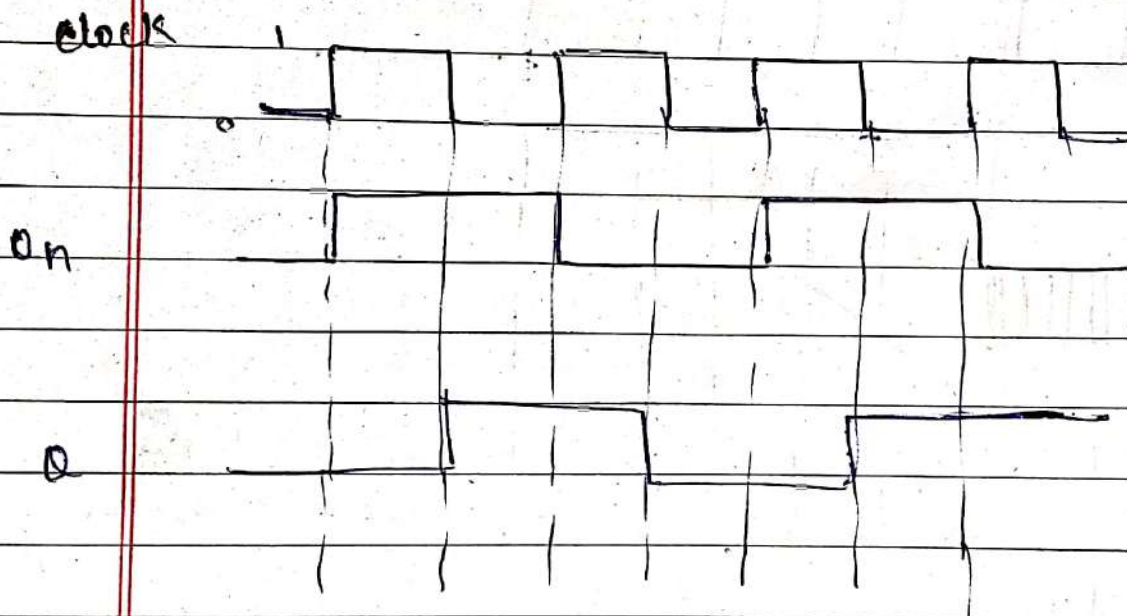
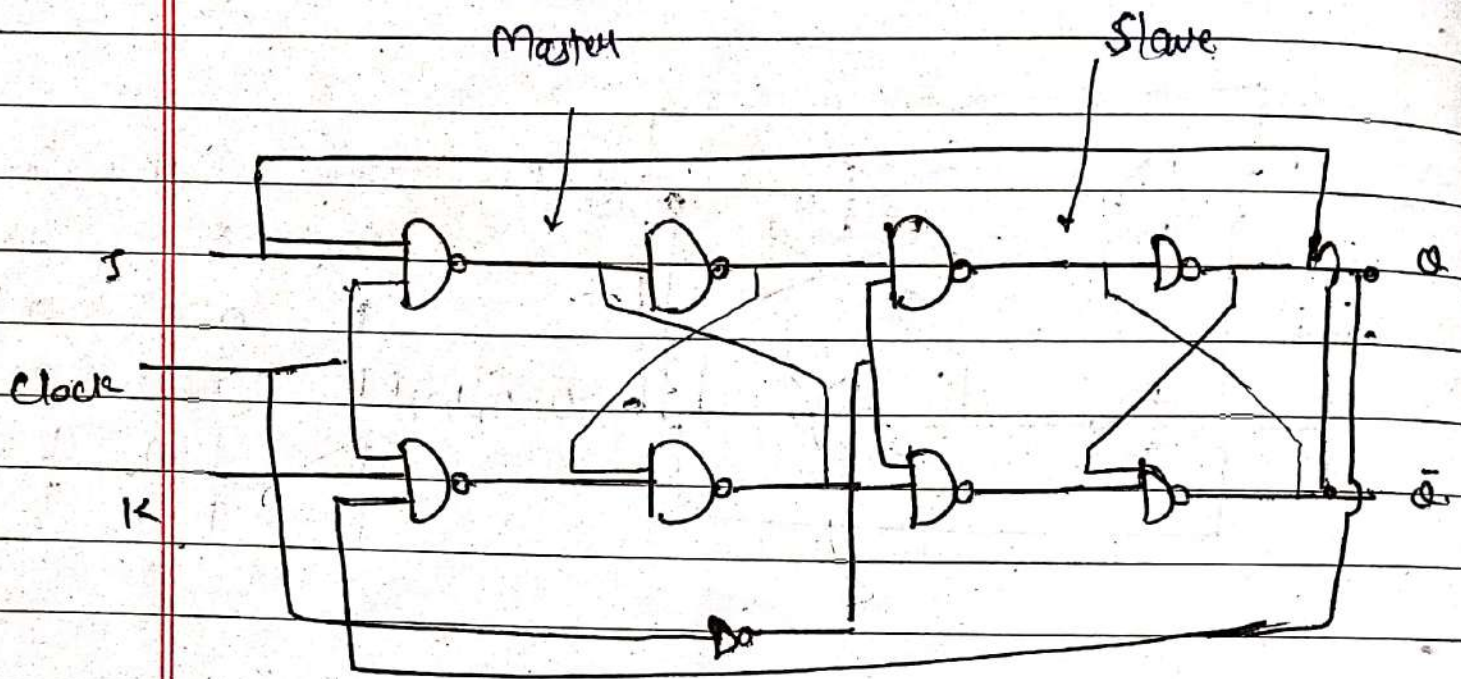
Race Around Condition



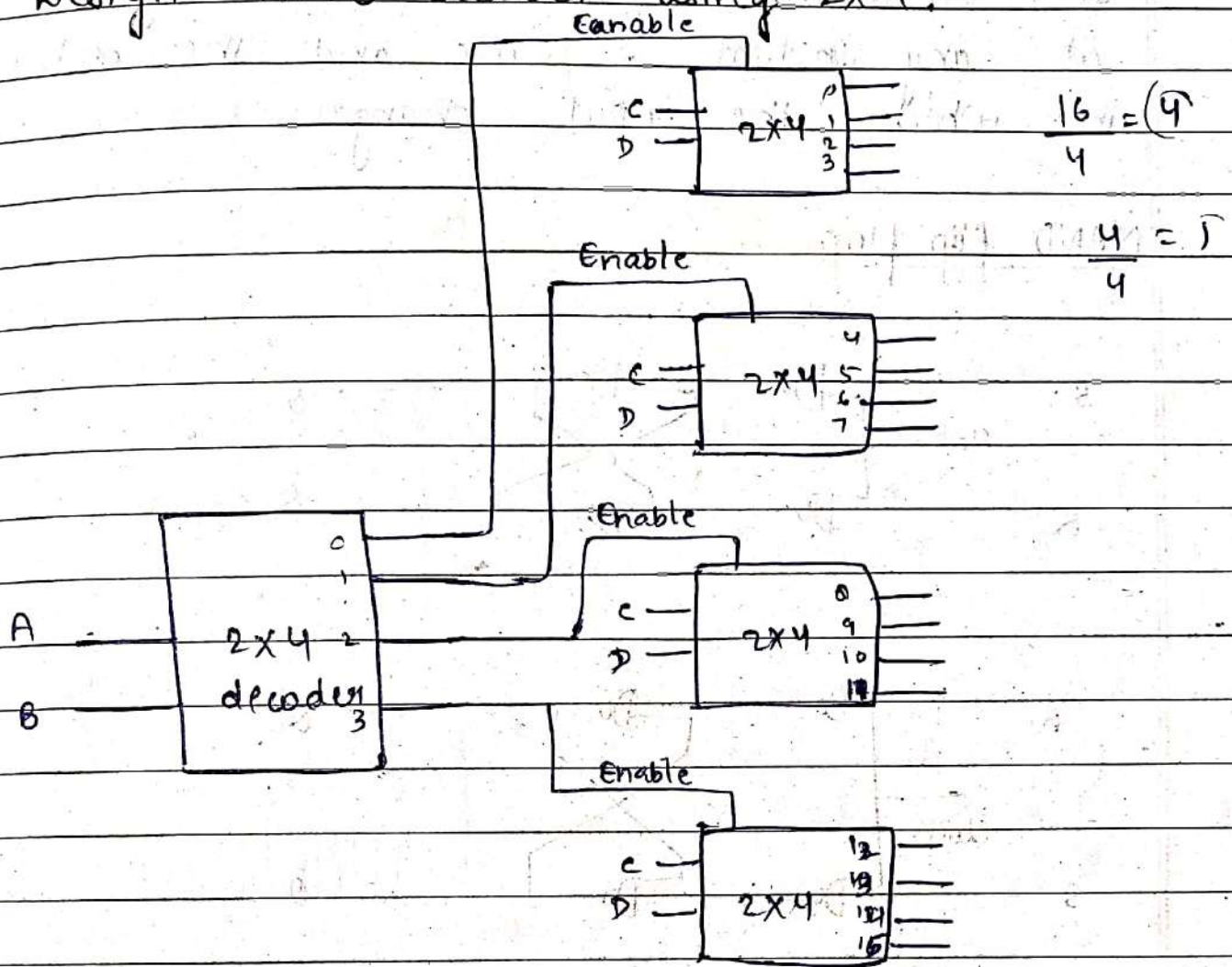
Condition to Overcome Racing

- ① $T/2 < \text{Delay}$ (Not practical, so, not used)
- ② edge Triggering
- ③ master slave [this operation is similar to -ve edge triggering]

Master Slave J-K flip flop



Design 4x16 decoder using 2x4.



★ Sequential circuit are of two types :-

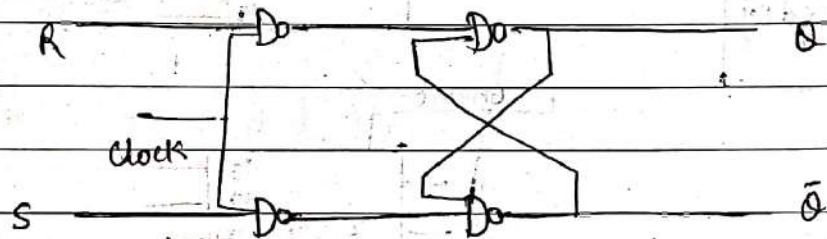
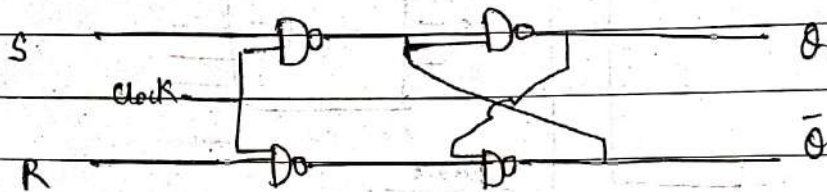
(i) Synchronous sequential circuit :- It is circuit whose behaviour/output can be defined from the knowledge of its input at discrete instant of time.

→ Synchronisation is achieved by a timing device called clock generator which provides clock signal in the form of periodic train of clock pulses.

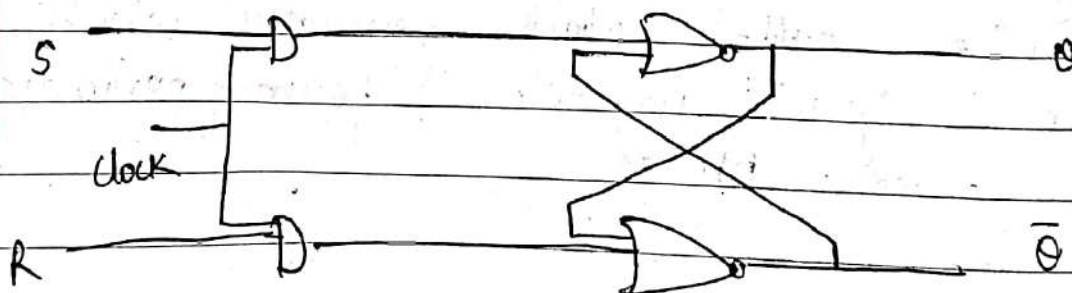
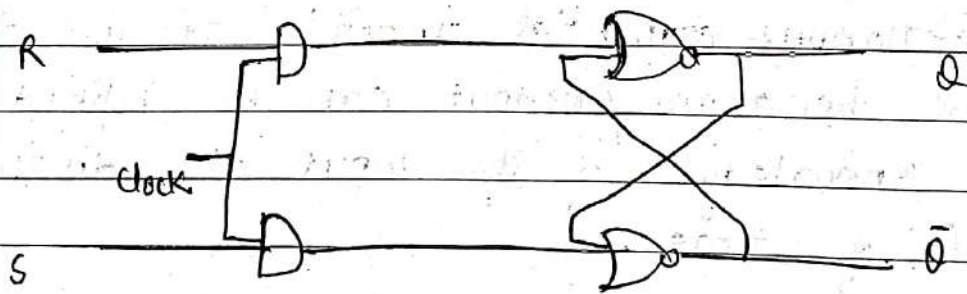
Asynchronous sequential circuit :-

Its behaviour depends on input signal at any instant of time and the order in which the input changes.

NAND flip flop



NOR flip flop



Design J-K flip flop using S-R NOR flip flop.

required

available

		S	K	Qn	S	R
Qn	J					
0	0	0	0	0	0	X
1	0	0	1	0	0	X
2	0	1	0	1	1	0
3	0	1	1	1	1	0
4	1	0	0	1	X	0
5	1	0	1	0	0	0
6	1	1	0	0	X	0
7	1	1	1	0	0	1

for S

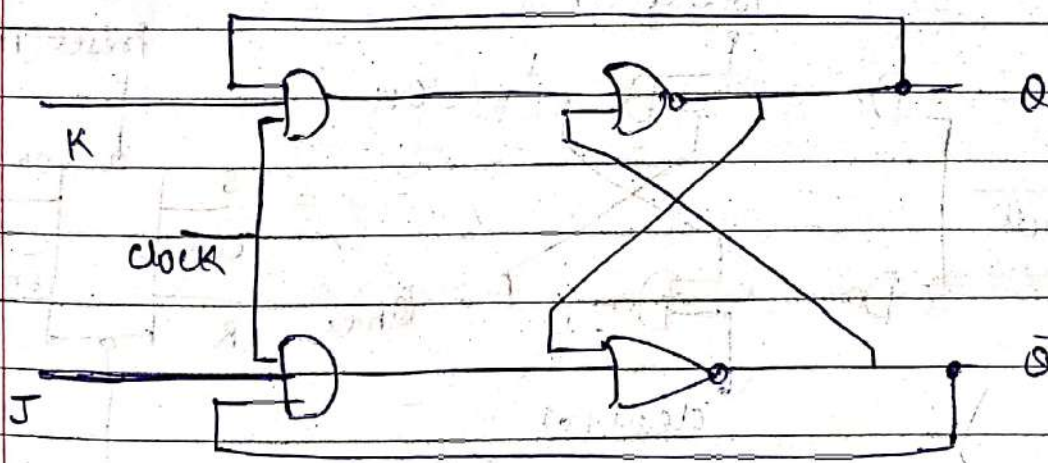
	$\bar{J}\bar{K}$	$\bar{J}K$	JK	$J\bar{K}$
$\bar{Q}_n$	0	1	1	1
Q_n	X	0	0	X

for R

	$\bar{J}\bar{K}$	$\bar{J}K$	JK	$J\bar{K}$
$\bar{Q}_n$	X	0	0	1
Q_n	0	1	1	0

for S = $\bar{Q}_n J$

for R = $Q_n K$



Preset And clear I/P.

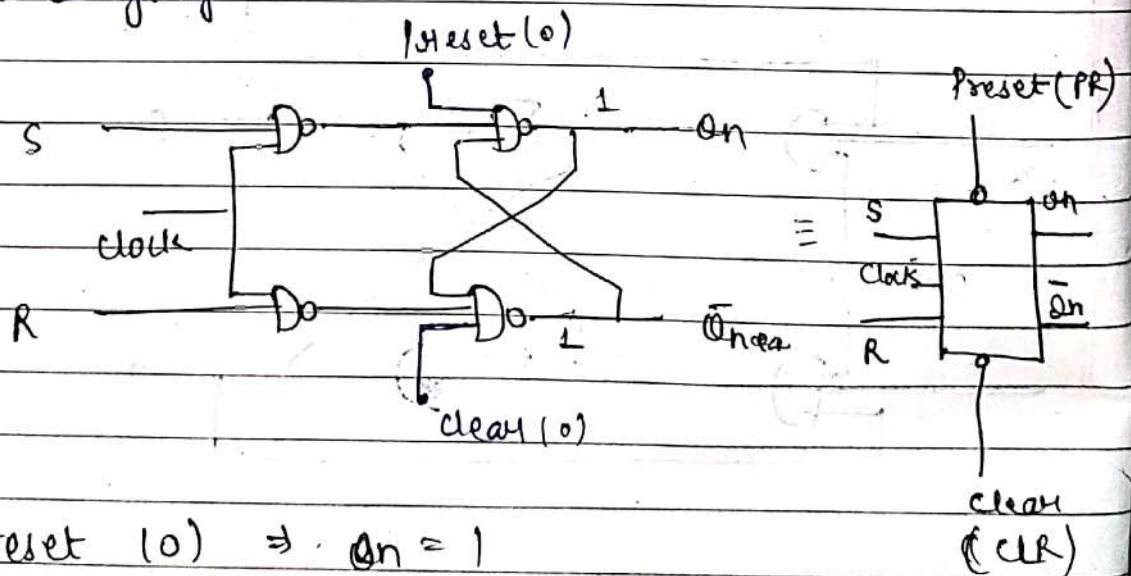
* It is necessary to learn b/c we use them in counter [as counter is count in decreasing order].
then, we need already stored value.

* Whenever preset and clear come into picture means, whatever be the value of S and R.

The value of Qn is changed ~~not~~ directly it will not see what is the value of S and R and clock, it will change directly.

Asynchronous I/P Synchronous I/P
or overridding I/P are S, R, J, K, D & T
or direct input

→ Preset and ~~set~~ ^{clear} come into ~~as~~ asynchronous ~~the~~ category.



preset (0) $\Rightarrow Qn = 1$

clear (0) $\Rightarrow Qn = 0$ b/c $\bar{Qn} \neq 1$

we will get this O/P whatever be the value of clock and synchronous inputs.

Truth Table

Preset	Clean	Qn
0	0	x (Not used)
0	1	1
1	0	0
1	1	flip flop will perform normally

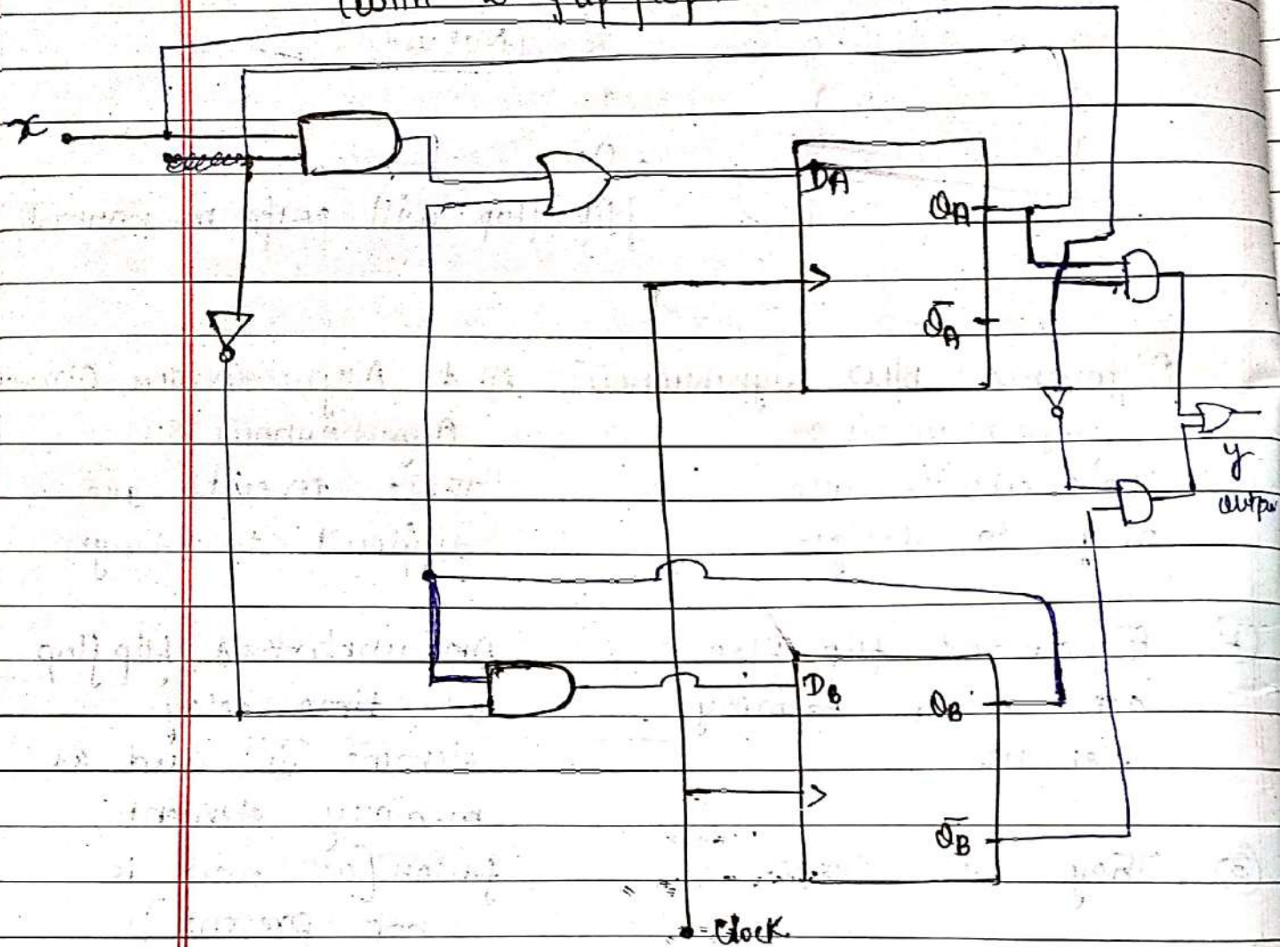
Difference B/w Synchronous And Asynchronous Circuit

Synchronous S.C

Asynchronous S.C

- | | |
|--|--|
| <p>① These circuits are easy to design</p> <p>② A clocked flip-flop acts as a memory element</p> <p>③ They are slower</p> <p>④ The status of memory element is affected only at the active edge of clock, if input is changed</p> <p>⑤ flip flop used in synchronous circuit</p> | <p>These circuits are difficult to design</p> <p>An unlocked flip flop or time delay element is used as memory element faster (as clock is not present)</p> <p>⑥ The status of memory element will change any time as soon as input is changed</p> <p>⑤ Latches are used in asynchronous circuit</p> |
|--|--|

Analysis of clocked sequential circuits (with D flip flop).



$$D_A(t+1) = Q_A x + Q_B$$

$$D_B(t+1) = Q_B \bar{Q}_A$$

$$y = \bar{x} \bar{Q}_B + x Q_A$$

for (A(t+1))

	$\bar{B}\bar{x}$	$\bar{B}x$	Bx	$B\bar{x}$
$\bar{A}$	0	1	✓ 3	✓ 2
A	4	✓ 5	✓ 7	✓ 6

for (B(t+1))

	$\bar{B}\bar{x}$	$\bar{B}x$	Bx	$B\bar{x}$
$\bar{A}$	0	1	✓ 3	✓ 2
A	4	5	7	6

	$\bar{B}\bar{x}$	$\bar{B}x$	Bx	$B\bar{x}$
$\bar{A}$	✓ 0	1	3	2
A	✓ 4	✓ 5	7	6

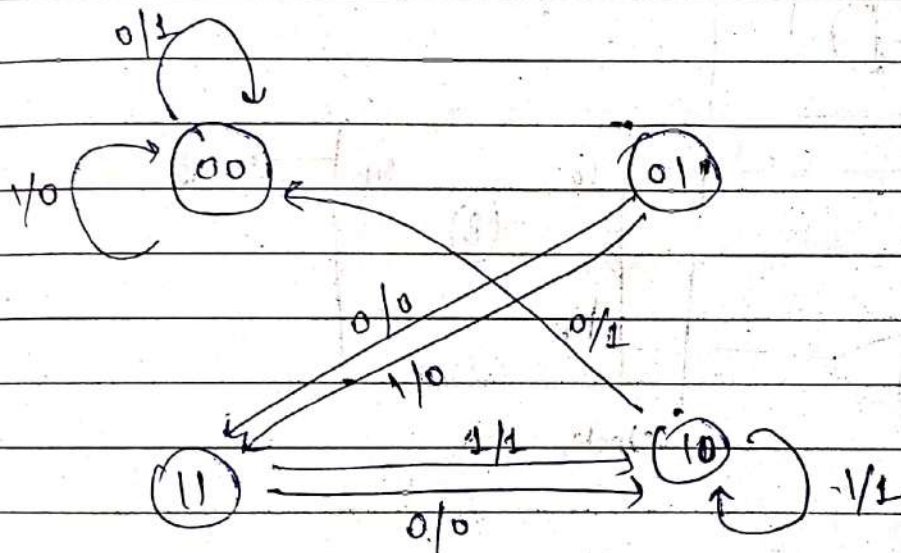
In D flip flop $Q_{n+1} = D$

Page No.

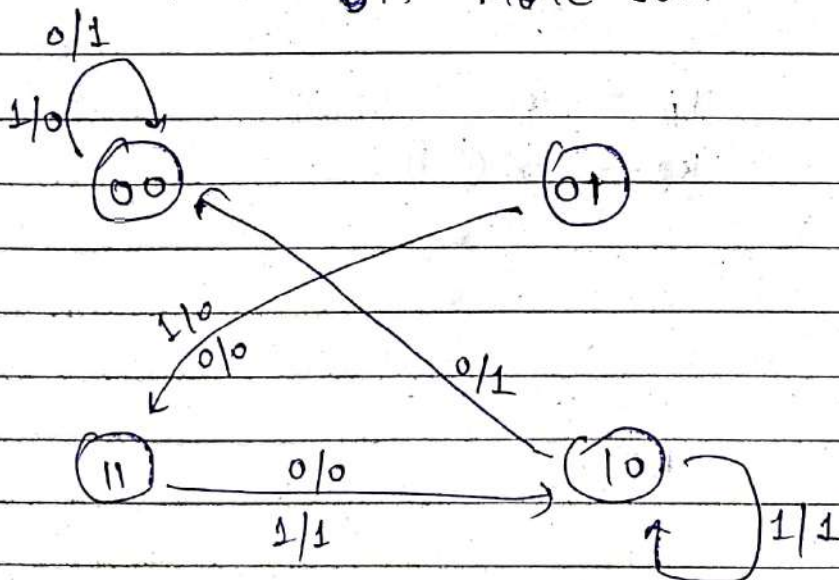
your writing partner

Date / /

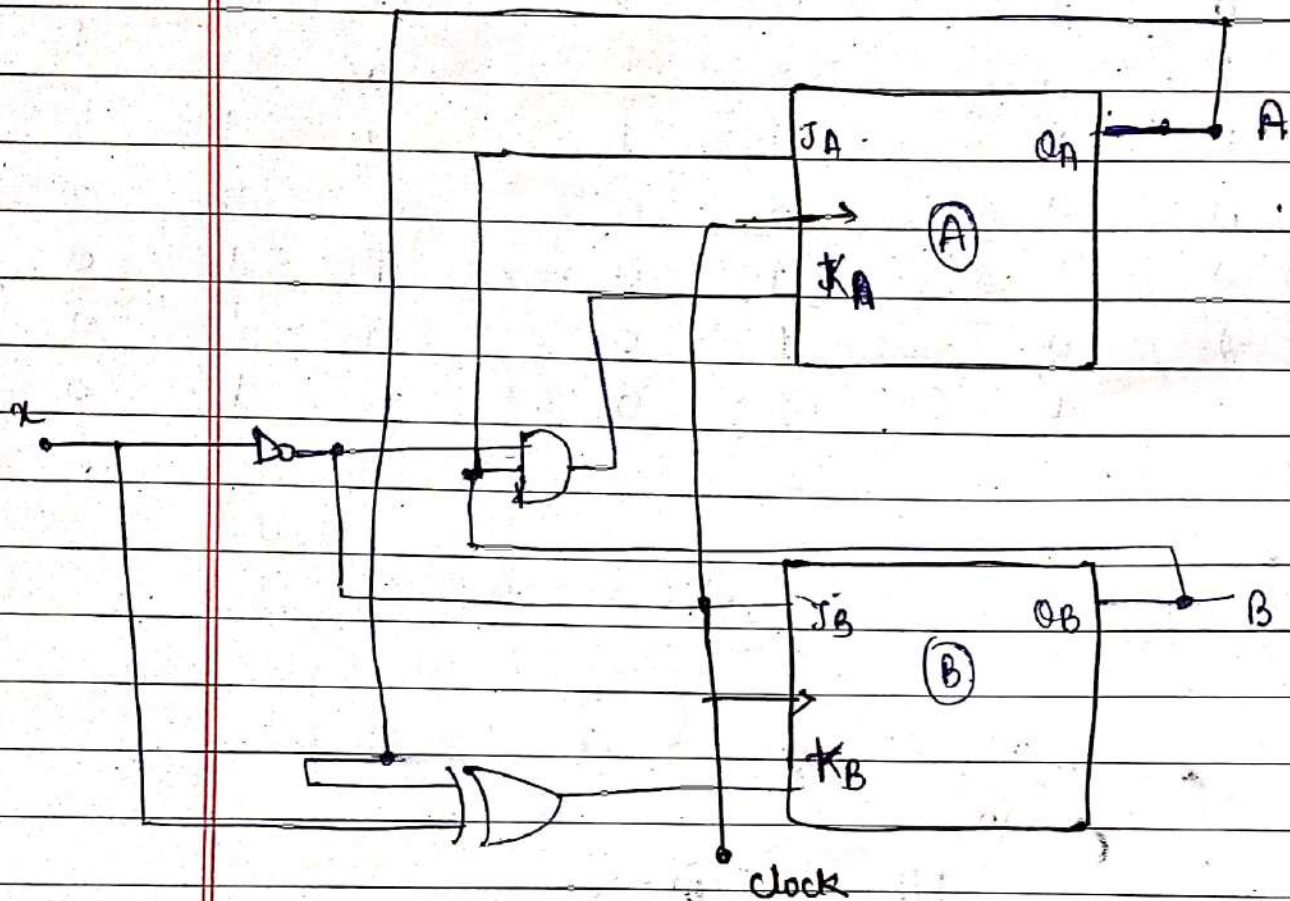
	Present state		I/p x	Next state		output (y)		
	A	B		A	B	y	D _A	D _B
0	0	0	0	0	0	1	0	0
1	0	0	1	0	0	0	0	0
2	0	1	0	1	1	0	1	1
3	0	1	1	1	1	0	1	1
4	1	0	0	0	0	1	0	0
5	1	0	1	1	0	1	1	0
6	1	1	0	1	0	0	1	0
7	1	1	1	1	0	1	1	0



OR more suitable



Analysis of Clocked Sequential Circuits (with JK flip flop).



$$A(t+1) = A \oplus x$$

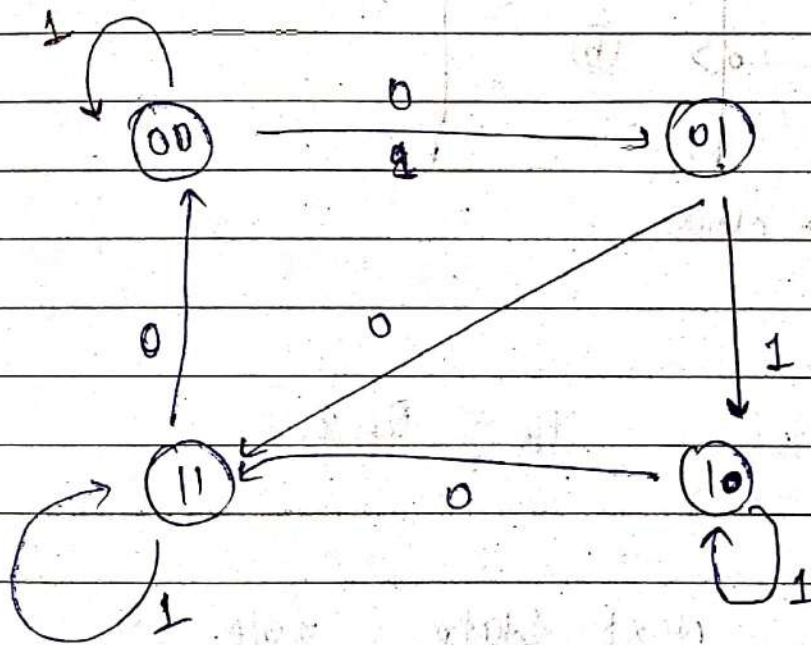
$$J_A = B$$

$$K_A = \bar{x} B$$

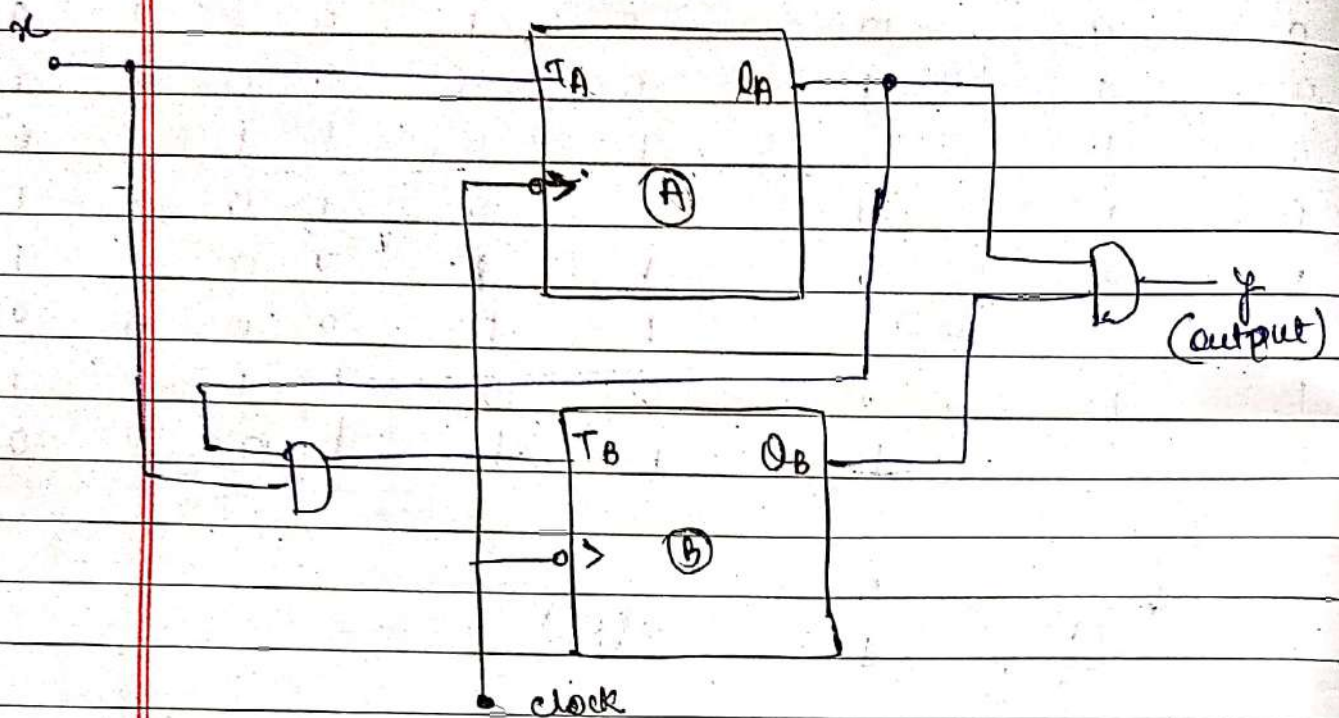
$$J_B = \bar{x}$$

$$K_B = x \oplus A$$

Present State		I/P x	Next State		O/P			
A	B		A	B	J _A	K _A	J _B	K _B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	0	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0



Analysis of clocked sequential circuit with T flip flop.

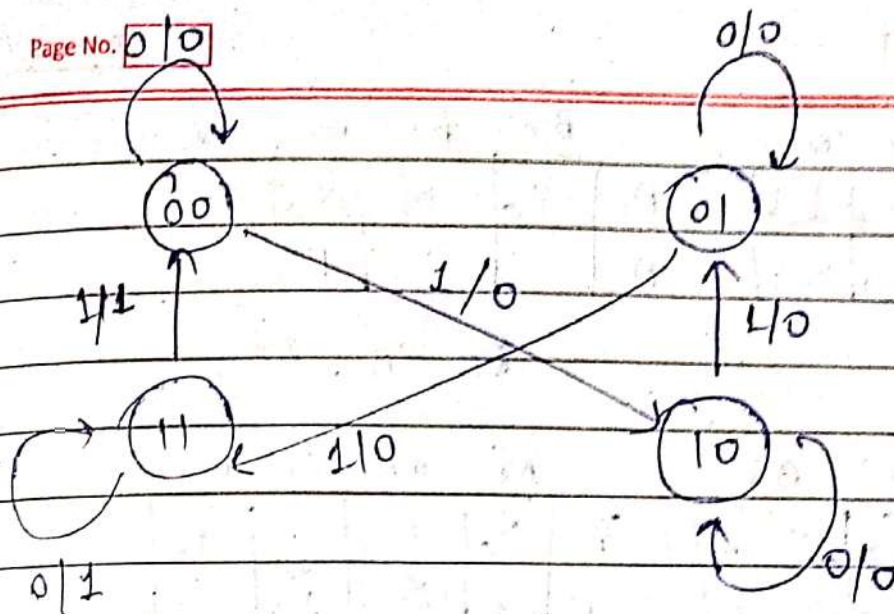


$$T_A = x$$

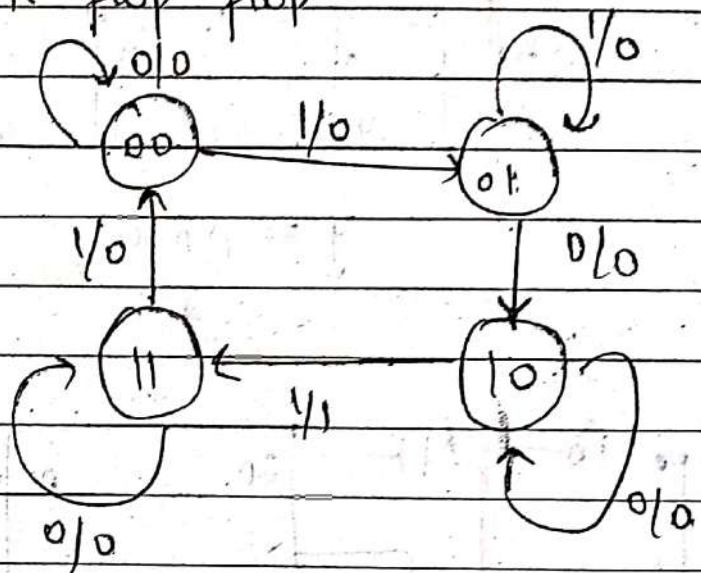
$$T_B = Q_A x$$

$$y = Q_A Q_B$$

Present State		I/P	Next State		O/P	T	
A	B	x	A	B	y	T_A	T_B
0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0
0	1	0	0	1	0	0	0
0	1	1	1	1	0	1	0
1	0	0	1	0	0	0	0
1	0	1	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	0	1	1	1



Ques. Design clocked sequential circuit for the given state diagram using J-K flop flop



Present State	I/p	Next state	O/p	J _A	K _A	J _B	K _B
A	B	A	B				
0	0	0	0	0	x	0	x
0	0	0	1	0	x	1	x
0	1	1	0	0	x	x	0
0	1	0	1	1	x	x	1
1	0	1	0	x	0	0	x
1	0	1	1	x	0	1	x
1	1	1	0	x	0	x	0
1	1	0	0	x	1	x	1

	$\bar{B}\bar{X}$	$\bar{B}X$	BX	$B\bar{X}$
$\bar{A}$	0	1	3	2
A	X ₄	X ₅	X ₇	X ₆

$$J_A = B\bar{X}$$

	$B\bar{X}$	$\bar{B}X$	BX	$B\bar{X}$
$\bar{A}$	X ₀	X ₁	X ₃	X ₂
A	X ₄	X ₅	X ₇	X ₆

$$K_A = B\bar{X}$$

	$\bar{B}\bar{X}$	$\bar{B}X$	BX	$B\bar{X}$
$\bar{A}$	0	1	X ₃	X ₂
A	X ₄	1	X ₇	X ₆

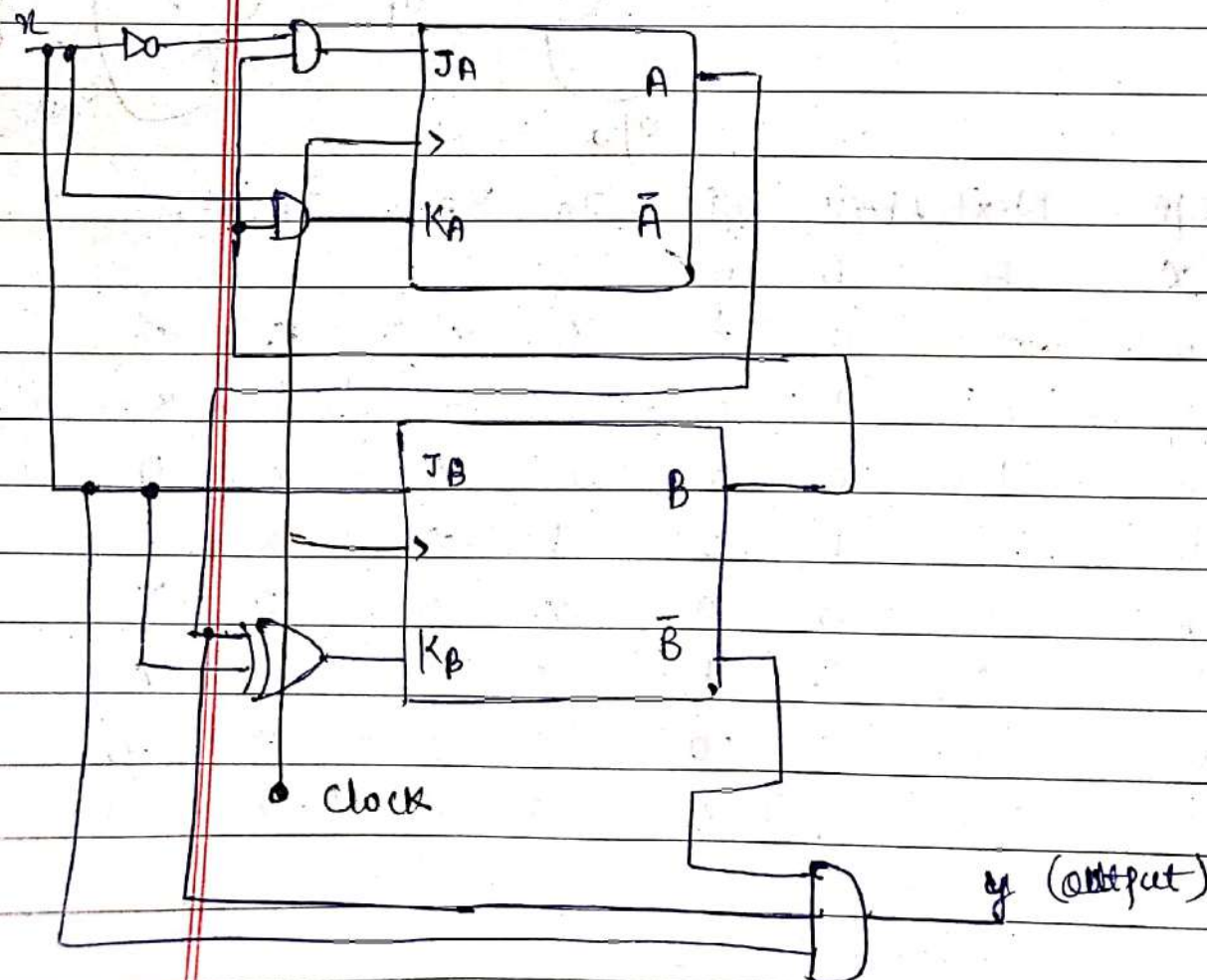
$$J_B = X$$

	$\bar{B}\bar{X}$	$\bar{B}X$	BX	$B\bar{X}$
$\bar{A}$	X ₀	X ₁	X ₃	1 ₂
A	X ₄	X ₅	1 ₇	X ₆

$$K_B = \bar{A}\bar{X} + AX$$

	$\bar{B}\bar{X}$	$\bar{B}X$	BX	$B\bar{X}$
$\bar{A}$	0	1	3	2
A	4	1	7	6

$$Y = ABX$$



Registers:-

It is a group of flip flop used to store binary information.

A "n" bit register has a group of n-flip flop and is capable of storing n-bit binary information.

→ On the basis of applications registers are of two types ① Shift registers ② Storage registers.

Shift Register:-

A register capable of shifting binary information either to the right or to the left is called shift register.

There are two methods for shifting a data:

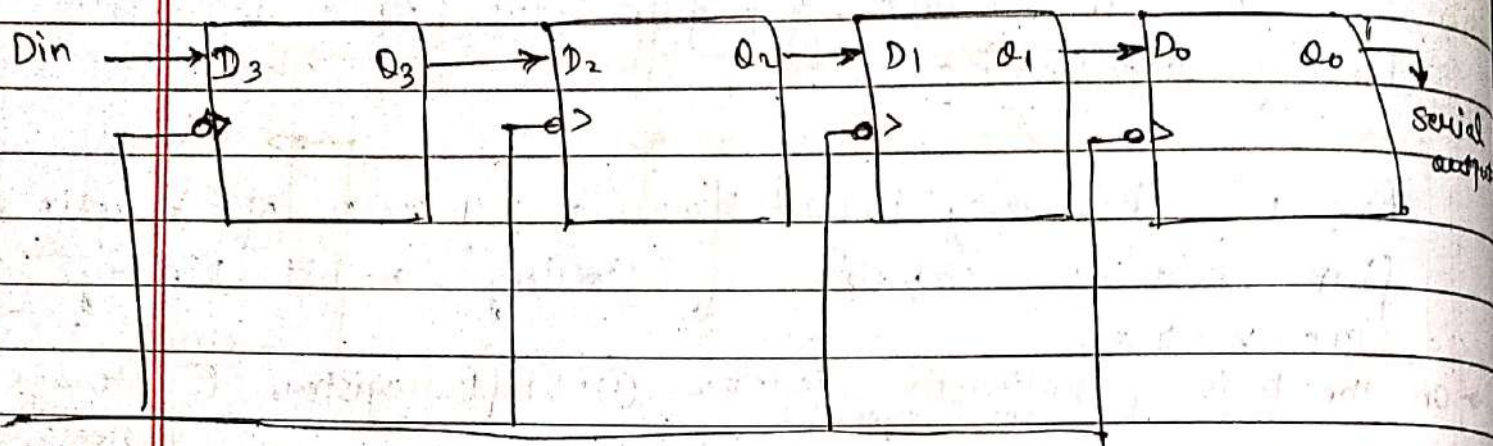
- ① Serial shifting :- In serial shifting, 1 bit shift at a time for each clock pulse in a serial fashion starting with either M.S.B or L.S.B.
- ② Parallel shifting :- In parallel shifting, all data get shifted simultaneously at a single clock pulse.

Shift registers are classified into the following four types based on how binary information is entered or shifted out.

- ① serial In - serial out
- ② serial In - parallel out
- ③ parallel In - parallel out
- ④ parallel In - serial out

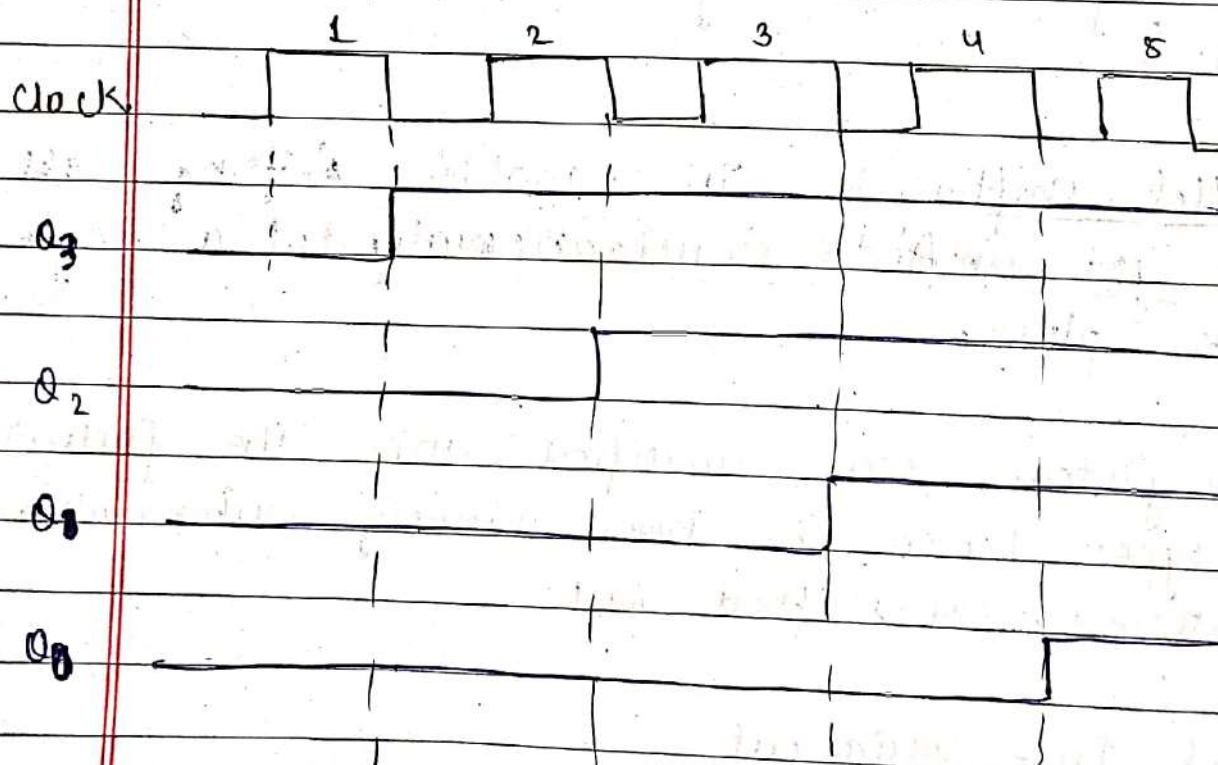
① Serial In Serial Out

' 1111 '



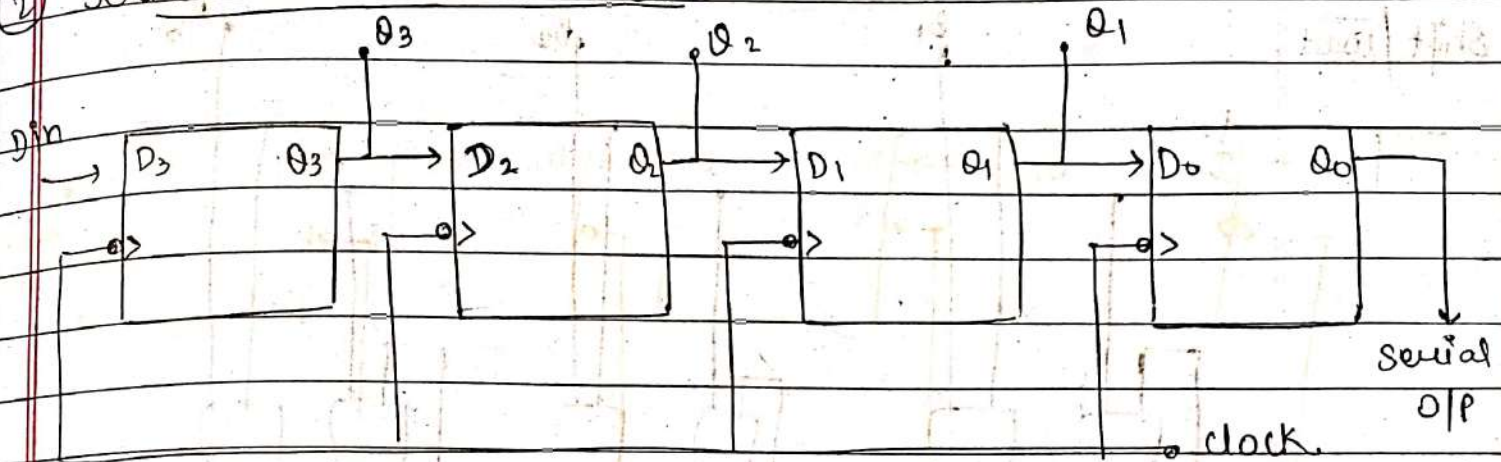
clock.

CLK	Q_3	Q_2	Q_1	Q_0	CLK	D	Q _{next}
Initially	0	0	0	0	1	0	0
↓	1	0	0	0	1	1	1
↓	1	1	0	0			
↓	1	1	1	0			
↓	0	1	1	1			

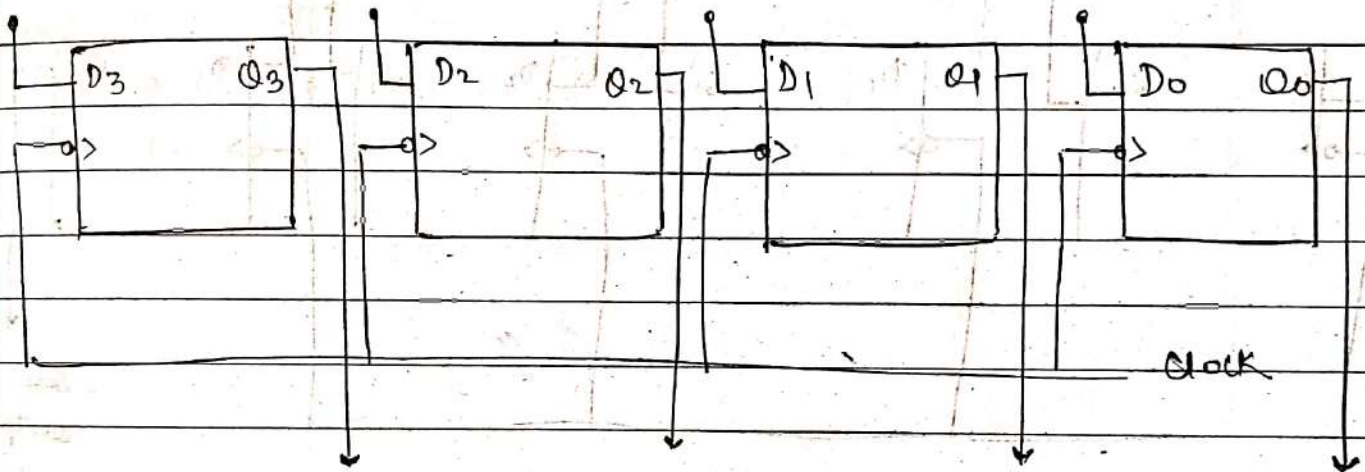


Page no. Date

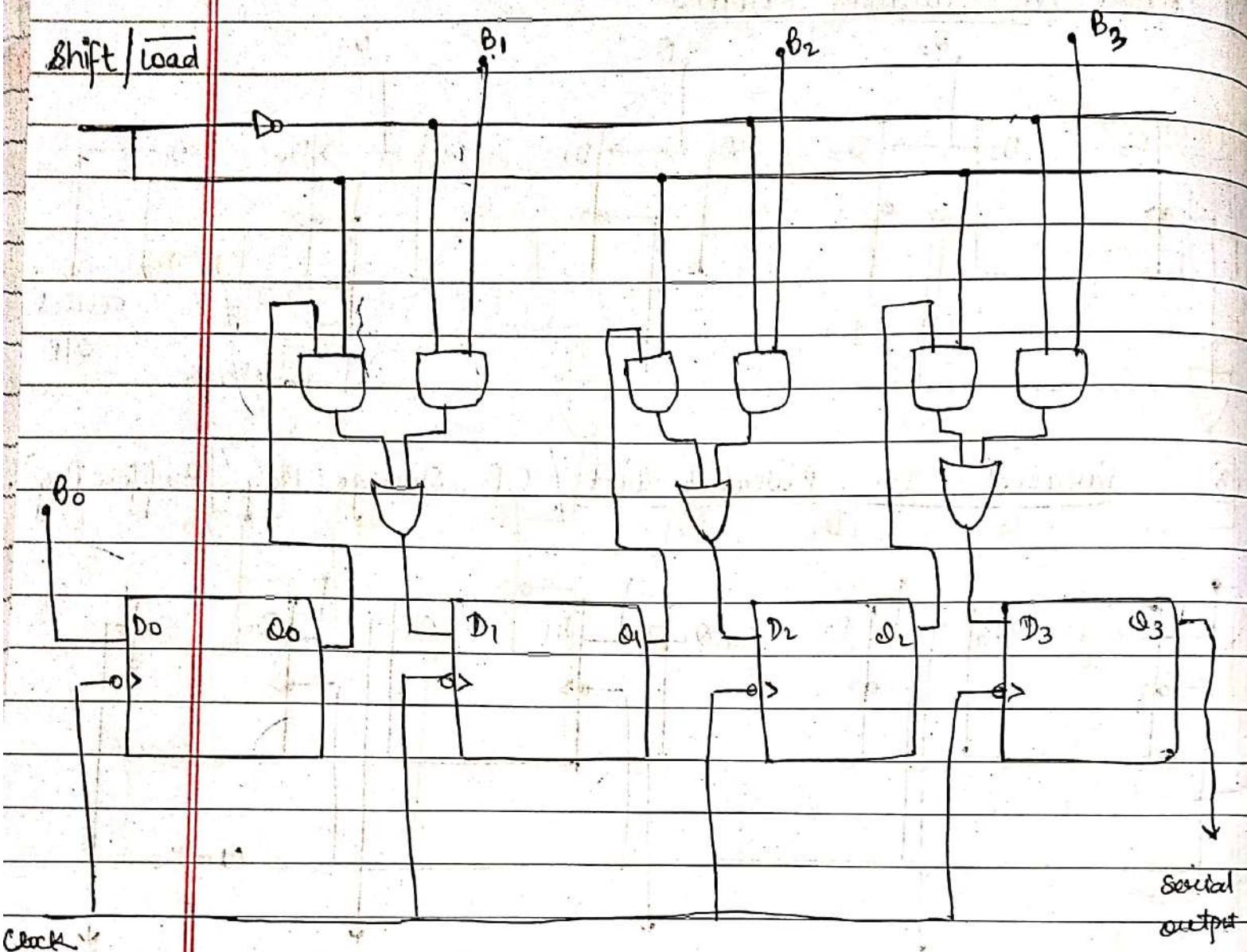
② Serial In Parallel Out :-



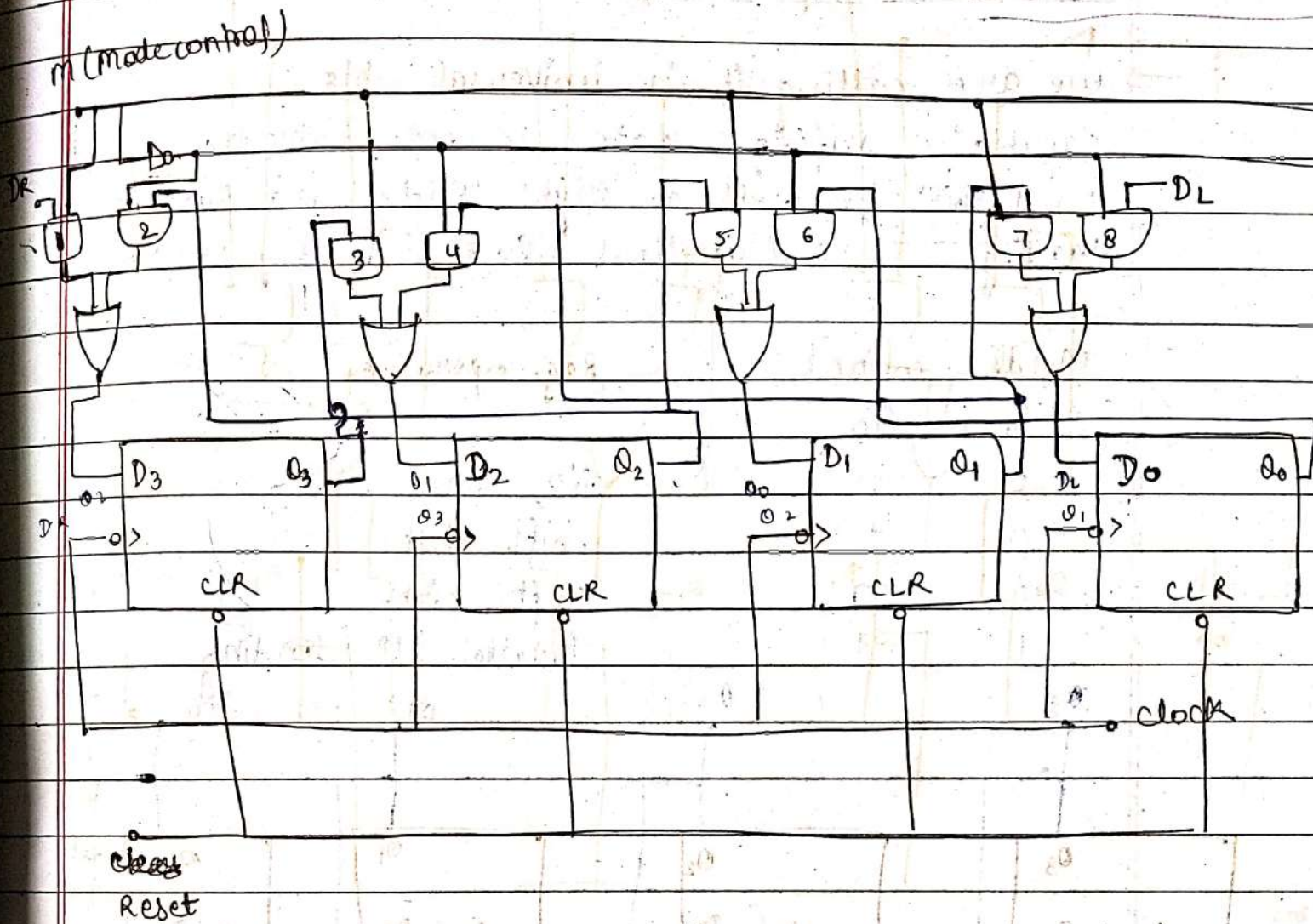
③ Parallel In Parallel Out / OR Storage Reg / Buffer Reg.



→ Here we require single clock. to store the data.
so, storage of data is very fast



Bidirectional Shift Register (b/c it shift data in both dirⁿ)



when $M = 1$ shift right
 $M = 0$ shift left

Universal Shift Register

→ we are calling it is universal b/c it is capable of shifting data in both direction (left shift as well as right shift) and parallel loading [i.e. Bidirectional + Parallel loading = U-S-R]

Mode control

S_1 S_0

0 0

0 1

1 0

1 1

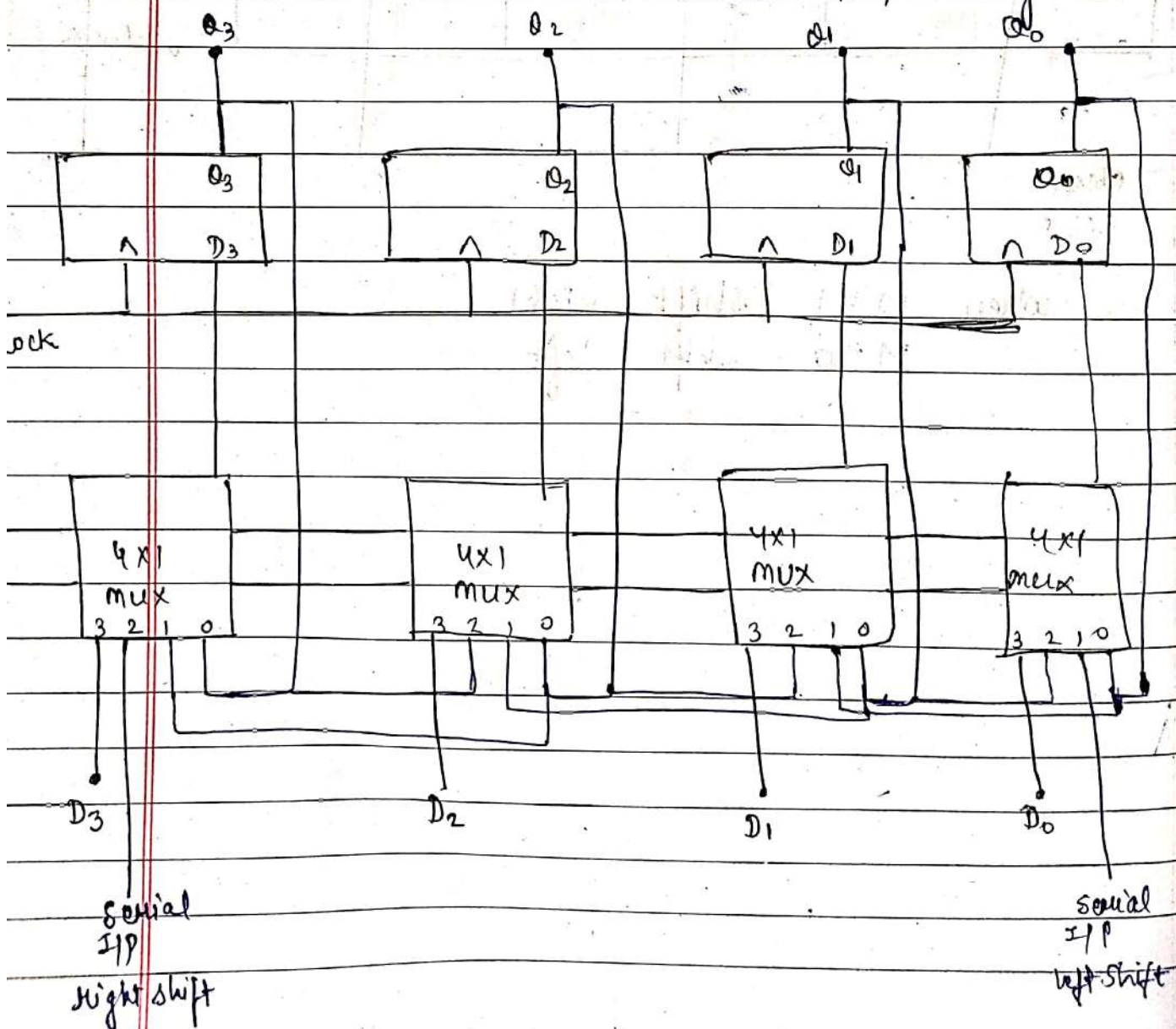
Reg. operations

No change --

Shift Left

Shift Right

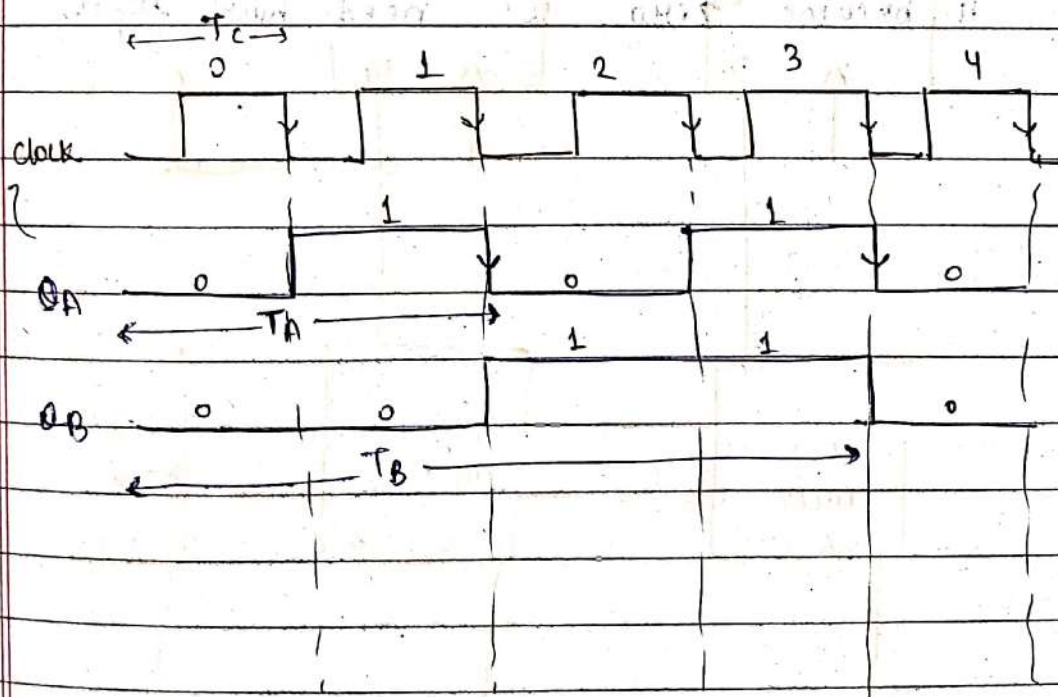
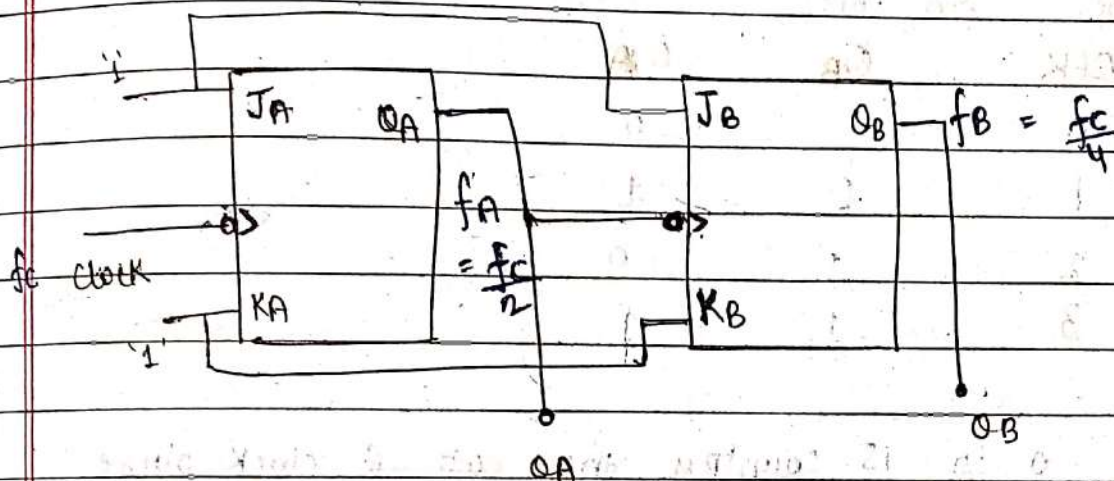
Parallel I/P / loading



Counters:

It is sequential circuit consisting of a set of flip flop connected in a suitable manner to count the sequence of the input pulses present to it in a digital form.

flip flop as divide by 2 circuit :-



$$T_B = 2T_A$$

$$T_A = 2T_c$$

$$\frac{1}{f_B} = \frac{2}{f_A}$$

$$\frac{1}{f_A} = \frac{2}{f_c}$$

$$f_A = \frac{f_c}{2}$$

that's why
we are
called
divide by 2
circuit)

$$f_B = \frac{f_c}{4}$$

If we are having "P" no. of flip flop and having value of J and K as 1 and it is negative edge triggered then we divide by 2^P .

i.e $P=2 \quad 2^2 = 4$
 $P=4 \quad 2^4 = 16$

frequency divide by 4.
 frequency " " 16.

counter	0 to 3 :-	(ms)	(μ s)
	CLK	Q _B	Q _A
	0	0	0
	1	0	1
	2	1	0
	3	1	1

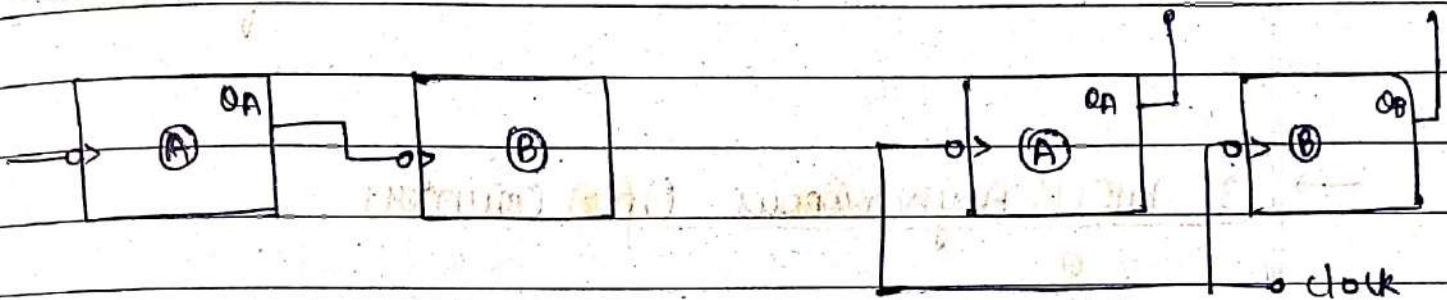
if 0 to 15 counter then at 16 clock pulse it become zero we need four flip flop

Types of Counter

- ① Asynchronous counters / Ripple
- ② Synchronous counters

Asynchronous counters

Synchronous counters



- | | |
|---|---|
| <p>① flip flop are connected in such a way that the op of first flop are not drives the clock of next flip flop</p> <p>② flip flops are not clocked simultaneously</p> <p>③ Circuit is simple for more number of states</p> <p>④ Speed is slow as clock is propagated through no. of stages</p> | <p>There is no connection between o/p of first flip flop and clock of next flip flop.</p> <p>flip flops are clocked simultaneously.</p> <p>Circuit becomes complicated as no. of states increases</p> <p>Speed is high as clock is given at a same time</p> |
|---|---|

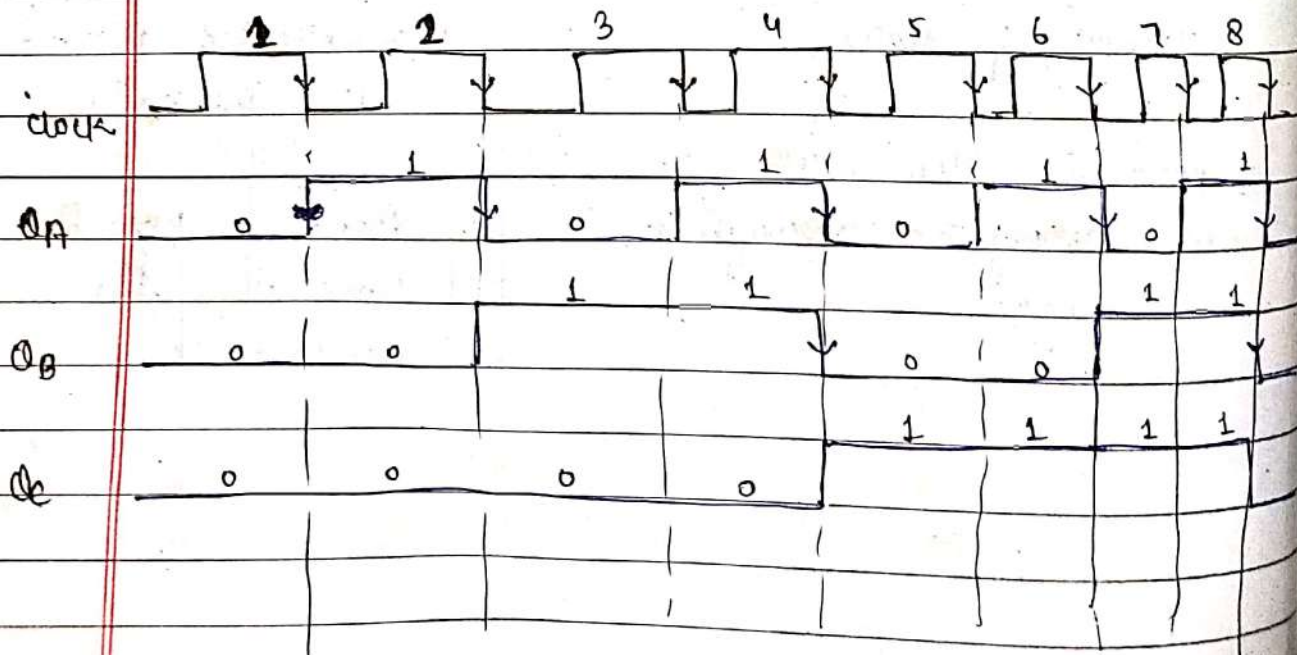
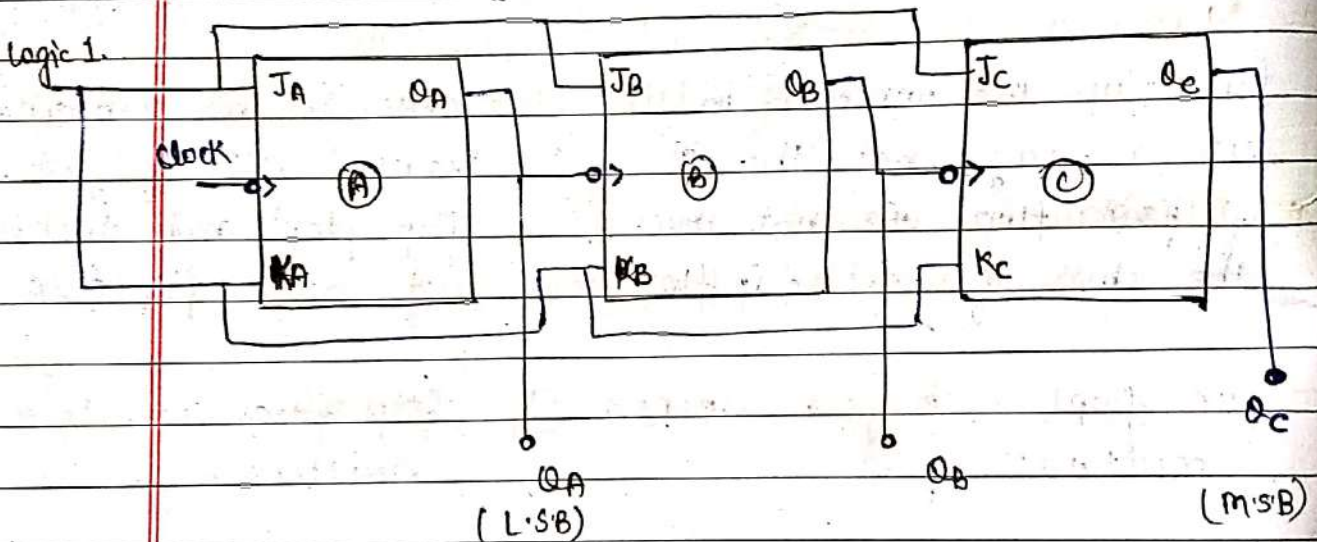
Counters (AS/S)

~~The~~ also.

Counters are classified into three categories

- ① Up counters (1, 2, ..., 7)
- ② Down counters (7, ..., 0)
- ③ Up-down counters (combination of these two)

→ 3 bit Asynchronous Up Counter



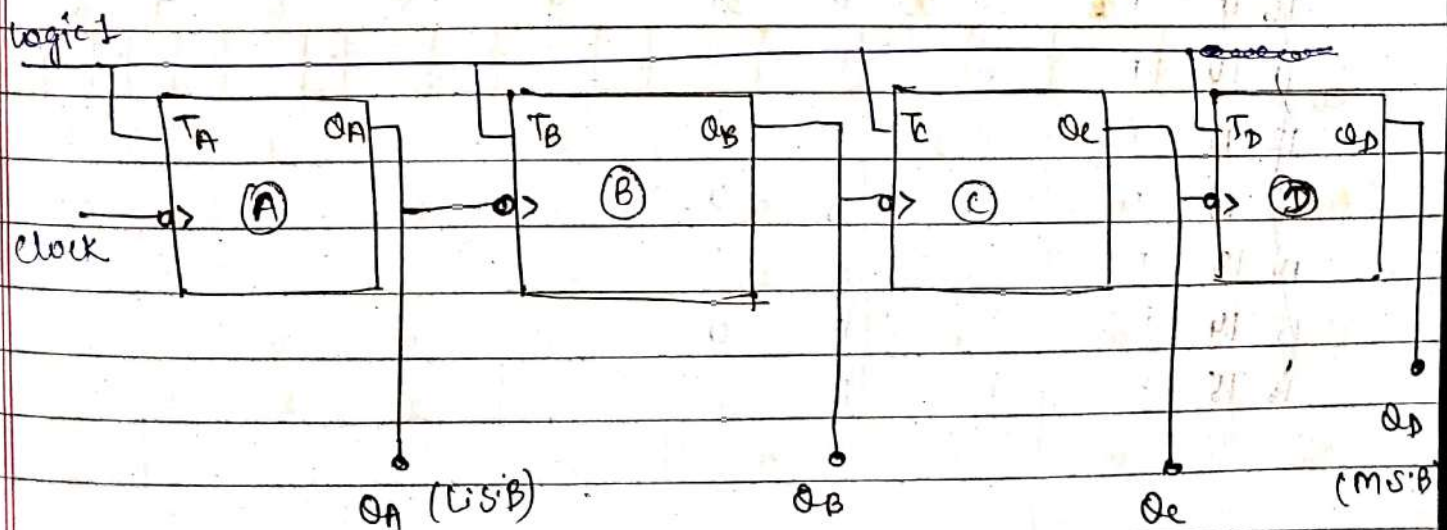
clock		Q_A	Q_B	Q_C	Decimal eq ⁿ
1	Initially	0	0	0	0
2	1	0	0	1	1
3	2	0	1	0	2
4	3	0	1	1	3
5	4	1	0	0	4
6	5	1	0	1	5
7	6	1	1	0	6
8	7	1	1	1	7
9	8	0	0	0	0

Number of States $2^n = 2^3 = 8$.
 where n of flip flop, or no. of bits

Maximum count = $2^n - 1 = 8 - 1 = 7$.
 n = no. of flip flop used, or no. of bits

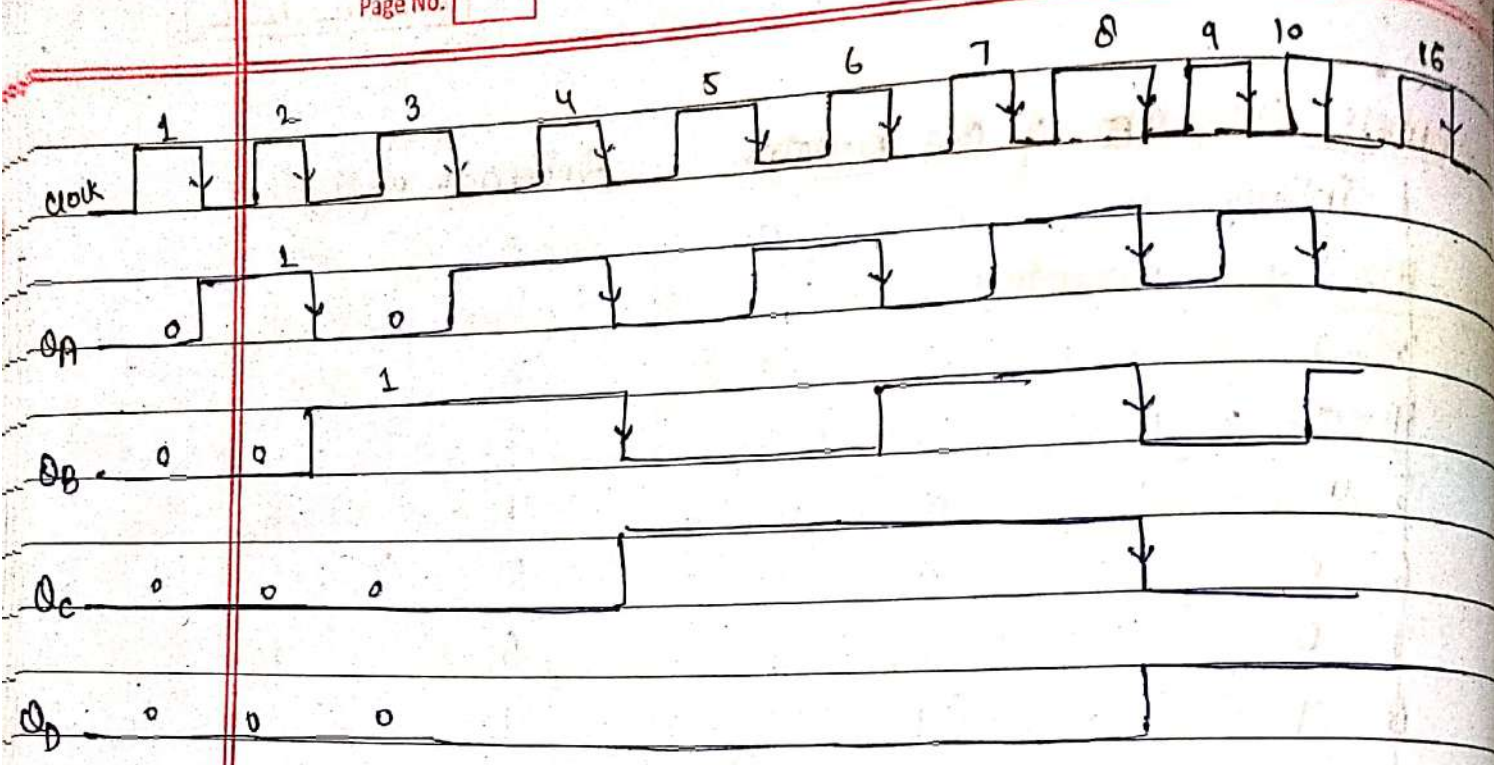
→ 4-bit Asynchronous Up Counters

→ Now as we only want to toggle the input
 So, we can use T flip flop also.



No. of States $2^4 = 16$

Maximum count → $2^4 - 1 = \underline{15}$.



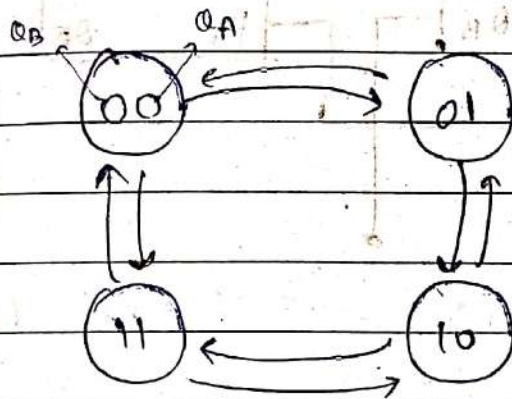
Clock	QA	QB	QC	QD
Initially	0	0	0	0
1	1	0	0	0
2	0	0	0	0
3	1	0	0	0
4	0	1	0	0
5	1	1	0	0
6	0	0	1	0
7	1	1	1	0
8	0	0	0	1
9	1	1	1	1
10	0	0	0	0
11	1	1	1	1
12	0	0	0	0
13	1	1	1	1
14	0	0	0	0
15	1	1	1	1
16	0	0	0	0

State Diagram of a Counter

2-bit Up Counter

Maximum count = $2^2 - 1 = 3$ → 0 to 3

No. of states = $2^2 = 4$

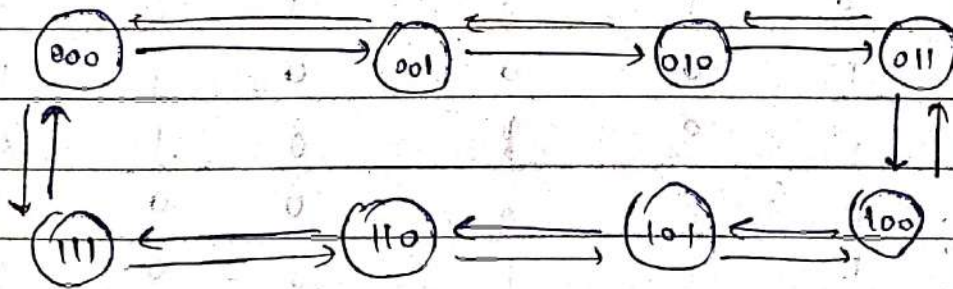


Q_B	Q_A
0	0
0	1
1	0
1	1

3-bit Up Counter

Maximum count = $2^3 - 1 = 7$

No. of states = 8

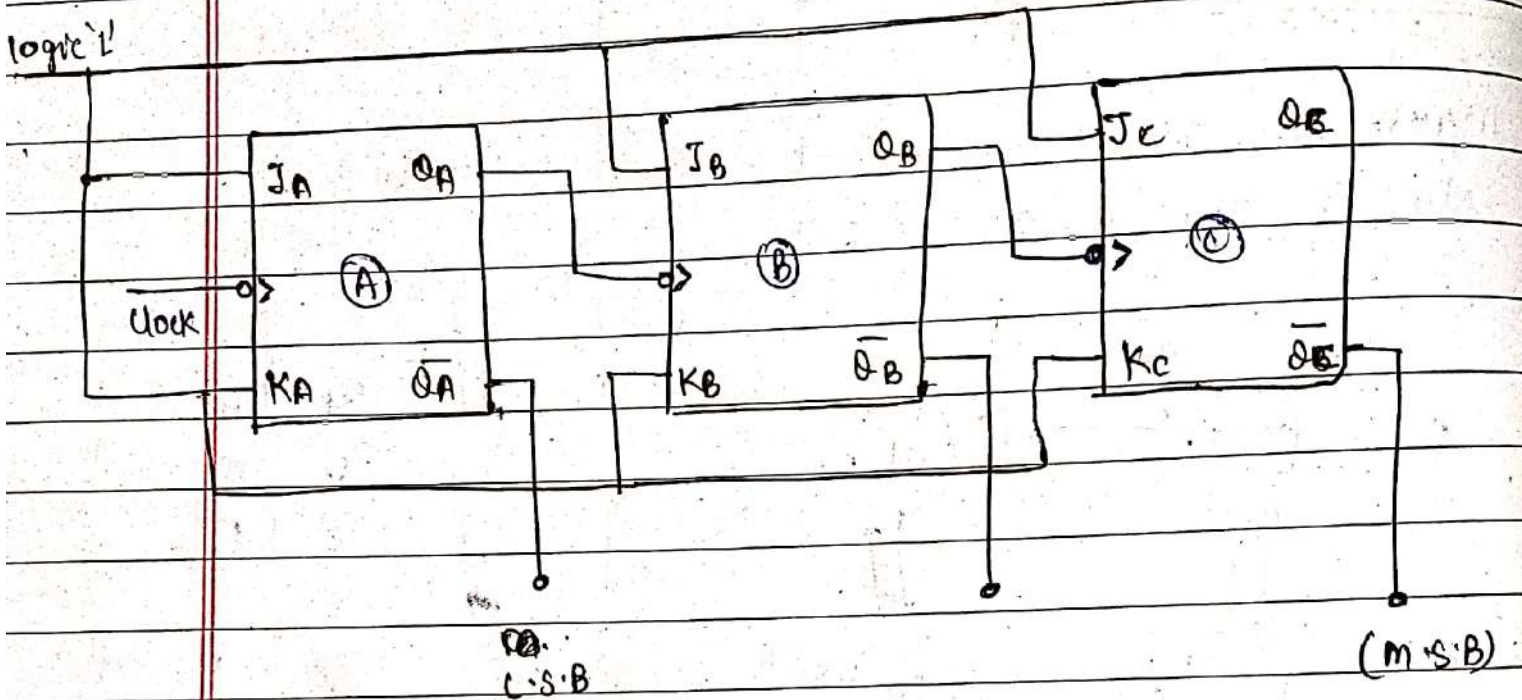


Q_C	Q_B	Q_A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

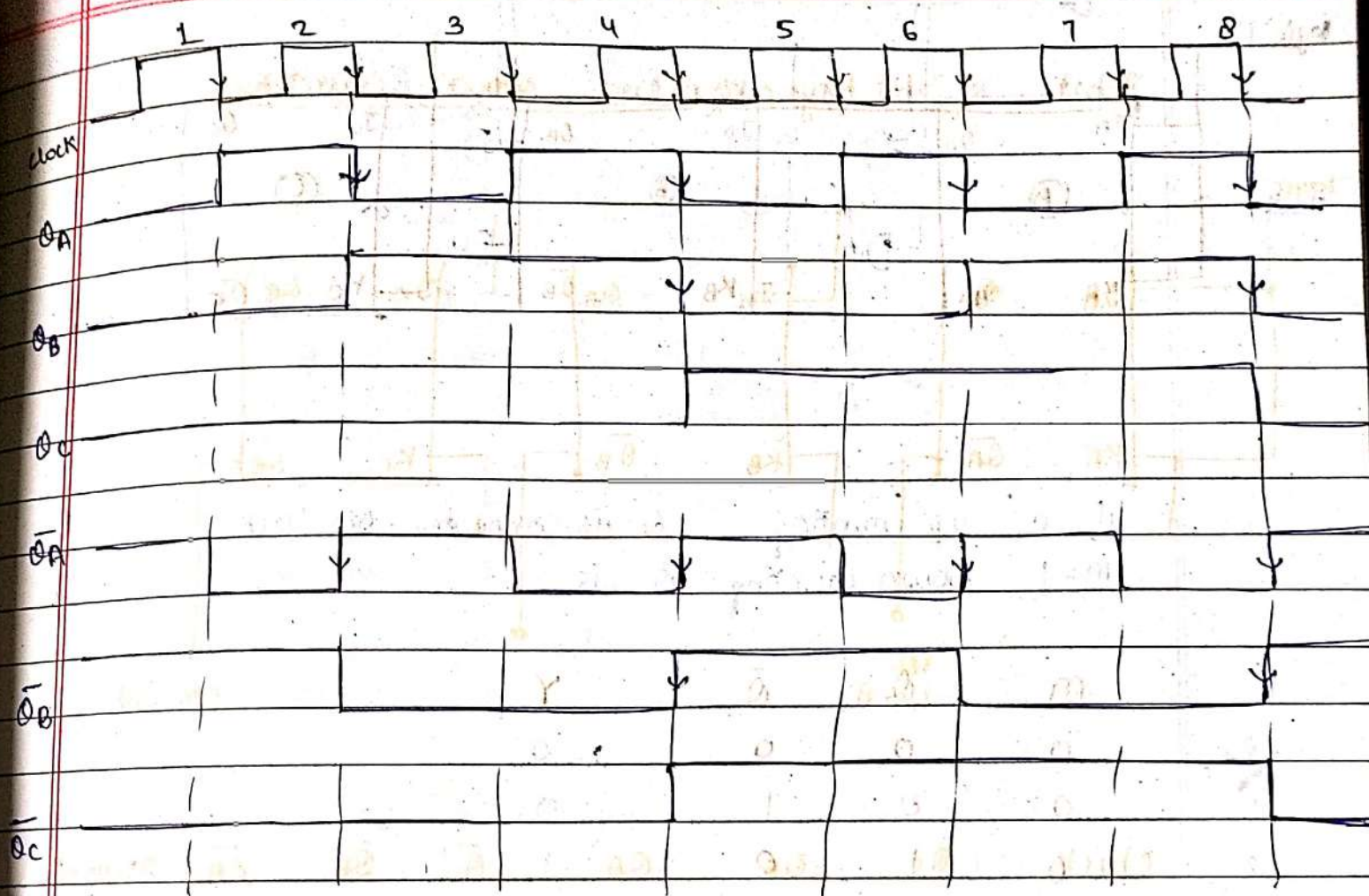
• → up counter

• → down counter

3 bit & 4 bit Asynchronous Down Counter



Clock	QC	QB	QA	$\bar{Q}_C$	$\bar{Q}_B$	$\bar{Q}_A$	Decimal
Initially	0	0	0	1	1	1	7
1	0	0	1	1	1	0	6
2	0	1	0	1	0	1	5
3	0	1	1	1	0	0	4
4	1	0	0	0	1	1	3
5	1	0	1	0	1	0	2
6	1	1	0	0	0	1	1
7	1	1	1	0	0	0	0



3-bit 4-bit UP/down Ripple counter.

Here we use mode control to select either we want up counting or down counting.

- This combinational circuit is
- This sequential circuit is capable to ^{count} store the data (either up or down)

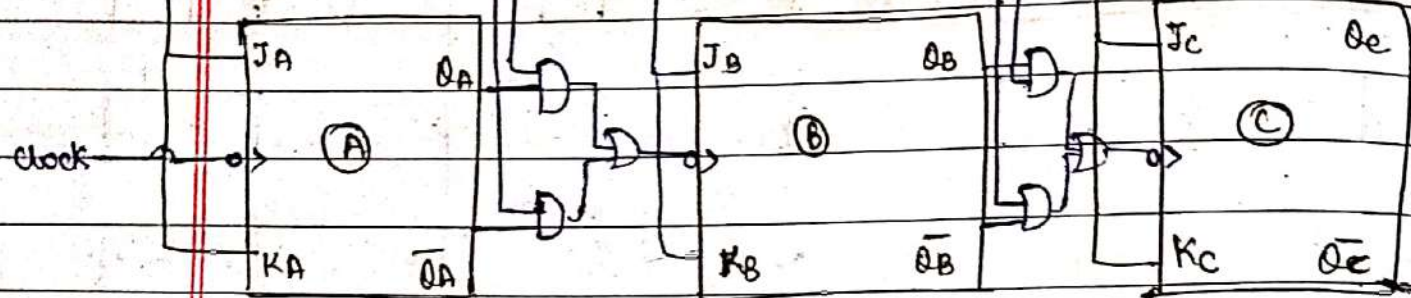
(mode control)

M

Page No.

Date

logic '1'



M=0 up counting Q is connected to clock

M=1 Down counting $\bar{Q}$ is " " " "

	M	Q	$\bar{Q}$	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

	00	01	11	10
$\bar{M}$	0	1	1	1
M	4	5	7	6

$$Y = \bar{M}Q + M\bar{Q}$$

Counting Up to Particular Value 8-

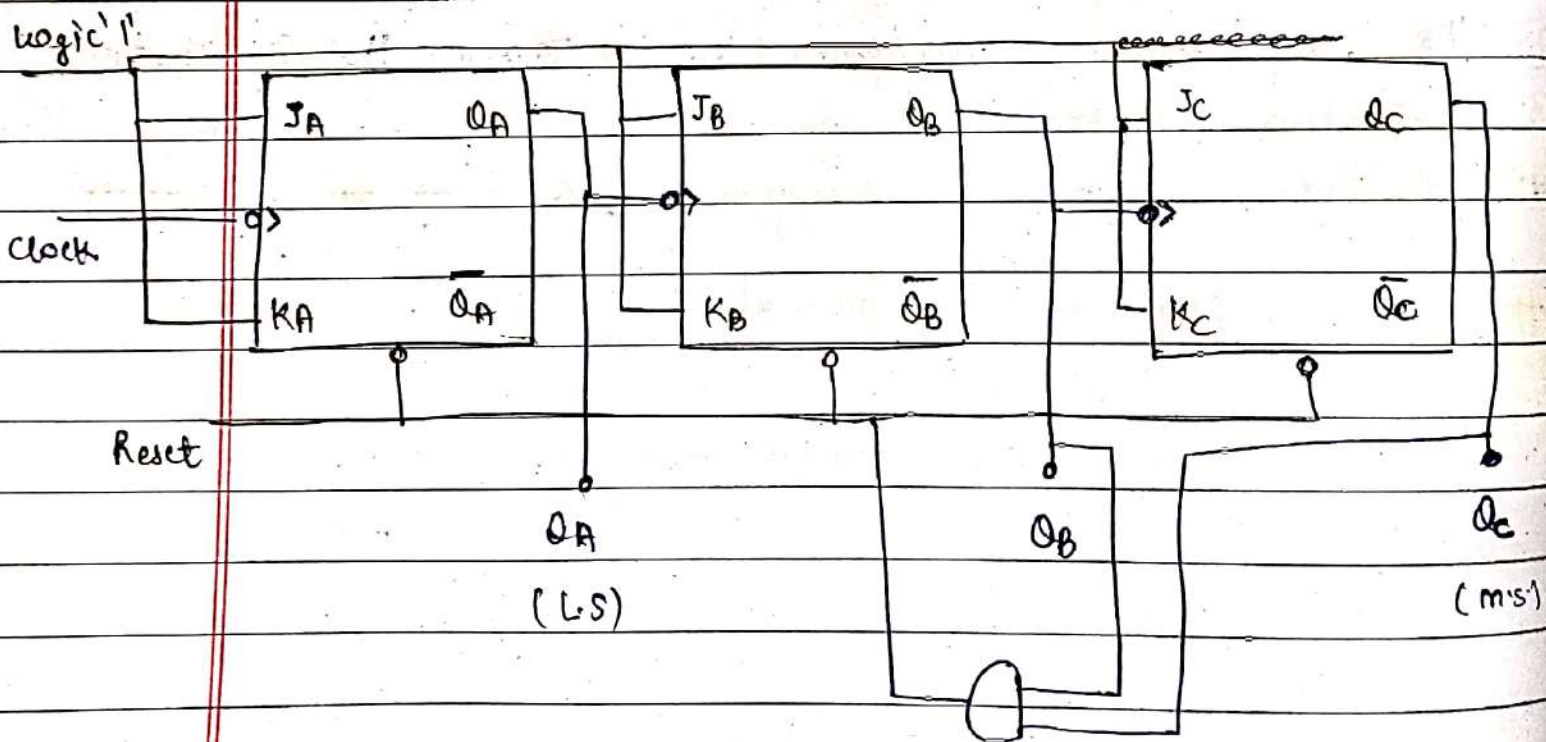
Let us take example

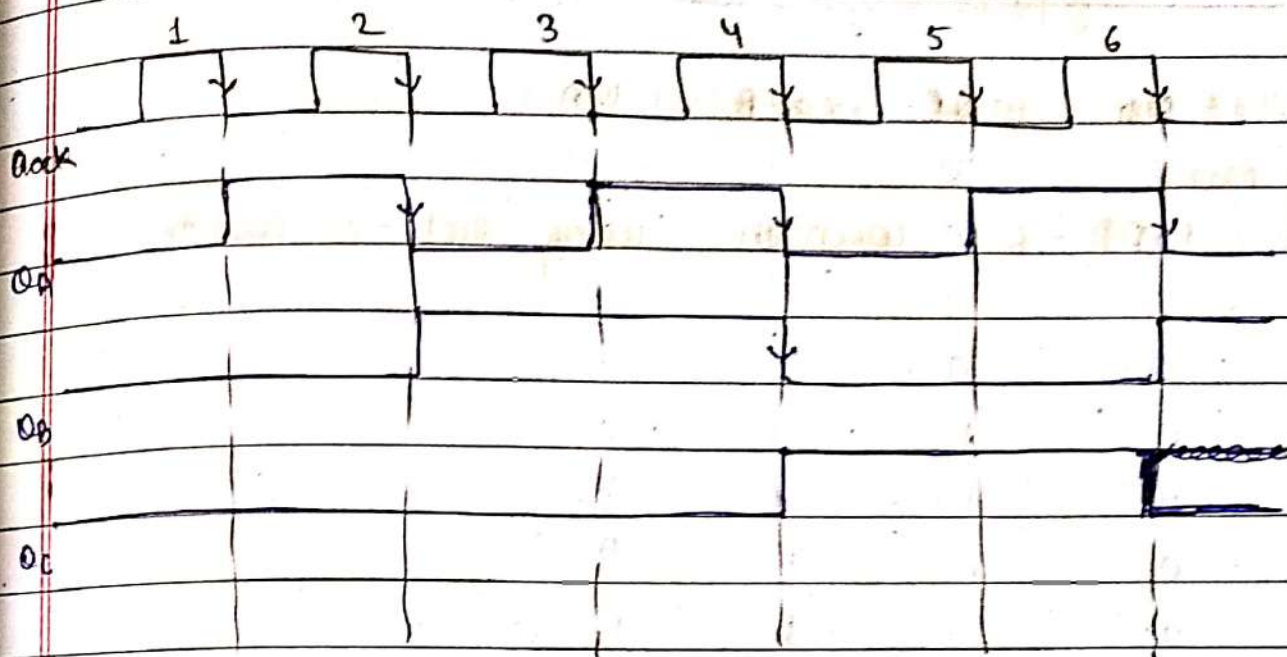
make
Ques. MOD-6 counter using MOD-8 counter.

States - 6

Maximum count = 5

Binary	Decimal eq ⁿ
$Q_C Q_B Q_A$ 0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5

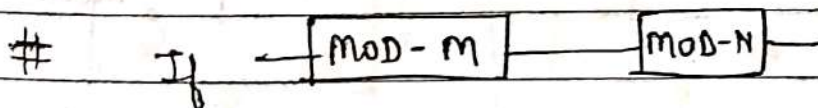




Decade (BCD) Ripple Counter

Important points

1. Negative edge Triggered $\bar{A}$ ~~clock~~ $\rightarrow$ up counter
2. Positive edge Triggered $\bar{Q}$ $\rightarrow$ up counter
3. Negative edge Triggered $\bar{Q}$ $\rightarrow$ down counter
4. Positive edge Triggered Q $\rightarrow$ down counter.



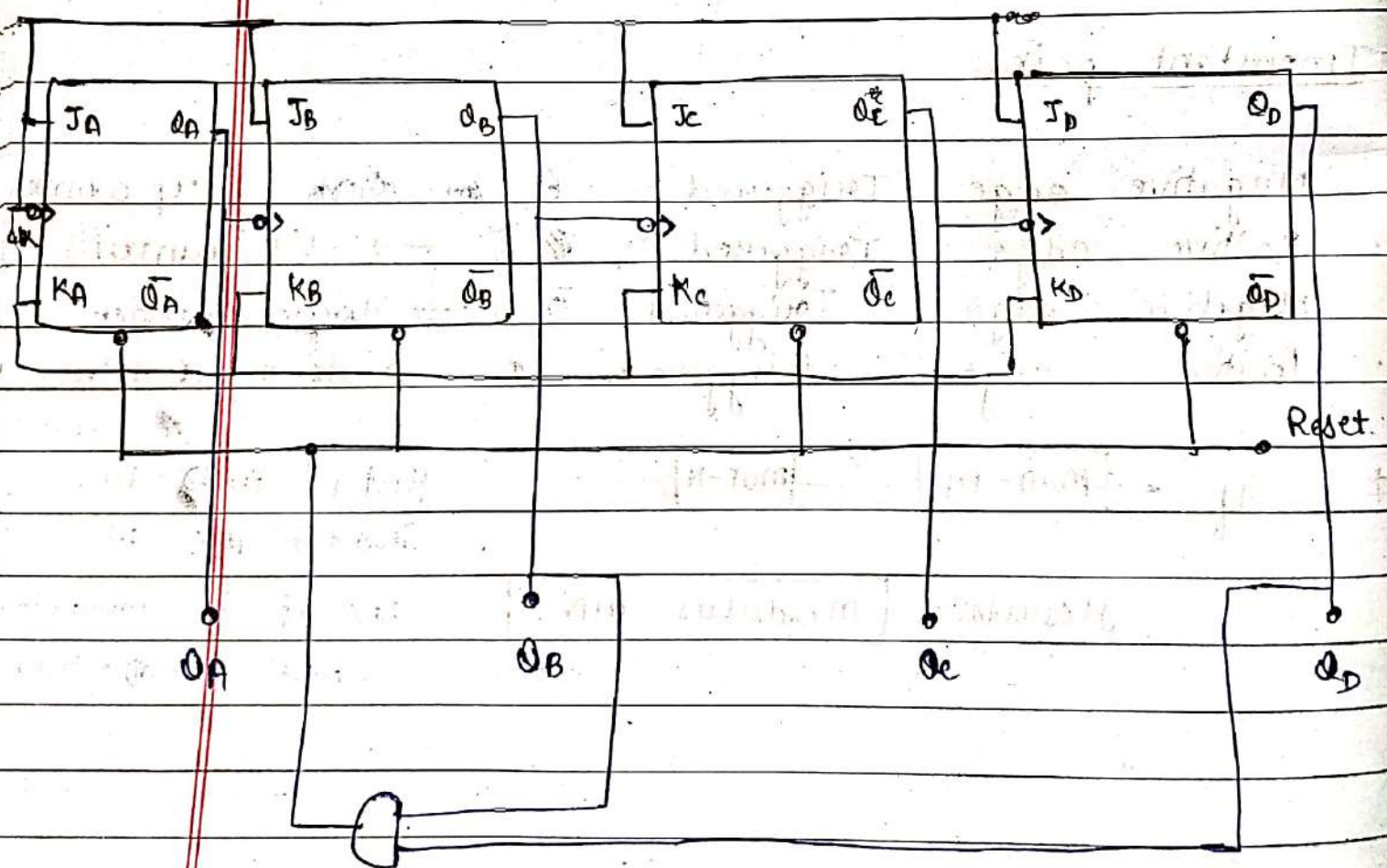
first FF MOD - M
Second FF MOD - N

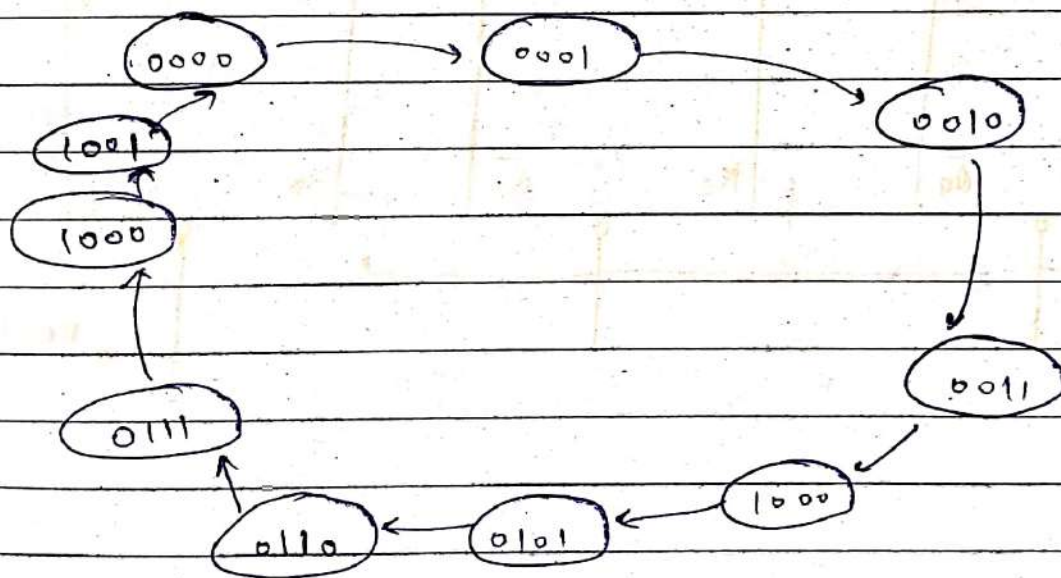
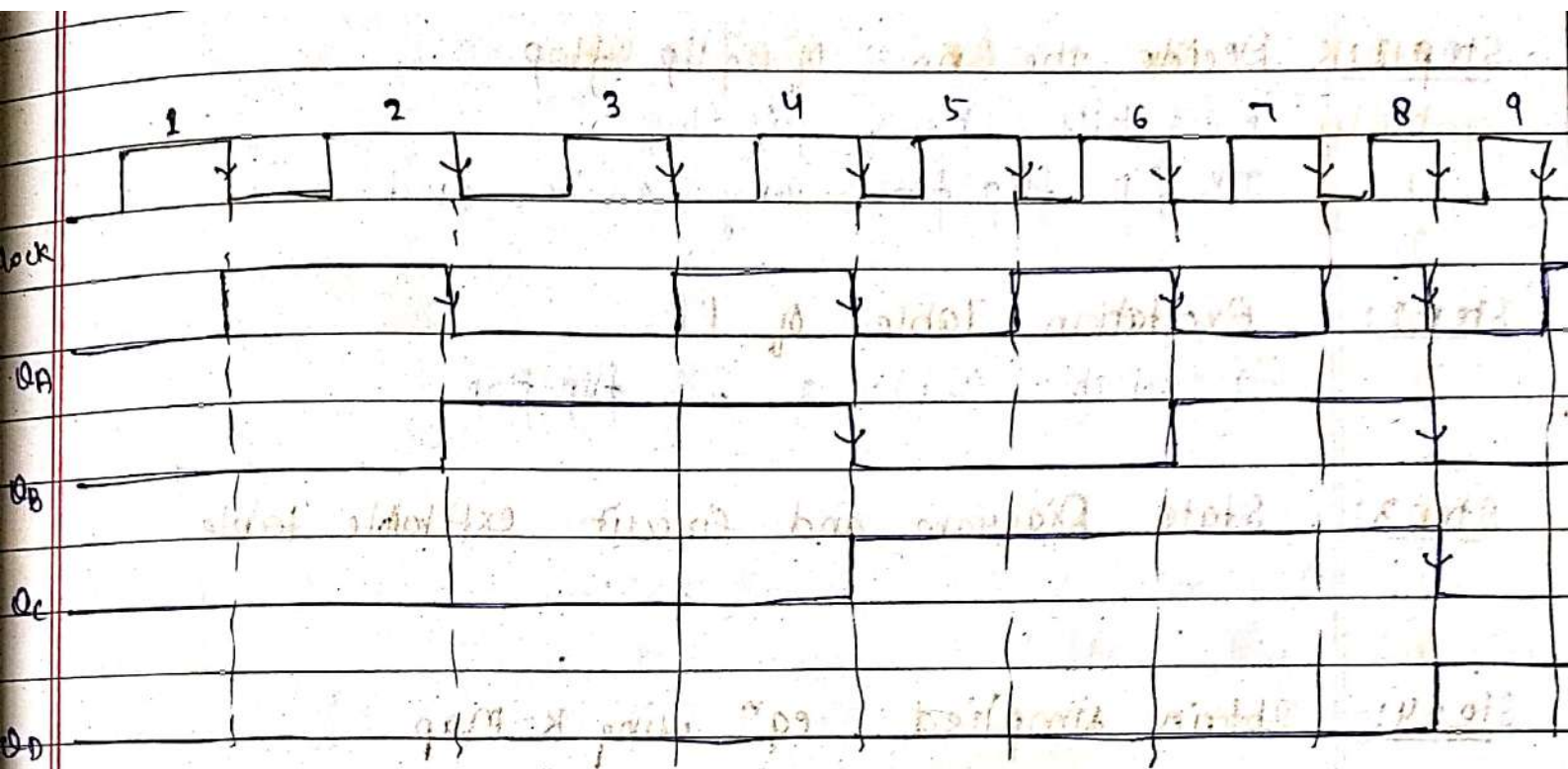
Resultant modulus MN

and if we cascade the
Resultant MOD - MN

BCD Ripple counter

clock	Q_D	Q_C	Q_B	Q_A
Initially	0	0	0	0
1	0	0	1	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	0	0	0	0
9	1	0	0	1





How to Design synchronous counters

Step 1: Decide the no. of flip flop
no. of bits = no. of flip flop
J.K / D flip flop any can be used.

Step 2: Excitation Table of FF.
→ Excitable Table of JK flip flop

Step 3: State Diagram and Circuit excitable table.

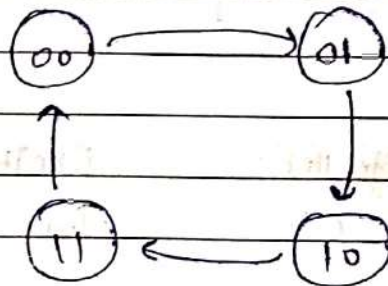
Step 4:- Obtain simplified eqⁿ using K-Map

Step 5:- Draw the logic diagram.

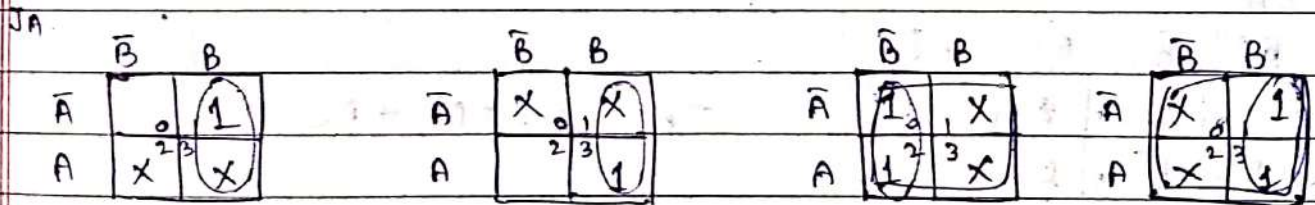
Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Ques. Design 2-bit Synchronous up counter.

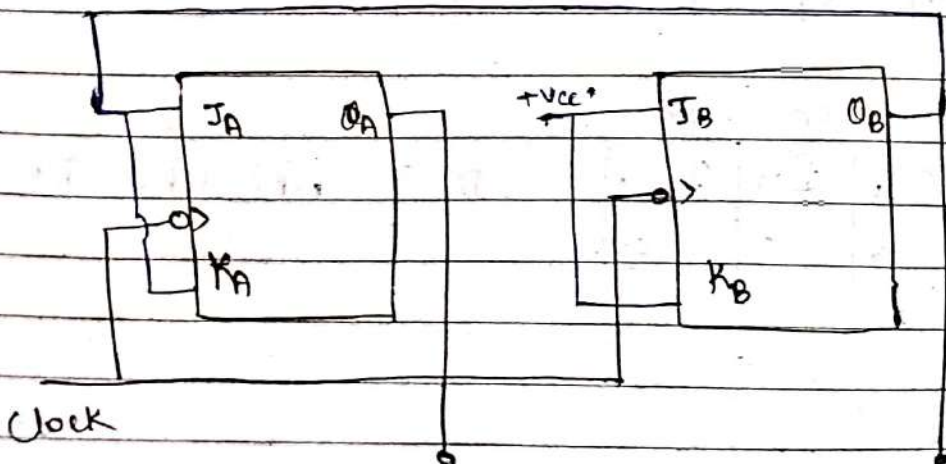
No. of states $2^n = 2^2 = 4$



Present State		Next State		Excitation I/P			
A	B	A	B	J_A	K_A	J_B	K_B
0	0	0	1	0	x	1	x
1	1	1	0	1	x	x	1
2	1	0	1	x	0	1	x
3	1	0	0	x	1	x	1



$J_A = B$ $K_A = \bar{B}$ $J_B = A$ $K_B = \bar{A}$



Ques Design a binary synchronous counters following binary sequence

0, 2, 3, 1, 5, 4, 7, 6 and repeat

	Present state			Next State			Excitation J/K		
	A	B	C	A	B	C	D _A	D _B	D _C
0	0	0	0	0	1	0	0	1	0
2	0	1	0	0	1	1	0	1	1
3	0	1	1	0	0	1	0	0	1
1	0	0	1	1	0	1	1	0	1
5	1	0	1	1	0	0	1	0	0
4	1	0	0	1	1	1	1	1	1
7	1	1	1	0	0	0	0	0	0
6	1	1	0	0	0	0			

D_A

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

$$D_A = \bar{B}C + \bar{A}\bar{B} + AC$$

D_B

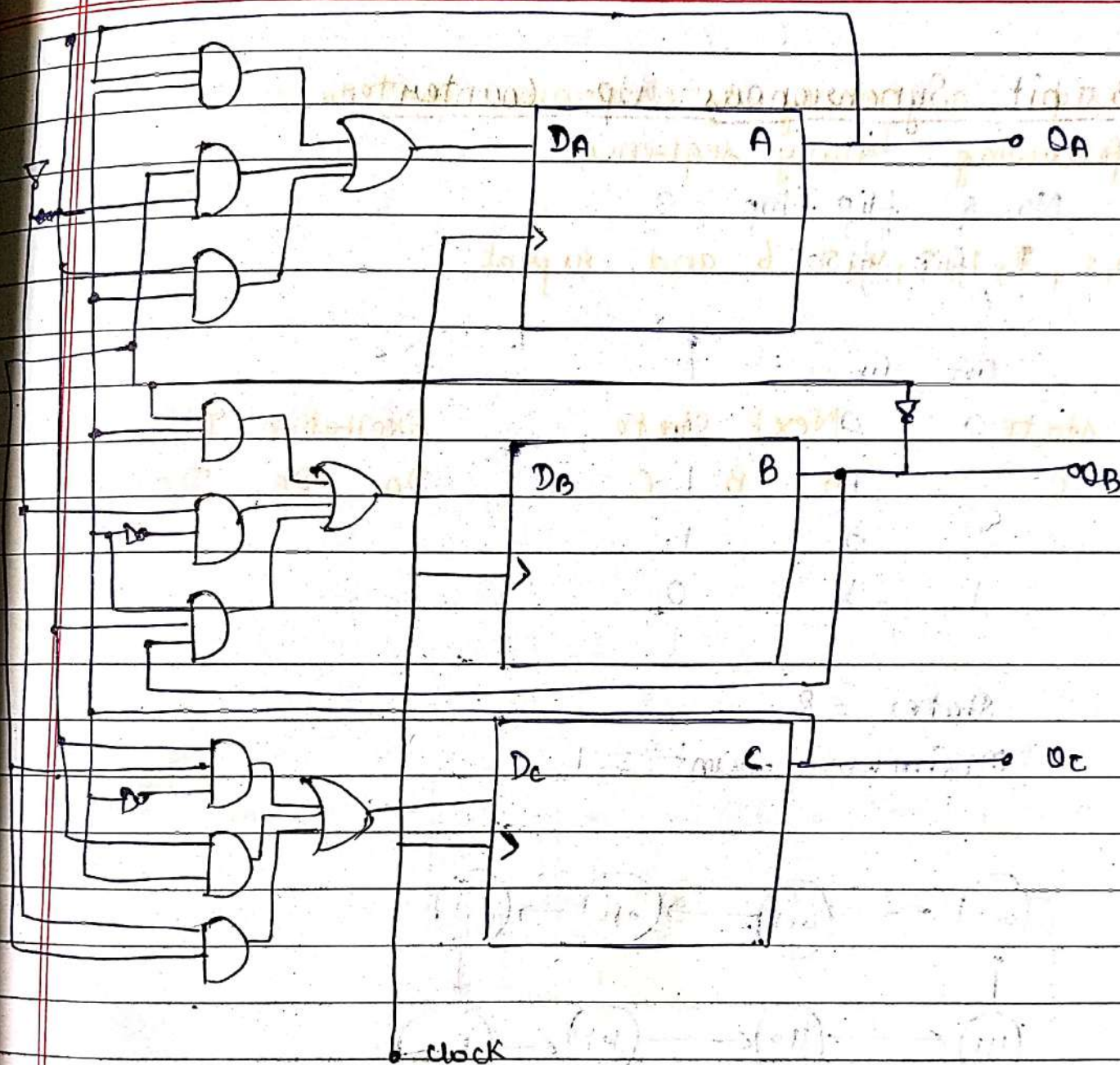
	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1	0	1	0
A	0	1	0	1

$$D_B = \bar{B}\bar{C} + \bar{A}\bar{C} + ABC$$

D_C

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

$$D_C = \bar{A}\bar{B}\bar{C} + \bar{A}C + \bar{A}B$$



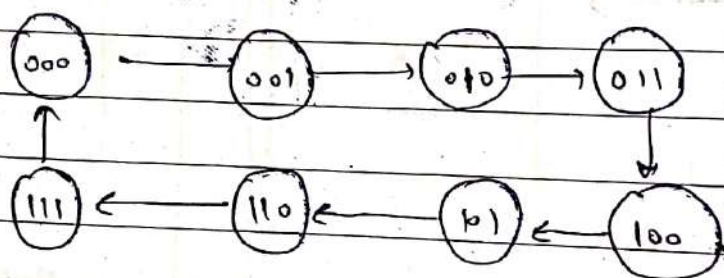
3-bit Synchronous Up Counter

Step 1: No. of flip flop = 3.
T flip flop.

Step 2:

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

Step 3: States = 8
Maximum count = 7.



	Present State			Next State			Excitation Table		
	Q_c	Q_B	Q_A	Q_c	Q_B	Q_A	T_c	T_B	T_A
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
2	0	1	0	0	1	1	0	0	1
3	0	1	1	1	0	0	1	0	1
4	1	0	0	1	0	1	0	0	1
5	1	0	1	1	1	0	0	1	1
6	1	1	0	1	1	1	0	0	1
7	1	1	1	0	0	0	1	1	1

$$T_A$$

	$\overline{Q_B}\overline{Q_A}$	$\overline{Q_B}Q_A$	$Q_B\overline{Q_A}$	Q_BQ_A
$\overline{Q_C}$	1 0	1 1	1 3	1 2
Q_C	1 4	1 5	1 7	1 6

$$T_B$$

0	1 1	1 3	0 2
4	2 5	1 7	0 6

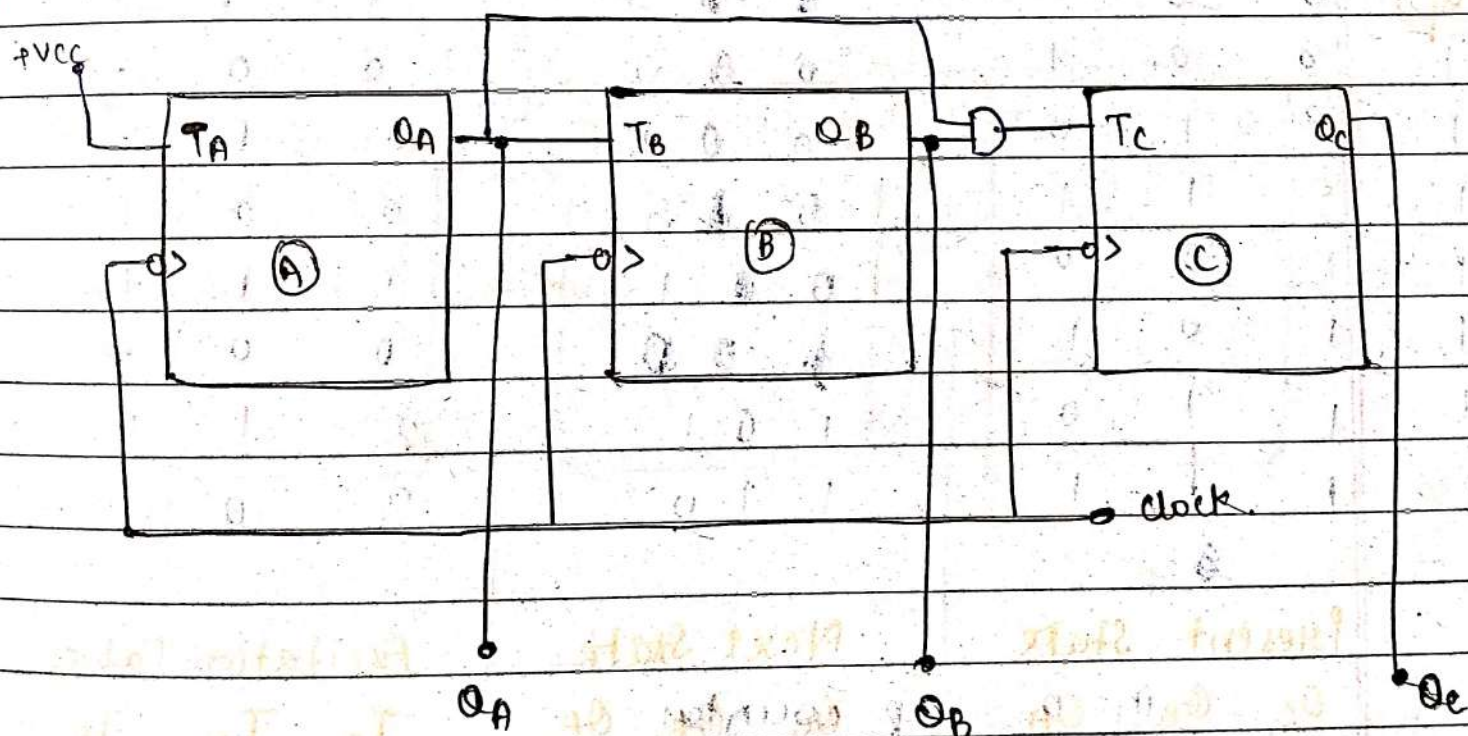
$$T_A = 1$$

$$T_B = Q_A$$

$$T_C$$

0	1	1 3	0 2
4	5	1 7	0 6

$$T_C = Q_BQ_A$$



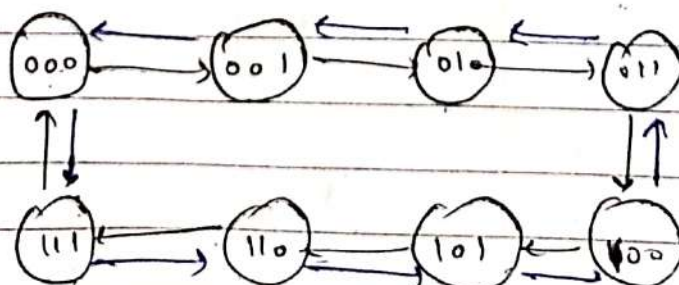
3 bit ~~mod~~ ~~up-down~~ Up Down Synchronous Counter

M	Present State			Next State			Excitation state		
	Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	T_C	T_B	T_A
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
2	0	1	0	0	1	1	0	0	1
3	0	1	1	1	0	0	1	1	1
4	1	0	0	1	0	1	0	0	1
5	1	0	1	1	1	0	0	1	1
6	1	1	0	1	1	1	0	0	1
7	1	1	1	0	0	0	1	1	1
8	0	0	0	0	0	1	1	1	1
9	0	0	1	0	0	0	0	0	1
10	0	1	0	0	0	1	0	1	1
11	0	1	1	0	1	0	0	0	1
12	1	0	0	0	1	1	1	1	1
13	1	0	1	1	0	0	0	0	1
14	1	1	0	1	0	1	0	1	1
15	1	1	1	1	1	0	0	0	1

§

M = 0 up counter

M = 1 down counter [-1 from present state]



T_C

$\bar{A}\bar{B}\bar{C}$ $\bar{A}B\bar{C}$ $A\bar{B}\bar{C}$ $AB\bar{C}$

$\bar{A}\bar{B}\bar{C}$	0	1	2
$\bar{A}B\bar{C}$	4	5	6
$A\bar{B}\bar{C}$	12	13	14
$AB\bar{C}$	8	9	10

T_B

0	1	2
4	5	6
12	13	14
8	9	10

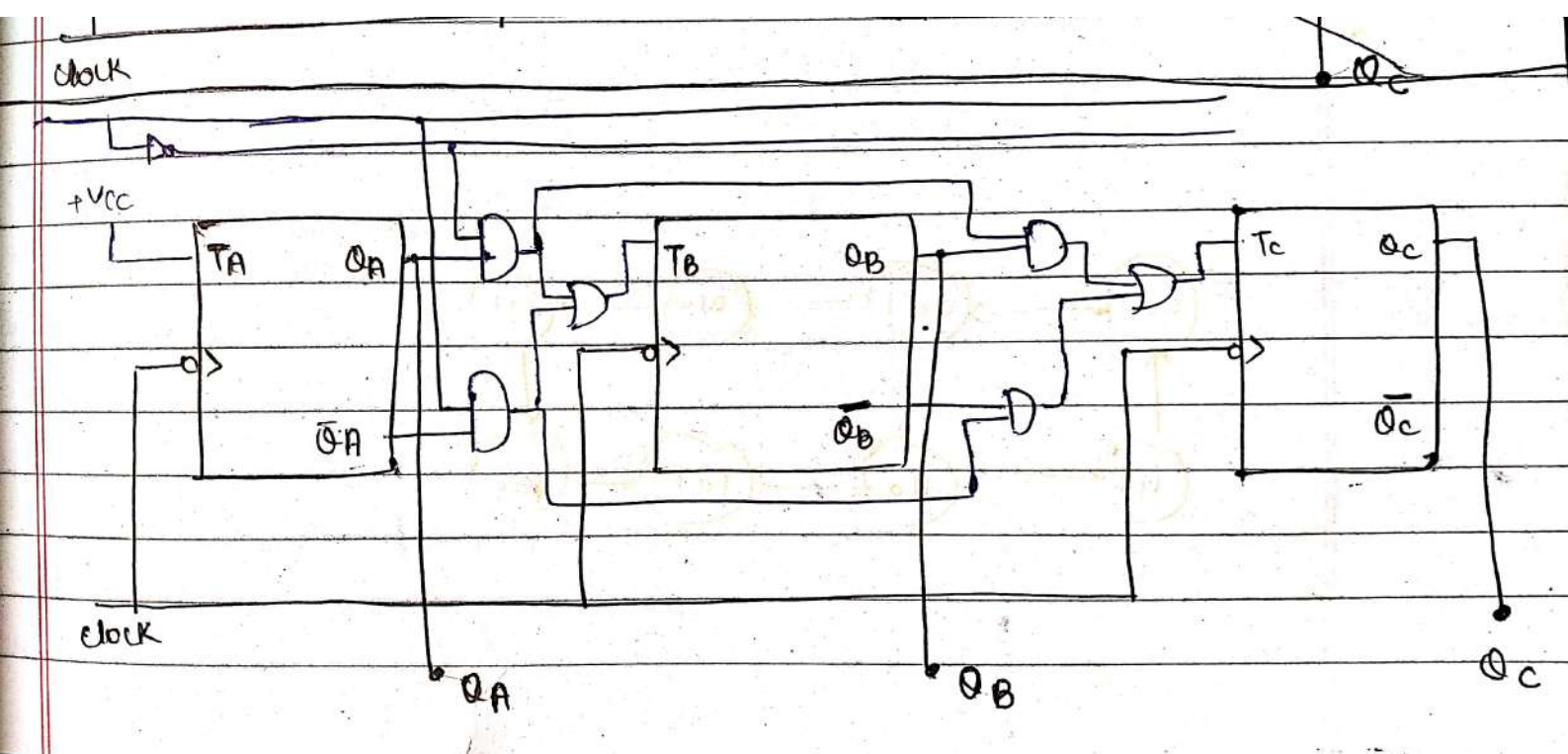
T_A

1	2	3
4	5	6
12	13	14
8	9	10

$$T_C = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + AB\bar{C}$$

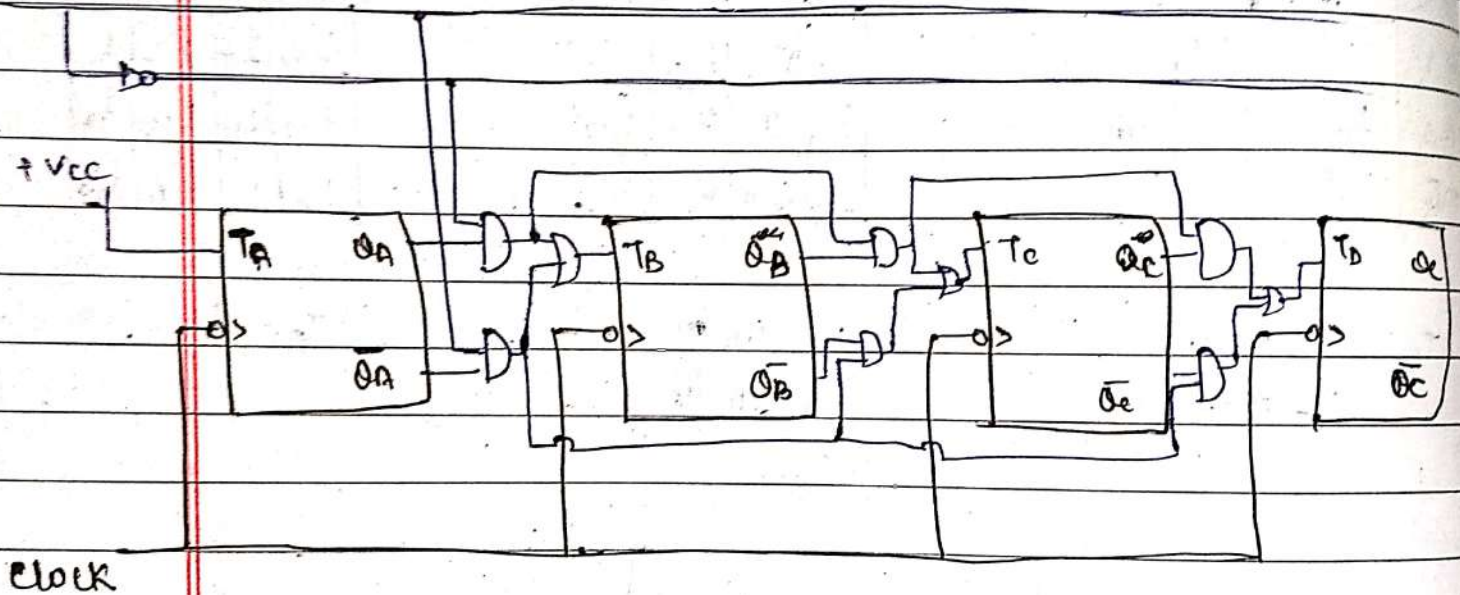
$$T_B = \bar{A}\bar{B} + \bar{A}B + A\bar{B} + AB$$

$$T_A = 1$$



4-Bit Up/Down Counter

Circuit:



PR = 0 Q = 1

CLR = 0 Q = 0

Ring Counter

It is type of counter composed of flip flop connected to a shift register with the output of the last flip flop fed into the input of first making a circular or ring structure.

Application:-

Used in these application in which

→ where we have to perform several function sequentially.

→ Used in hardware design

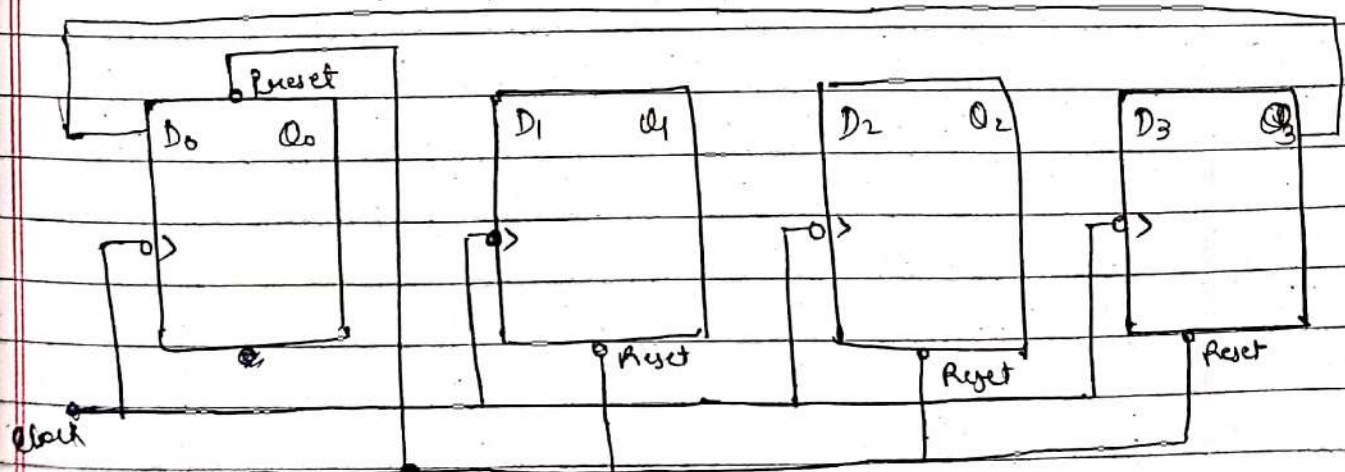
ex. FPGP (field programmable gate array)

Advantage:-

Propagation delay of ring counter will be nearly constant regardless of no. of bits in the code.

Disadvantage:-

It can only represent n-state



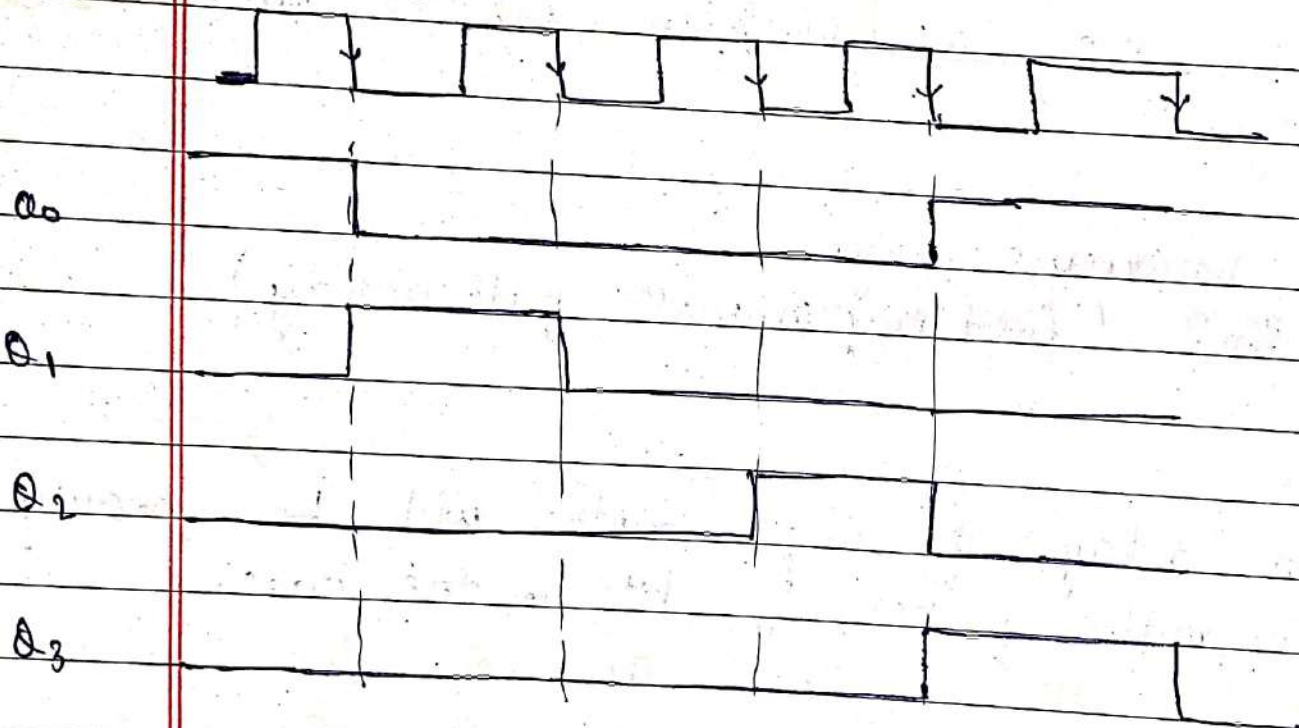
ORI (over riding inputs)

	ORI	Clock	Q ₀	Q ₁	Q ₂	Q ₃
(0)	Low	x	1	0	0	0
	1	1	0	1	0	0
	1	1	0	0	1	0
	1	1	0	0	0	1
	1	1	1	0	0	0

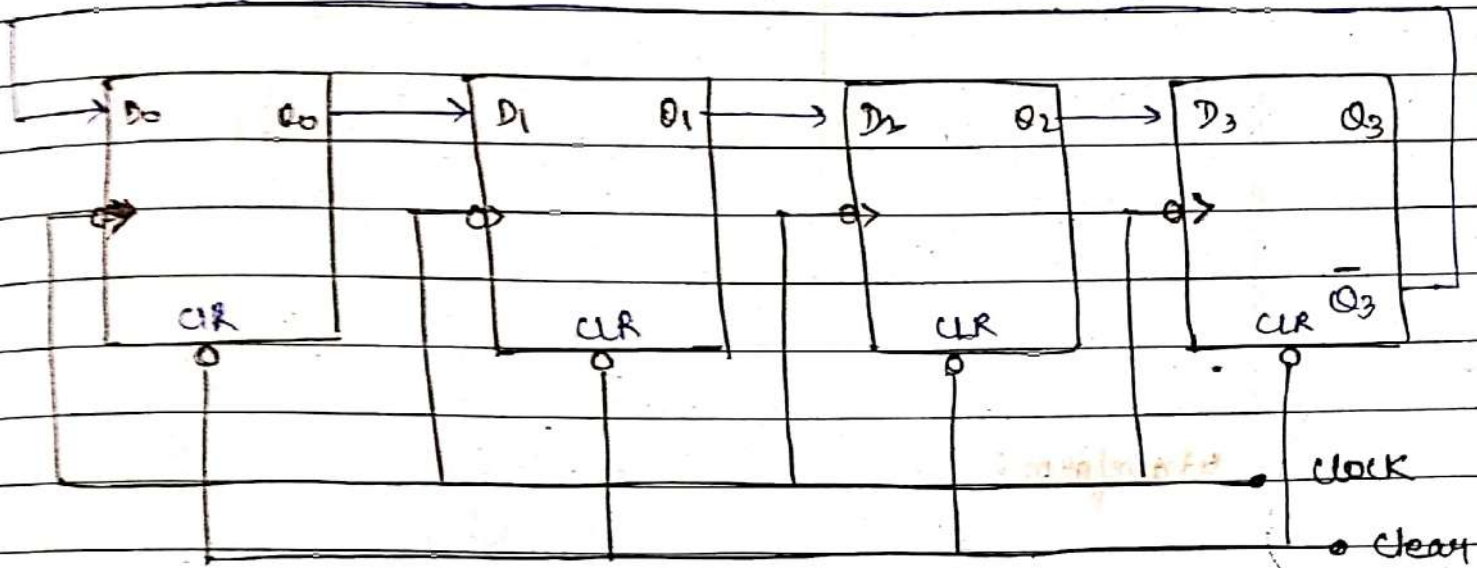
↓

As this is forming a ring that's why it is called ring counter.

Waveform:



Johnson's counter (Twisted / Switch Tail Ring counter)



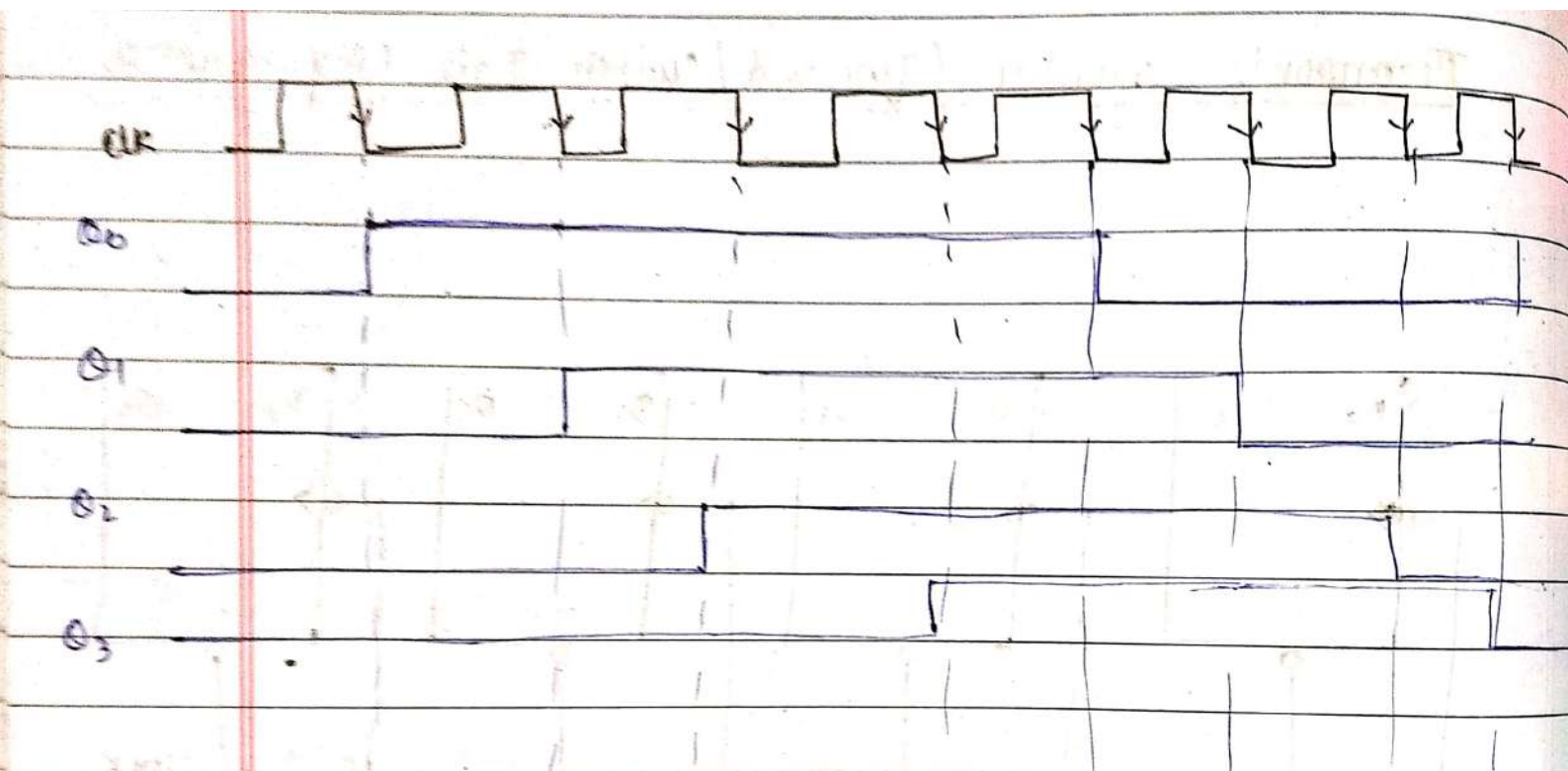
Advantage:-

= No. of States = $2 \times$ No. of flip flops.

In ring counter we start from 1 but here in Johnson's counter also ^{start} counting when we do not initiate it to 1.

i.e In Johnson's counter no initialisation required

CLK	CLK	Q ₀	Q ₁	Q ₂	Q ₃	
	X	0	0	0	0	①
1	1	1	0	0	0	②
1	1	1	1	0	0	③
1	1	1	1	1	0	④
1	1	1	1	1	1	⑤
1	1	0	1	1	1	⑥
1	1	0	0	1	1	⑦
1	1	0	0	0	1	⑧
1	1	0	0	0	0	⑨



Memory

- Memory is indispensable part of computer and micro processor based system.
- * A memory unit is a device to which binary information is saved in and then can be retrieved at a later time.
- Data used in a program as well as the instructions for executing the program are stored in memory.
- * It is a collection of cells that can store large quantities of binary information into memory is referred to as a memory write program operation.
- * The process of transferring the stored information out of memory is referred to as a memory read operation.

Classification of memories

- i) Registers, Main Memory And Secondary Memory
- ii) Sequential access memory & Random Access Memory
- iii) Static and Dynamic Memory
- iv) Volatile and Non-Volatile
- v) Magnetic and Semiconductor memory

* (i) Registers, Main memory and Secondary memory

Register :- Registers are available within the CPU to store data temporarily during arithmetic and logical operations like addition, subtraction, AND, OR etc.

They have very low access time, as they are available within the CPU.

Main Memory :-

~~They have very low access time, as they are available inside the CPU.~~

Main memory of a computer is usually a semiconductor type, these are available external to CPU to store program and data (during the execution of program).

→ In main memory each memory location is identified by an unique address and is accessed for read/write operation in a lesser speed than registers.

Secondary Memory

As the storage capacity of main memory is inadequate, secondary or auxiliary memories are added to enhance storage capability.

→ Secondary memory operates at lesser speed when compared to registers and main memory.

* Secondary memory are of magnetic memory type (Magnetic tape, Magnetic drum, floppy disk and Hard disk) that are used to store large quantities of data.

② Sequential Access memory And Random Access memory

Sequential Access memory :-

It is one in which a particular memory is accessed sequentially.

The access time of sequential memory varies depending on the locations to be accessed.

Ex. Magnetic tape memory.

Random access memory :-

It is one in which any location can be accessed in a random manner and thus has equal access time for all memory location.

Ex. ~~Random~~ semiconductor RAM.

③ Static And Dynamic Memory

Static Memory

Its content does not change with time.

Static memory devices require no refreshing, and hold data as long as DC power is supplied.

Ex. register & MOS cell

Dynamic Memory

Its content changes with time.

Dynamic memory cell use the capacitance of a transistor as storage device.

The capacitor must be refreshed periodically without being discharged in order to prevent loss of information.

Ex. semiconductor dynamic RAM, circulating registers using charge-coupled devices (CCD).

* Volatile And Non-volatile Memory

→ Volatile memory loses its stored data when power to the memory circuit is removed.

→ Non-volatile memory retains stored data permanently even after the power supply is turned off.

Ex. Magnetic core memory & Read only memory

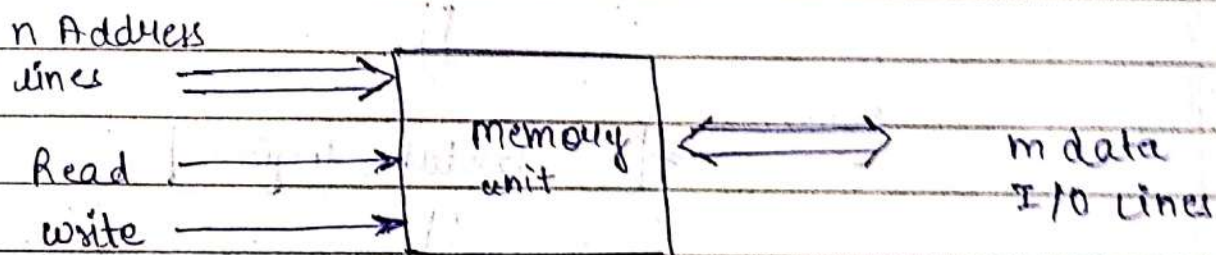
Magnetic and semiconductor memory

Magnetic memory :- The magnetic memories are constructed using magnetic material
Ex. magnetic tape, floppy & compact disk

Semiconductor memory :- These are constructed out of semiconductor material using LSI and VLSI technologies.

Ex. RAM and ROM.

Basic Memory Structure



ROM (Read only Memory)

A read only memory is a semiconductor memory device used to store the information permanently.

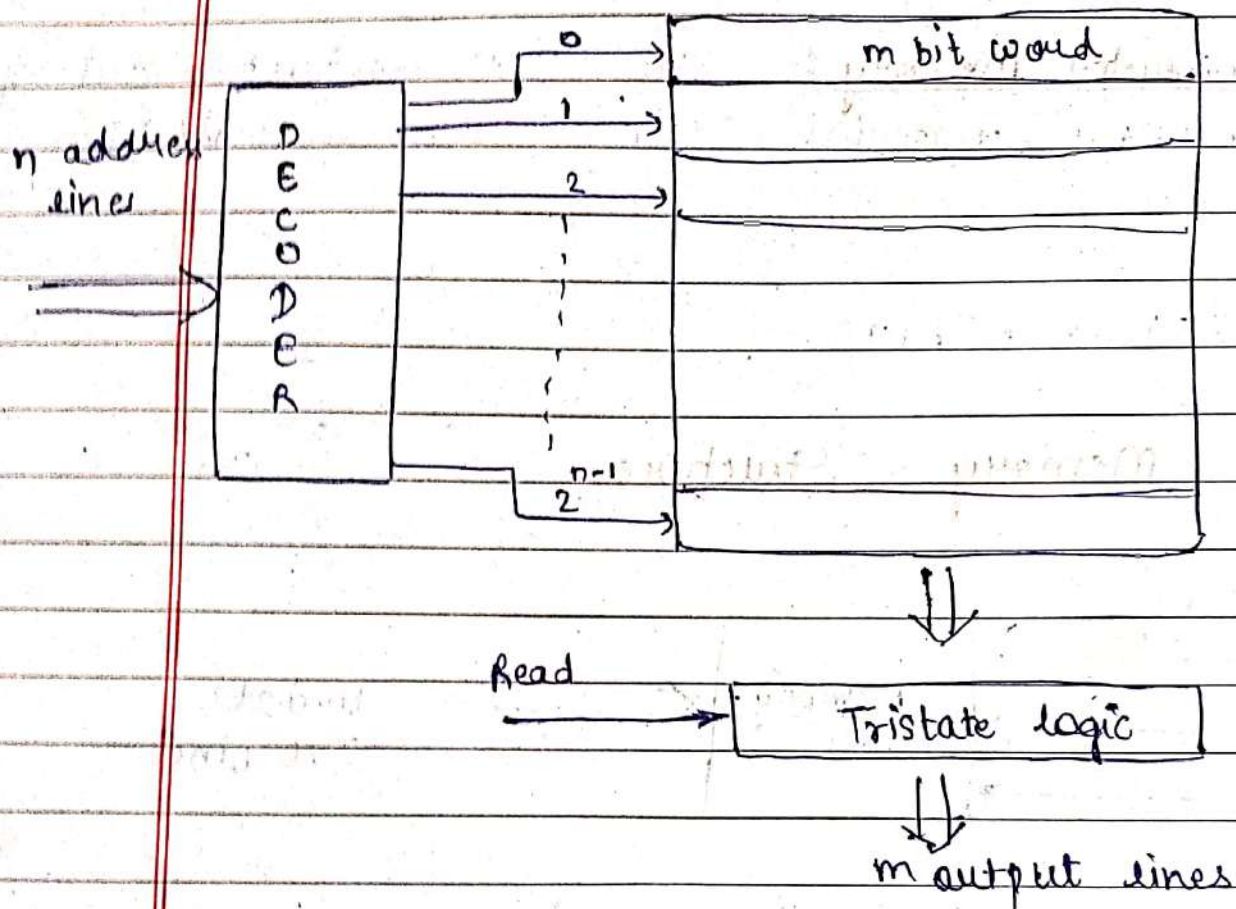
→ It only performs read operation. and do not have write capability

→ ROM is programmed for a particular purpose during the manufacturing process

* User cannot alter this information.

→ The ROM is a combinational logic circuit. It includes both the decoder and the OR gate within the single IC package.

Block diagram of a ROM:-



It consists of n address lines and m output lines.

→ Each bit combination of the address variables is called an address.

Each bit combination that comes out of the output lines is called data word.

No. of bits per word is equal to no. of output lines.

2^n distinct addresses In a ROM, there are 2^n distinct words that are said to be stored in the unit.

→ consider 32×8 ROM,

This unit consists of 32 words of 8 bit each

Applications of ROM :-

- ① ROM are used for a variety of tasks within a digital system.
- ② Use for code conversion.

PLA [Programmable Logic Array]

* It is a type of fixed architecture logic devices with programmable AND gates followed by programmable OR gates.

Step 1:

A	B	C	Y_1	Y_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

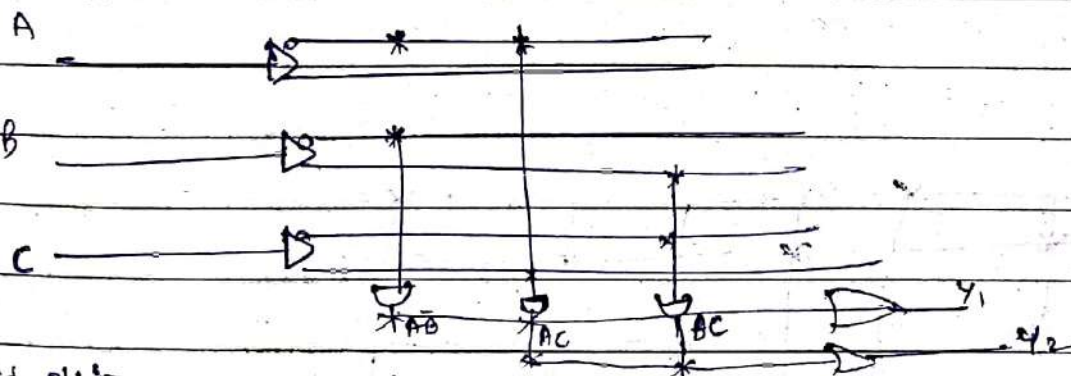
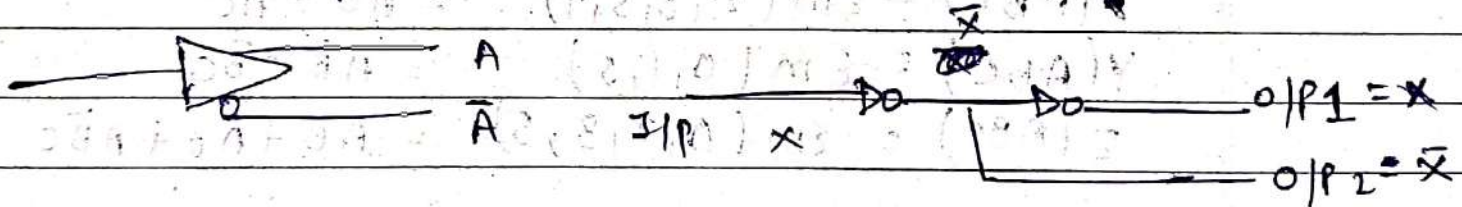
Step 2

$$Y_1 = A\bar{B}\bar{C} + A\bar{B}C + ABC$$

$$Y_1 = A\bar{B} + AC$$

$$Y_2 = BC + AC$$

Step 3: No. of input Buffer = No. of variables

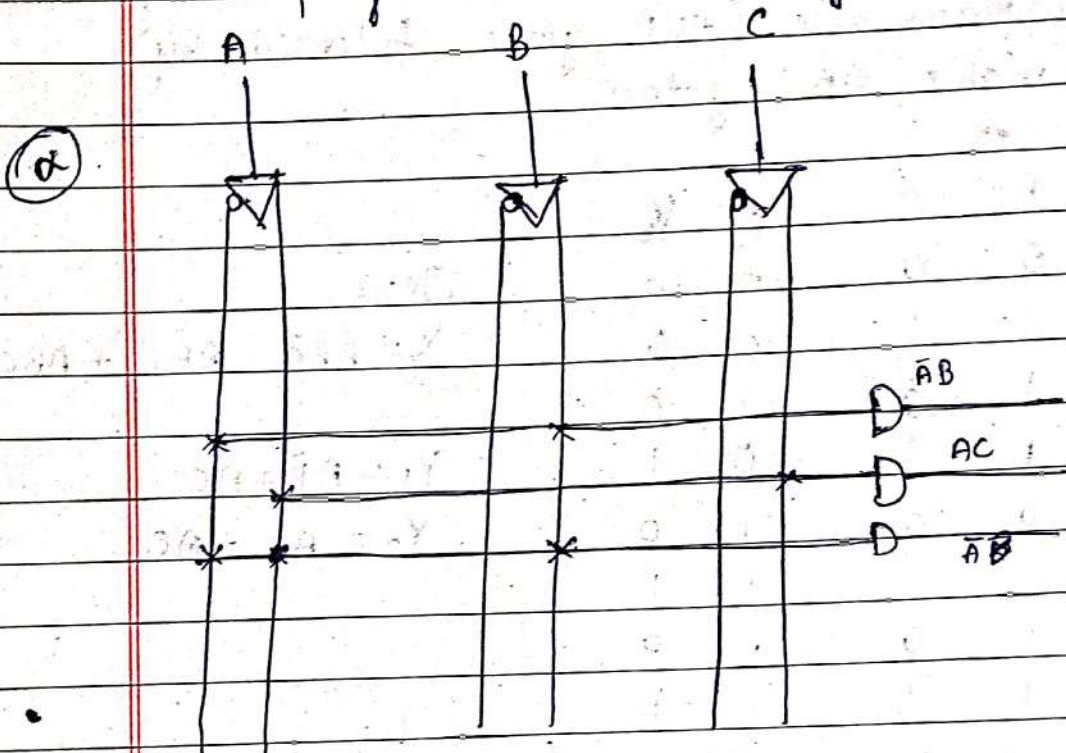


Step 4: No. of Programmable AND gate = 3 (= No. of Minterms)

Step 5: No. of programmable OR gate = 2 (= No. of outputs)

PAL (Programmable array logic)

- It is most commonly used type of PLD.
- Has programmable AND array and fixed OR array.



$$X(ABC) = \sum m(2, 3, 5, 7) = \bar{A}B + AC$$

$$Y(ABC) = \sum m(0, 1, 5) = \bar{A}\bar{B} + \bar{B}C$$

$$Z(ABC) = \sum m(0, 2, 3, 5) = \bar{A}B + \bar{A}C + A\bar{B}C$$

