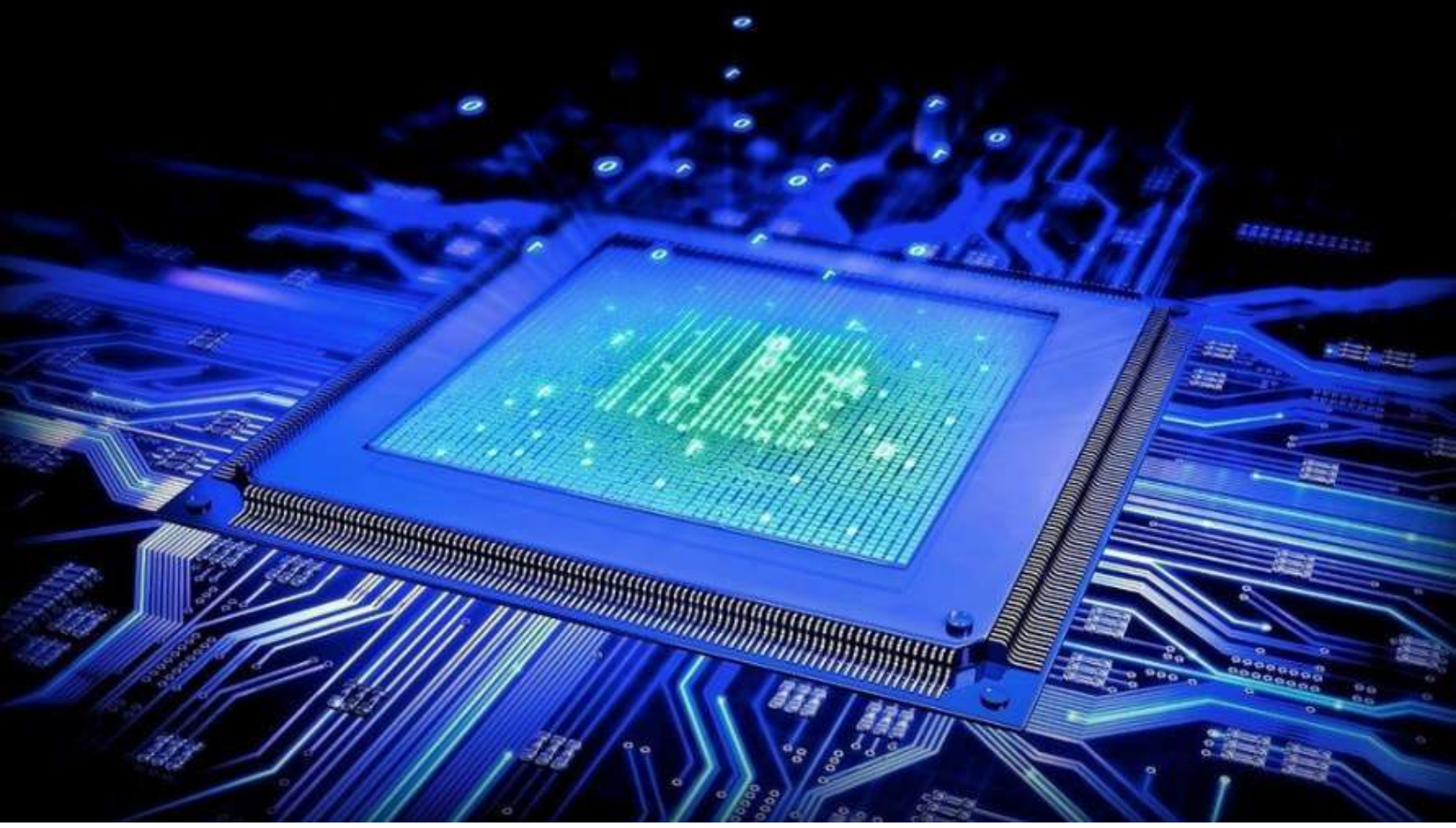
DIGITAL ELECTRONICS



Digital Electronics

DEVARAKONDA SURESH JAHNAVI

→ Signal: A func, that represents the variation of a physical quantity with respect to any parameter

(or)

Signal is variation of electrical quantity (V (or) I) with time.

Analog signal: can define any value within interval

Discrete time signal: can define for discrete time interval (subset of analog signal).

Digital signal:- Here, we represent both time & magnitude. We divide the magnitude axis in to fixed number of levels and "the signals can take value equal to these values/levels only".

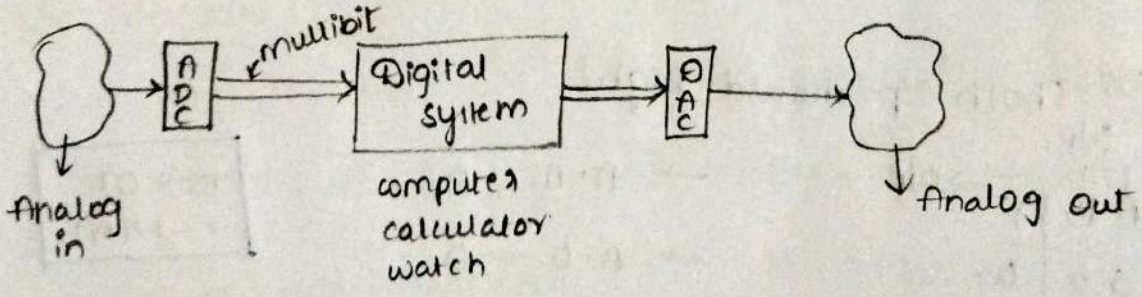
Need of Digital signal:-

→ All real life signals are analog

→ Digital signal is used in communication process to minimize the effect of noise (unwanted signal).

→ Digital signal is good when noise is small

Introduction of digital signal:-



Digital system

↓

Sub system

↓

Module

↓

Basic unit (logic gates)

↓

Circuits (transistors, resistors, capacitors)

Advantages of digital signal over analog:-

→ Noise immunity
→ Low B.W usage
→ Encryption
→ Efficiency is high for long distance transmission

0Volt → "0" → "OFF"
5Volt → "1" → "ON"

342 vs 101
↓ ↓
3-digit number 3-bit number

→ 8 bit = 1byte

n = no.of levels
m = no.of switchs = no.of bits
$n = 2^m$

Eg: 0-5 (1 switchs)
0V, 5V

0-5 (2 switchs)
0V, 1.25V, 2.5V, 3.75V

m↑, accuracy ↑.

Introduction to Boolean Algebra:-

→ It is the set of rules used to simplify the given logic expression "without changing its functionality".

→ It is used when no.of variables are less.

Rules:

i) Complement: (NOT) A complement = $\bar{A}$ or A' or (not A) [$\bar{0}=1$, $\bar{1}=0$]

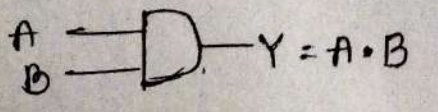
* $\overline{(\bar{A})} = A$

(ii) AND: (both i/p should high)

i/p A	i/p B	o/p Y
0	0	0
0	1	0
1	0	0
1	1	1

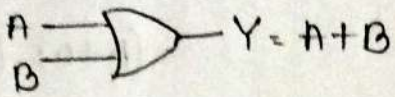
→ $A \cdot A = A$
→ $A \cdot 0 = 0$
→ $A \cdot 1 = A$
→ $A \cdot \bar{A} = 0$

+ → OR
· → AND



iii) OR: (either one i/p should be high)

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



→ $Y = A+A = A$

→ $A+0 = A$

→ $\boxed{A+1 = 1}$

→ $A+\bar{A} = 1$

iv) Distributive law:-

→ $A\cdot(B+C) = A\cdot B + A\cdot C$

→ $A+(B\cdot C) = (A+B)\cdot(A+C)$

* → $\boxed{A+\bar{A}B = A+B}$

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

$= 1\cdot(A+B)$

$= A+B$

* → $\boxed{\bar{A}+AB = \bar{A}+B}$

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

$= 1\cdot(\bar{A}+B)$

(v) Commutative law:-

→ $A+B = B+A$

→ $A\cdot B = B\cdot A$

(vi) Associative law:-

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

Priority:- NOT, AND, OR

* vii) De Morgan law:

V.I

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

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

Eg: 1) $Y = BA\bar{C} + \bar{B}A\bar{C} + B\bar{C}$

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

$= A\bar{C}\cdot 1 + B\bar{C}$

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

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

2) $Y = AB + AB' \Rightarrow A[B+B'] = A$

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

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

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

$= A[B+\bar{B}]$

$= A$

Advantages of minimization:

→ Reduction of hardware (↓ gates)

→ Cost ↓, speed ↑, area ↓

Eg's: 4) $Y = (A+B+C)(A+\bar{B}+C)(A+B+\bar{C})$

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

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

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

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

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

$= A+BC$

$Y = A+BC$

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

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

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

$= A\cdot\bar{A}$

$Y = 0$

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

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

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

$= AB+\bar{A}\cdot C+ABC+\bar{A}BC$

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

$= AB\cdot 1+\bar{A}C\cdot 1$

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

Redundancy theorem:-

→ three variables

→ Each variable is repeated twice

→ one variable is complemented

→ Take the complemented variable means don't redundant that variable terms.

$Y = A\cdot B+\bar{A}\cdot C+B\cdot C$ → redundant term

$= A\cdot B+A'C$

7) $F = AB+B\bar{C}+AC$

redundant

$F = B\bar{C}+AC$

8) $f = A\bar{B}+BC+A\bar{C}$ → redundant

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

9) $F = (A+B)\cdot(\bar{A}+C)\cdot(B+C)$ → redundant

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

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

redundant

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

10) $F = \bar{A}\bar{B}+A\bar{C}+B\bar{C}$ → ?

if all variables are complemented then take non-complemented term.

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

Sum of product (sop form): (used when the o/p is "1")

	A	B	C	F
m_0	0	0	0	0
m_1	0	0	1	0
m_2	0	1	0	1
m_3	0	1	1	0
m_4	1	0	0	1
m_5	1	0	1	1
m_6	1	1	0	1
m_7	1	1	1	1

Total no. of combinations $= 2^n$

n = no. of variables

$= 2^3$

$= 8$

$\bar{x} = 0$

$x = 1$

(sop form) $F = \bar{A}\cdot B\cdot\bar{C}\ (m_2) + A\cdot\bar{B}\cdot\bar{C}\ (m_4) + A\bar{B}\cdot C\ (m_5) + A\cdot B\bar{C}\ (m_6) + AB\cdot C\ (m_7)$

standard (or) canonical sop form (because we directly written from truth table).

Minterms → SOP

$F(A,B,C) = m_2 + m_4 + m_5 + m_6 + m_7$

$= \Sigma m(2,4,5,6,7)$

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

$= \bar{A}B\bar{C} + A\bar{B}[\bar{C}+C] + AB[\bar{C}+C]$

$= \bar{A}B\bar{C} + A\bar{B} + AB \Rightarrow \bar{A}B\bar{C} + A[\bar{B}+B]$

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

$F = A + B\bar{C}$ ← minimal SOP form

reduce: $A + \bar{A}B$, $A + B$

↳ Each minterm does not have all the variables in normal or complemented form. but in canonical/standard sop form each minterm is having all the variables in normal or complemented form. [eg: $F = \bar{A}B + AB + \bar{A}\bar{B}$ → here we have $A, B, \bar{A}, \bar{B}$]

Q)

	A	B	Y
m_0	0	0	0
m_1	0	1	1
m_2	1	0	0
m_3	1	1	1

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

$Y = B$

Q) Simplify the expression for

$Y(A,B) = \Sigma m(0,2,3)$

$= m_0 + m_2 + m_3$

$= \bar{A}\bar{B} + A\bar{B} + AB$ ✓ (AB can sol)

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

$Y = \bar{B} + A$ → minimal SOP

Product of sum (pos):-

Ex:

A	B	C	Y	
0	0	0	0	M0
0	0	1	0	M1
0	1	0	1	M2
0	1	1	0	M3
1	0	0	1	M4
1	0	1	1	M5
1	1	0	1	M6
1	1	1	1	M7

→ pos form is used when the o/p is "0".

0 → A
1 → $\bar{A}$

$Y = (A+B+C)\cdot(A+B+\bar{C})\cdot(A+\bar{B}+\bar{C})$ → canonical / standard pos form

3 maxterms

Pos = $\overline{Sop}$ $\quad Y = (A,B,C) = \Pi(M0, M1, M3)$

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

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

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

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

$= A + B\bar{C}$

$Y = (A+B)\cdot(A+\bar{C})$ → Minimal pos form

QY

	A	B	Y
M_0	0	0	1
M_1	0	1	0
M_2	1	0	1
M_3	1	1	0

$Y = (A+\bar{B})\cdot(\bar{A}+\bar{B})$ → standard pos form

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

$Y = \bar{B}$ → Minimal form.

(In canonical pos form each maxterm will have all the variables (in normal/complimented form) & for minimal pos vice versa

$Y = \Pi(M_1, M_3)$

↑ Product

$Y = \Pi M(1, 3)$

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

Sop Form :- More AND gates

pos form :- More OR gates

sop & pos form examples:

	A	B	C	Y
m_0	0	0	0	1
M_1	0	0	1	0
m_2	0	1	0	1
m_3	0	1	1	1
M_4	1	0	0	0
M_5	1	0	1	0
m_6	1	1	0	1
m_7	1	1	1	1

sop $Y(A,B,C) = \Sigma m(m_0, m_2, m_3, m_6, m_7)$

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

pos $Y(A,B,C) = \pi M(1,4,5)$ rest numbers

Sop: $Y = \bar{A}\cdot\bar{B}\cdot\bar{C} + \bar{A}\cdot B\cdot\bar{C} + \bar{A}\cdot B\cdot C$ } canonical sop form

$A\cdot B\cdot\bar{C} + \dot{A}\cdot BC$

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

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

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

$Y = \bar{A}\bar{C} + B$ → Minimal sop form

Pos: $Y = (A+B+\bar{C})\cdot(\bar{A}+B+C)\cdot(\bar{A}+B+\bar{C})$ ↳ canonical pos form

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

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

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

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

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

$Y = (B+\bar{C})\cdot(B+\bar{A})$ → Minimal pos form

Minimal to Canonical form Conversion

Q) $Y = \underset{M_1}{A} + \underset{M_2}{\bar{B}C}$ [1 OP]

Step-1: how many variables : 3
what are they : A, B, C

Step-2:

M_1		M_2	
A	✓	A	✗
B	✗	B	✓
C	✗	C	✓

Step 3: See example

2) [POS] $F = (A+B+\bar{C})(\bar{A}+C)$

(i) A, B, C

(ii)

Term-1		Term-2	
A	✓	A	✓
B	✓	B	✗
C	✓	C	✓

1) eg 1 [1 OP]

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

$Y = A \cdot 1 \cdot 1 + \bar{B}C \cdot 1$

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

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

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

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

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

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

$= (A+B+\bar{C})(\bar{A}C+B).(\bar{A}+C+\bar{B})$ → Canonical pos

Q) In a minimal sop form, the number of minterms in logical expression $A+\bar{B}C$ are 5.

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

$F =$ eg 1 [see top example]

Q) With 2 variables maximum possible minterms & maxterms are:

2^n = min/max terms

$n = 2$

4 max possible min & max terms are there.

Ok for n=2, what is the total no.of logical expression?

L no.of variable.

n=2 (A,B)

1	A	$A\bar{B}$	$AB=BA$
0	$\bar{A}$	$\bar{A}B$	$A+B=B+A$
$\bar{A}B+A\bar{B}$	B	$A+\bar{B}$	$\bar{A}\bar{B}$
AB $\bar{A}\bar{B}$	$\bar{B}$	$\bar{A}+B$	$\bar{A}+\bar{B}$

16 logical exp

* ∴ Total no of logical expression = 2^{2^n}

n = no.of variables

if n=4

total no.of logical expression = $2^{2^4} = 2^{16} = 65536$

Positive and Negative logic:

+ve logic (binary)

→ Higher voltage corresponds to logic '1'

→ Lower voltage corresponds to logic '0'

5V → logic '1'

0V → logic '0'

-ve logic

→ Higher voltage corresponds to logic '0'

→ Lower voltage corresponds to logic '1'

5V → logic '0'

0V → logic '1'

Ok Logic 0 → -5V

logic 1 → 0V

→ +ve (or) -ve ? ✓

Ok logic 0 → -1.7V

logic 1 → -0.8V

positive logic ✓

Dual form:

→ +ve & -ve logic AND gate
→ +ve & -ve logic OR gate

AND

A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

OR

A	B	Y
L	L	L
L	H	H
H	L	H
H	H	H

+ve logic OR

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

+ve logic AND gate

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

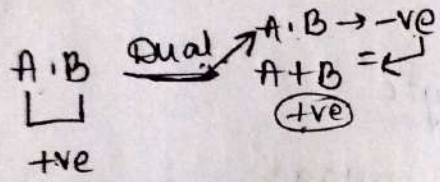
-ve logic AND gate

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

-ve logic OR

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

Ⓗ +ve logic AND gate = Ⓛ -ve logic OR gate
Ⓛ +ve logic AND gate = Ⓗ +ve logic OR gate



Self Dual:

→ for any logical expression, 2 times dual gives the same expression.
→ In self dual expression one time dual gives the same expression.

Dual

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

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

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

$F = \bar{\bar{F}}$

Self Dual:

$G = A \cdot B + B \cdot C + A \cdot C$

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

$G = \bar{G}$

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

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

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

$= A \cdot A + A \cdot C + B \cdot C$ ✓

→ n-variables → how many self duals are possible

$= 2^{2^{n-1}}$

n = 2

then, NO·OF S·D expressions $= 2^{2^{2-1}} = 4$

eg: A & B

$A \to A \mid B \to B$

$\bar{A} \to \bar{A} \mid \bar{B} \to \bar{B}$ } 4 S·D expressions

H·W

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

Self Dual or not: $\bar{Y} = (\bar{A}+\bar{B})\cdot(\bar{B}+\bar{C})\cdot(\bar{A}+\bar{C})$

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

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

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

ok n = 4

NO·OF S·D expressions $= 2^{2^{4-1}} = 2^{2^3} = 2^8$

Complement :-

$U = \{a, e, i, o, u\}$

$A = \{a, e, i\}$

$\bar{A}$ or A'

$\bar{A} = U - A$

$\bar{A} = \{o, u\}$

$U = A + \bar{A}$

$= \{a, e, i\} + \{o, u\}$

$U = \{a, e, i, o, u\}$

$A \leftrightarrow 0, 1$

$U = \{0, 1\}$

$A = \{0\}$

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

$U = A + \bar{A}$

$\bar{0} \to 1$

$\bar{1} \to 0$

eg: $F = \bar{A}\cdot B + \bar{B}\cdot A$

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

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

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

$F = \bar{A}\cdot\bar{B} + AB$

2) $G = ABC + \bar{A}BC + A\bar{B}C$

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

$= [\bar{B}+\bar{C} + (A\cdot\bar{A})]\cdot(\bar{A}+B+\bar{C})$ ($A\cdot\bar{A} = 0$)

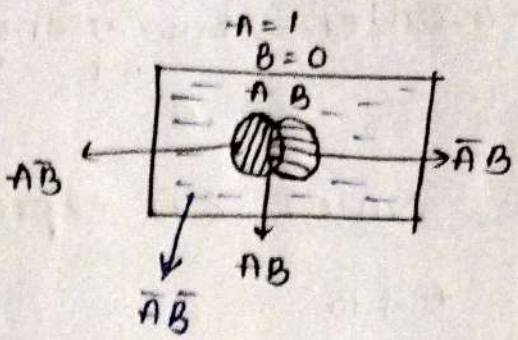
$= (\bar{B}+\bar{C})\cdot(\bar{A}+B+\bar{C})$

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

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

Venn Diagram:-

two variables (A, B)



Minimize the SOP expression for shaded region

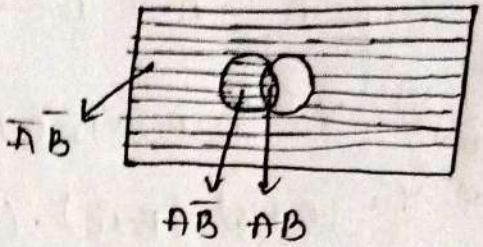
1) $Y = \bar{A}\bar{B} + A\bar{B} + AB$

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

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

$Y = A + \bar{B}$

$G = (\bar{A} + A)$

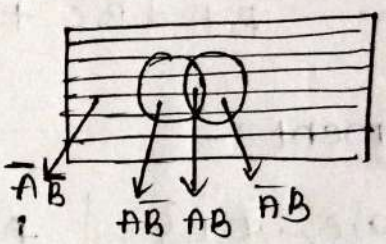


2) $G = \bar{A}\bar{B} + \bar{A}B + A\bar{B} + AB$

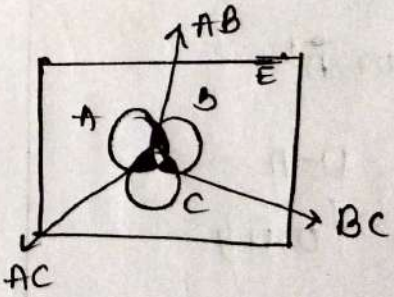
$G = \bar{B}(\bar{A} + A) + B(\bar{A} + A)$

$G = \bar{B} + B$ ...

$G = 1$



$(AB + BC + CA)\bar{E}$ ✓ ←



Switching circuit:

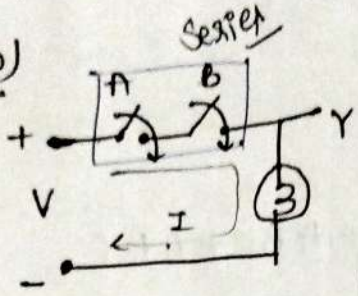
M → switch / variables

N → combinations

$\boxed{2^M = N}$ or $M = \log_2 n$

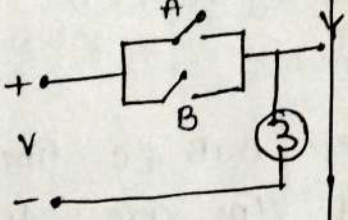
AND: $Y = A \cdot B$ → parallel

A	B	Y (bulb)
0	0	0
0	1	0
1	0	0
1	1	1



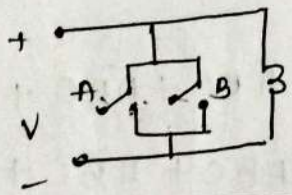
OR: "+"

A	B	Y (bulb)
0	0	0
0	1	1
1	0	1
1	1	1

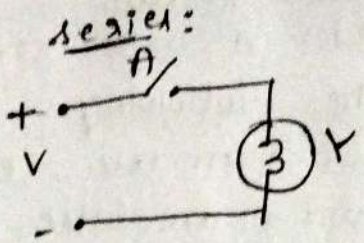


NOR:

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

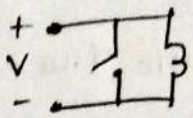


Series:



A	Y
0	OFF '0'
1	ON '1'

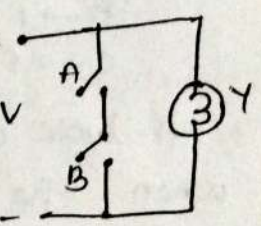
Parallel:



A	Y
0	ON '1'
1	OFF '0'

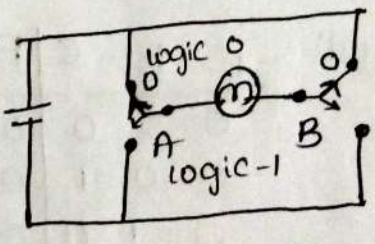
NAND:

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



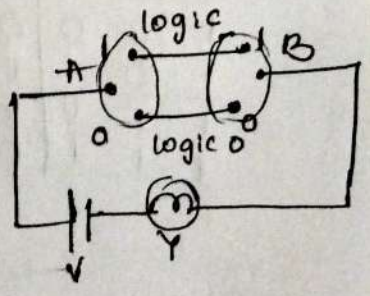
EXOR: "Odd 1's detector"

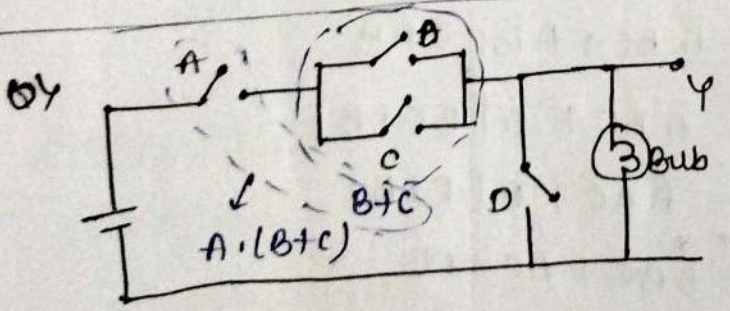
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

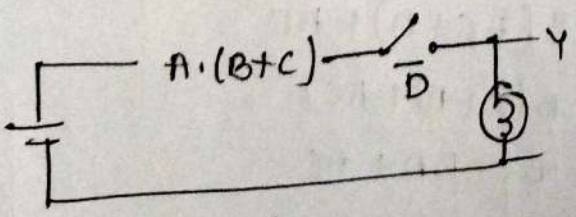


EXNOR:

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







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

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

Q) statement problem:

1) Consider a logic circuit with 3 i/p A, B and C. O/p Y is SOP (high) for the following

(i) B and C are true $= B \cdot C$

(ii) A and C are false $= \bar{A}\bar{C}$

(iii) A, B, C are true $= A \cdot B \cdot C$

(iv) A, B, C are false $= \bar{A} \cdot \bar{B} \cdot \bar{C}$

Minimize the function Y.

Sol: $Y = B \cdot C + \bar{A}\bar{C} + ABC + \bar{A}\bar{B}\bar{C}$

$= BC(1+A) + \bar{A}\bar{C}(1+\bar{B})$

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

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

2) Y for the

(i) A & B are true

(ii) $A \rightarrow T$, $B \rightarrow F$, $C \rightarrow T$

(iii) $A \rightarrow T$, $B \rightarrow F$, $C \rightarrow F$

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

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

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

$Y = A$

3) A logic ckt have 3 i/p's A, B & C. The output F is high when the majority of i/p's are logic 1.

a) Minimize the func

b) Implement the ckt

Sol:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$f = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

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

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

$f = \bar{A}BC + A[\bar{B}C + B]$

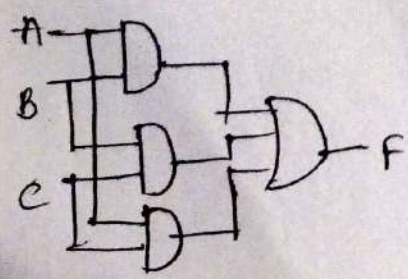
$f = \bar{A}BC + A[C + B]$

$f = \bar{A}BC + AC + AB$

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

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

$f = BC + BA + AC$



Number System:-

→ Number system defines a set of values used to represent quantity.

Name	Base (r) (radix (or))	0 to (r-1)
Binary	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Duodecimal	12	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
	4	0, 1, 2, 3

→ Number system & codes → weighted [decimal, Binary, octal]
→ unweighted [Gray code, excess 3-code, BCD etc...]
↳ divided based on weights & positions

weighted eg:

$7392 = 7000 + 300 + 90 + 2$

$= 7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$

radix $r_1 \rightarrow r_2$

$N_1 \rightarrow N_2$

N_1 ↙ total no. of digits

$\boxed{r_1 < r_2 \quad N_1 > N_2}$

eg:

$(7392)_{10} \rightarrow (1110011100000)_2$

$(7392)_{10} \rightarrow (16340)_8$

$(7392)_{10} \rightarrow (1CE0)_{16}$

Binary No. system:-

Base/radix = 2 → {0, 1}

Binary digits (0, 1) are called as "bits".

eg: 1 0 1 0 1 (2^4 2^3 2^2 2^1 2^0)

$\rightarrow 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^3 + 0 \times 2^1 + 1 \times 2^0$

$= 1 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$

$= 16 + 0 + 4 + 0 + 1$

$= 21$

MSB & LSB

↳ if we change LSB (no much change)

↳ (if we change (Almost everything will change)

$b_4\ b_3\ b_2\ b_1\ b_0$

1 0 1 0 1 ↳ LSB

MSB ↵

→ Bit is smallest unit of data

1 Nibble = 4 bits

1 Byte = 8 bits

1 word = 16 bits = 2 bytes

1 double word = 32 bits = 4 bytes

Conversions:- Decimal ← Binary, octal, Hexadecimal | Binary ← Decimal, octal, hexadecimal

octal ← Decimal, Binary, Hexadecimal | Hexadecimal ← Binary, Decimal, hexadecimal.

Decimal to Binary/octal/hexadecimal:-

For integer part :- weighted division

for fractional part : weighted multiplication:

D to B:- eg1) $(13)_{10} \rightarrow (\ ?\ )_2$

$= (1101)_2$

2 | 13
2 | 6 -1 ↑ LSB
2 | 3 0
2 | 1 -1
-1 ↑ MSB

eg2) $(25.625)_{10} \rightarrow (11001.101)_2$

2 | 25
2 | 12 -1 ↑ LSB
2 | 6 - 0
2 | 3 - 0
1 -1
MSB

25 → 11001

0.625 x 2 = (1).25 = 1 ↓
0.25 x 2 = (0).50 = 0
0.50 x 2 = (1).00 = 1
stop

0.625 → 101

D to O: ① $(112)_{10} \rightarrow (160)_8$

8 | 112
8 | 14 - 0
1 - 6 ↑

② $(25.625)_{10} \rightarrow (31.50)_8$

8 | 25
3 - 1 ↑

$0.625 \times 8 = 5.000$
$0.000 \times 8 = 0.000$

D to H: $(25.625)_{10} \rightarrow (19.A0)_{16}$

16 | 25
1 - 9 ↑

$0.625 \times 16 = 10.0000$ (↓ A)
$0.000 \times 16 = 0.0000$

Binary to Decimal:-

[Bigaly, octal, Hexadecimal to Decimal]

B to D: $(10101.11)_2 \rightarrow (21.75)_{10}$

1 0 1 0 1 . 1 1
$2^4\ 2^3\ 2^2\ 2^1\ 2^0\ .\ 2^{-1}\ 2^{-2}$

weighted multiplication

$\rightarrow 1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$

$= 16+0+4+0+1+\frac{1}{2}+\frac{1}{4}$

$= 21+0.75$

$= (21.75)_{10}$ ✓

O to D:-

$(57.4)_8 \rightarrow (47.5)_{10}$

5 7 . 4
$8^1\ 8^0\ .\ 8^{-1}$

$= 5\times8^1+7\times8^0+4\times8^{-1}$

$= 47.5$

H to D:-

$(BAD.4) \rightarrow (2989.25)_{10}$

$B\ A\ D\ .\ 4$
$16^2\ 16^1\ 16^0\ .\ 16^{-1}$

$= 11\times16^2+10\times16^1+9\times16^0+4\times16^{-1}$

$= 2816+160+13+0.25$

$= 2989.25$

Octal to Binary & Binary to octal:-

O to B:-

$(34.41)_8 \rightarrow (011111.100101)_2$

011 111 . 100 101

B to O:

$(1001.1)_2 \rightarrow (11.4)_8$

001 001 . 100

1 1 . 4

Hexadecimal to Binary & Binary to hexadecimal:-

H to B:-

1100 1010

$(CA7E.3D)_{16} \rightarrow (1100\ 1010\ 1111\ 1110.0011\ 1101)_2$

1111 1110 0011 1101

B to H:

$(1000\ 1001.11)_2 \rightarrow (89.C)_{16}$

1000 1001 . 1100

8 9 C

Hexa to octal:-

hexa to Binary to octal:

$(CAD)_{16} \rightarrow (\quad)_8$

$(CAD)_{16} \rightarrow (1100\ 1010\ 1101)_2$

$(1100\ 1010\ 1101)_2 \rightarrow (6255)_8$

octal to hexa:-

octal to Binary to hexa:

$(652)_8 \rightarrow (\quad)_{16}$

$(652)_8 \rightarrow (110\ 101\ 010)_2$

$0001\ 1010\ 1010 \rightarrow (1AA)_{16}$

Binary Addition:-

```
    1 1 0
(+) 1 0 1
---------
  1 0 1 1
  ↓ ↓
Carry Sum
```

	sum	carry
0+0	0	0
0+1/1+0	1	0
1+1	0	1

Binary Subtraction:-

(i)
```
    1 1 0 1 1
(-) 1 0 1 1 0
-------------
    0 0 1 0 1
```

$\rightarrow$ $1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$ (take borrow: $(2-1)$ 3, 1×2^2, $(1-1)$, $(1-0)$)

(-) $1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

$0\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$1\times2^3 + 0\times2^2$; $1\times2^3 = 2\times2^2$; $(2-1)\times2^2$; 1×2^2

(-) $0\times2^3 + 1\times2^2$

(ii)
```
  1 1 1 0
-   1 1 1
---------
  0 1 1 1
```

$0\times2^3 \rightarrow 1\times2^2 \rightarrow 2\times2^1 \rightarrow 1\times2^1 = 2\times2^0 - 1\times2^0$

$1\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

(-) $0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

$0 + 1\times2^2 + 1\times2^1 + 1\times2^0$

Binary Multiplication:-

$0\times0 = 0$

1×0 or $0\times1 = 0$

$1\times1 = 1$

Eg:-
```
    1 0 1 0
(x)   1 0 1
```

$1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

(x) $+ 1\times2^2 + 0\times2^1 + 1\times2^0$

$1\times1=1$ $1\times0=0$ $1\times1=1$ $1\times0=0$

$0\times1=0$ $0\times0=0$ $0\times1=0$ $0\times0=0$ X

(+) $1\times1=1$ $1\times0=0$ $1\times1=1$ $1\times0=0$ X X

1 1 0 0 1 0

```
     1 0 1 0
(x)  0 1 1
-----------
     1 0 1 0
   1 0 1 0 x
 0 0 0 0 x x
-----------
 0 1 1 1 1 0
```

```
     1 0 1 0
(x)      1 1
-----------
     1 0 1 0
   1 0 1 0 x
-----------
   1 1 1 1 0
```

Binary Division:-

```
        0 0 0 1 1 1
1 1 0 ) 1 0 1 0 1 0
       -0 (110x)
        1 0
       -  0 (110x)
        1 0 1
       -    0 (110x)
        1 0 1 0
       -1 1 0 x1
        0 1 0 0 1
    (-)   1 1 0 x1
        0 0 1 1 0
    (-)     1 1 0
```

1 < 110

10 < 110

101 < 110

1010 > 110

01001 > 110

```
  1 0 0 1
- 1 1 0
-------
  0 0 1 1
```

$\rightarrow 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$ ($1\times2^3 \rightarrow 2\times2^2$, 1×2^2, $2\times2^1 - 1\times2^1$, 1×2)

(-) $1\times2^2 + 1\times2^1 + 0\times2^0$

$0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$101010 = 110(000111) + 0$

Octal addition:-

(i)
```
  2 4 3
  2 1 2
  -----
  4 5 5
```

(ii)
```
    1 1
    5 6 7
    2 4 3
  -------
  1 0 3 2
```
10 > 7, 10 = 1×8+2

8 > 7 → 1×8+0

11 > 7 → 11 = 1×8+3

Octal multiplication

(i)
```
     7 2
   × 1 2
   -----
   1 6 4
   7 2 X
 (+)
 -------
 1 1 0 4
```
1×8+6

(ii)
```
      3 5 7
        2 3
  ---------
    1 3 1 5
  7 3 6 X
(+)
  ---------
  1 0 6 7 5
```
21 → 8×2+5; 8×2+1; 8×1+3; 8×1+6; 8×1+3; 1×8+0

Octal subtraction:-

```
    7 4 3
(-) 5 6 4
  -------
    1 5 7
```
8+3 = 11

```
  6 2 4
  2 6 5
  -----
  3 3 7
```
1+8 = 9, 8+4 = 12

Hexadecimal Addition:-

(i)
```
  5 6 8 9
  4 5 7 4
  -------
  9 B F D
```

(ii)
```
   A D D
   D A D
  ------
 1 8 8 A
```
13+13 = 26 → 16×1+10 → A

24 → 16×1+8

Hexadecimal subtraction

(i)
```
    9 6 5 4
(-) 5 3 2 1
  ---------
    4 3 3 3
```

(ii)
```
  8 9 7 4 B
    5 8 7 C
  ---------
    3 E C F
```
6+16 = 22−8 = 14; 3+16 = 19−7; 11+16 = 27−12 = 15

Hexadecimal Multiplication

(i)
```
     9 4
   × 1 2
   -----
   1 2 8
   9 4 X
   -----
   A 6 8
```
16×1+2

(ii)
```
       A B C
   (×)   2 3
   ---------
     2 0 3 4
   1 5 7 8 X
(+)
   ---------
   1 7 7 B 4
```
36 → 16×2+4

35 → 16×2+3

32 → 16×2+0

24 → 16×1+8

22+1 = 23 → 16×1+7

20+1 = 21 → 16×1+5

r's complement:- < r's complement (radix complement)
r-1's complement (diminished radix complement)

↳ used to simply subtraction operations.

ex:- comp of $(7)_{10}$

Ans:- 3

* Formula : $r^n - N$ → r's complement

r = radix

N = given no.

n = total no. of digits

1) N = 5690

Determine 10's complement

sol:- r = 10, N = 5690

n = 4

$10^4 - 5690 = 4310$

2) N = 1101

determine 2's comp

sol:- r = 2, N = 1101

n = 4

$= (2^4)_{10} - 1101$

$= 10000 - 1101$

$= 00011$

```
 10000
- 1101
 00011
```

3) N = 76895

10's comp

sol:- $10^5 - 76895$

$= 100000 - 76895$

$= 23105$

4) N = 11011

2's comp

sol:- $= 2^5 - 11011$

$= (32)_{10} - 11011$

$= (100000) - 11011$

```
 100000
- 11011
 00001
```

01

→ (r-1)'s complement:

r's comp	(r-1)'s comp
r = 10	10's comp → 9's comp
r = 2	2's comp → 1's comp
r = 8	8's comp → 7's comp
r = 16	16's comp → f's comp

r's complement = $r^n - N$

(r-1)'s complement = $r^n - N - 1$

(r-1)'s comp = r's comp − 1

eg:- 7's comp of 5674 (octal no)

$= 84 - 5674 - 1$

$= (4096)_{10} - 5674 - 1$

$= (10000)_8 - 5674 - 1$

$= 7777 - 5674$ ✓

$= 2103$

eg 2's comp of 1101

ex 1's comp 0010

2's comp 0011

(r-1)'s comp + 1 = r's comp → borrow involved

↓

NO borrow operation is involved

eg 8's comp of 5674

sol 7's comp of 5674 + 1

2103 + 1 = 2104

1's & 2's complement:

Eg 1) 1's complement of $(1010)_2$ is 0101 → 1111 − 1010 = 0101

2) 2's complement of $(10111010)_2$ is 01000110

1's - 01000101

2's - 01000101 + 1 = 01000110

3) 2's comp of 1100111 is 0011001

0011000 + 1 = 0011001

Shortcut for 2's comp:

Step 1: write down the given number

Step 2: Starting from LSB, copy all the zeros till the first 1.

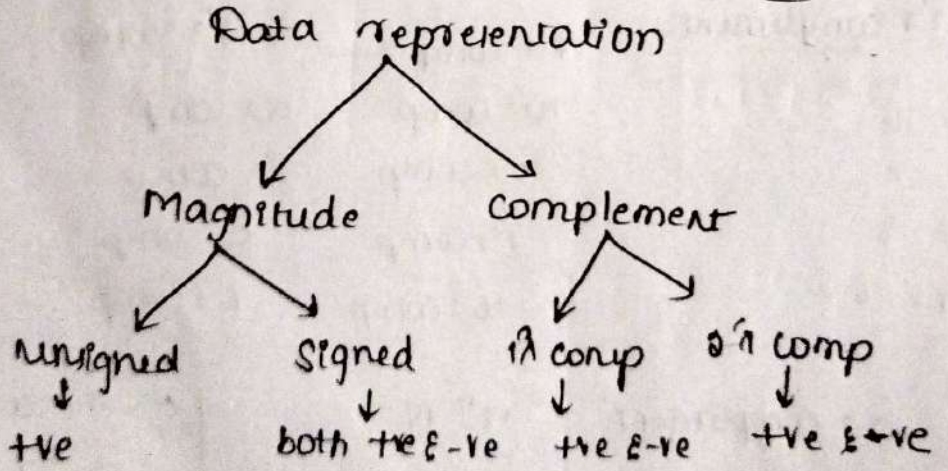
Step 3: Copy the first 1.

Step 4: Complement all the remaining bit.

Eg: 1) 10111000 → 01001000 (2's comp)

2) 101100 → 010100 (2's comp)

Data representation using Signed Magnitude:



→ In all 4 representation +ve no. is represented in same way.

(i) Unsigned Magnitude:-

+6 = 110

-6 = can't represent

(ii) Signed Magnitude:-

(i) +6 = 0110, -6 = 1110 } n=4 → Sign bit + Mag. (+ → 0, − → 1) + Mag

(ii) +13 = 01101, -13 = 11101 } n=5, -15 to +15

range:- * $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

n → no. of variable/bits

If n=4 → -7 to +7

(iii) +9 = 01100

-9 = 11100

(iv) +16 = 010000, -16 = 110000 } n=6, -31 to +31

(iii) 1's complement:

Eg (i) +6 = 0110, -6 = 1001 } 1's comp

(ii) +9 = 01001, -9 = 10110 } 1's comp

+0 = 0000

-0 = 1111 } to overcome this we have 2's comp

range:

* $-(2^{n-1}-1)$ to $(2^{n-1}-1)$

n=4 → -7 to +7 (4 bit range)

n=5 -15 to +15

n=6 -32 to 32

(iii) +10 = 01010

-10 = 10101

v) +4 = 0100

-4 = 1011

iv) +23 = 010111

-23 = 101000

because +23 is range of -32 to +32 ie n=6 (bits)

1's comp of any no. is -ve of that no.

---

v) +9 = 1001

-9 = 10111

(-9 with 2's comp for +9, we get 0111, but this can't be represented as -9 being signed rep, we will have to use 5 bits, hence -9 = 10111).

(iv) 2's complement:

(i) +6 = 0110, -6 = 1010 } 2's complement

(ii) +4 = 0100, -4 = 1100 } 2's comp

only one zero : 0000

Range:

$-(2^{n-1})$ to $(2^{n-1}-1)$

n=4

-2^3 to 2^3-1

-8 to +7

(iii) +5 = 0101

-5 = 1011

1 → -ve

(iv) -8 = 1000 (remember 4 bit representation has range from -8)

+8 = 01000 (4-bit rep has range from -8 to +7, so we need 5 bits)

Binary subtraction using 1's complement:

S1: Convert numbers to be subtracted to its 1's complement form

S2: Perform the addition.

$A - B = A + (-B)$

↑ 1's comp of $B = -B$

*S3: (i) If the final carry is '1', then add it to the result obtained in step 2. (end around carry)

(ii) If the final carry is '0', then result obtained in step 2 is negative & in the 1's comp form

Eg: 1) 1100 − 0101

A = 1100

S1) B = 0101

1's comp of B = 1010

S2) A + (−B) = 1100 + 1010

1's comp of B ⇒ 1100
1010
0110

final carry = 1

So add to result

S3)
0110
+ 1
0111

2) (0101) − (1100)

A = 0101

S1) B = 1100

1's comp of B = 0011

S2) A + (−B) =
0 0 1
0101
0011
1000

final carry = 0

S3) Result obtained = 1000

is −ve, 1's complement it

0111 ✓

Binary sub using 2's complement:

S1: Convert no. to be subtracted to its 2's comp form

S2: perform addition

S3: (i) FC is 1, then result is +ve & its true form

(ii) FC is 0, then result is −ve & its in 2's comp form

Note: Neglect FC in 2's comp (like don't write FC in ans)

Eg: ① (1001) − (0100)
A B

S1) 2's comp B is = 1011 + 1
= 1100

S2)
1001
+1100
1 0101

FC neglect

+ve ✓

Overflow

n = 4

range: −8 to 8, if we take FC = 1 then 10101 = 21 not in range

* Condition for overflow :-

x & y are the sign bit of, a no.

z is the sign bit of result

$\bar{x} \cdot \bar{y} \cdot z + x \cdot y \cdot \bar{z}$ = 0 (No overflow)

= 1 (overflow)

Eg ① $(0110)_2 - (1011)_2$

A = 0110

B = 1011

(1) 2's comp = 0100 + 1 = 0101
of B

(2) 0110
(+) 0101
1011 (-5)

no F.C, result is -ve in
2's comp form

(3) 10 1011
1's comp of result = 0100
2's comp of result = 0101 (+5)

Classification of Codes :- Codes = group of symbols

- **Weighted codes**
  - Eg: Binary
  - 8421
  - 2421
  - etc...

$a_3 a_2 a_1 a_0$ (8 4 2 1)

$b_3 b_2 b_1 b_0$

- **Non-weighted codes**
  - Eg: ex s-3
  - Gray
  - etc...
- **Reflective codes**
  - Eg: (Self Complementing)
  - 2421
  - Ex-S-3

Reflective Codes:

9 → comp of 0

8 → comp of 1

- **Sequential Codes**
  - eg: 8421
  - ExS-3
  - etc...
- **Alphanumeric Codes**
  - Eg: ASCII

ASCII read

- **Error detecting & correcting codes**
  - eg: hammig code

Decimal – 2421 code

0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

Binary coded decimal (BCD) code: (8421 code)

→ each decimal digit (0-9) is represented by a 4-bit binary no.

→ Positional weights are 8-4-2-1

Decimal	BCD(8421)	
0	0000	BCD code for decimal digits
1	0001	
2	0010	
3	0011	
4	0100	
5	0101	→ 10 out of 16 ($2^4 = 16$)
6	0110	
7	0111	
8	1000	
9	1001	
10	0001 0000	xxxx don't cares, invalid
11	0001 0001	
12	0001 0010	
13	0001 0011	
14	0001 0100	
15	0001 0101	

rest 6 are not decimal digits but they are decimal numbers

conversion of decimal to BCD:-

Eg: (i) $(17)_{10} \rightarrow (00010111)_{BCD}$

0001 0111

(ii) $(156)_{10} \rightarrow (\overbrace{0001\ 0101\ 0110}^{\text{packed BCD}})_{BCD}$

conversion of BCD to decimal:-

Eg: (i) $(10100)_{BCD} \rightarrow (14)_{10}$

$\frac{0001}{1}\ \frac{0100}{4}$

(ii) $(1001001)_{BCD} \rightarrow (49)_{10}$

$\frac{0100}{4}\ \frac{1001}{9}$

Packed BCD: BCD numbers representing beyond 9.

Binary vs BCD:

Binary	BCD
$(10)_{10}$ 1010	0001 0000
$(15)_{10}$ 1111	0001 0101

* BCD is less efficient than Binary.
* Arithmetic operations are difficult

BCD Addition:-

(i) sum ≤ 9, Final carry = 0 → Answer is crt

(ii) sum ≤ 9, Final carry = 1 → Answer is incorrect, to crt it just add 6 (0110)

(iii) sum > 9, final carry = 0 → "

Eg:(i) $(2)_{10} + (6)_{10}$

sum = 0010 + 0110

0010
0110
1000 = 8

Sum = 8 ≤ 9 ✓
Carry = 0

(ii) $(3)_{10} + (7)_{10}$

sum → 0011
0111
1010 = 10

sum > 9 ✗
carry = 0

But 4? → add 10 (0110) to result

1 1
1010
+ 0110
0001 0000
1 0

Any:

sum > 9 → Invalid don't care cond

10 If we add 6 we get crt ans

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

1000
1001
10001

sum = 1 < 9 incorrect
F.C = 1 → add 6

10001
0110
00010111
1 7

$(17)_{10}$ ✓

Imp eg:- (57) + (26)

0101 0111
+ 0010 0110
0111 1101

7 < 9
13 > 9, carry = 0 → add 0110

0111 1101
+ 0110 [* for LSB's add 0110]
1000 0111
8 3 Ans

Shift ADD-3 Method:-

to convert Binary to BCD:

15 → 1111 → 0001 0101

Operations	Tens	Ones	(Decimal) Binary
original no			1111
shift		①1	111
shift		②11	11
shift		③111	1
Add-3		011	
		1010	
shift	0001	0101	→ no number → stop

Why 3 times?

If we do 3 times means then only we get, > 4.

2-4-2-1 code (BCD code): (2*4 2 1 code)

why ? to distinguish two 2's.

Decimal	LSB Set-1 2421	Set-2 2421
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	1011
6	0110	1100
7	0111	1101
8	1110	1110
9	1111	1111

$3 = \overset{2+2\,1}{0011}$ MSB X or $\overset{2\,4\,2\,1}{0010}$

$4 = \overset{2\,4\,2\,1}{1010}$ ↓ MSB X

self comp property of 2421 code

[self complementary is called reflective code]

$9 = \bar{0}$ | $6 = \bar{3}$

$8 = \bar{1}$ | $5 = \bar{4}$

$7 = \bar{2}$

eg: $(37)_{10} \rightarrow 2421$

3 7

0011 0111

ii) 0100 1100 1110

4 6 8

Decimal

Excess-3 code:-

(BCD code)

Decimal ⟶ 8-4-2-1 code $\xrightarrow{\text{Add } 0011}$ Excess-3

Decimal	BCD	Excess-3
0	0000	0000+0011 = 0011
1	0001	0001+0011 = 0100
2	0010	0010 +0011 = 0101
3	0011	0011 +0011 = 0110
4	0100	0100 + 0011 = 0111
5	0101	0101 + 0011 = 1000
6	0110	0110 + 0011 = 1001
7	0111	0111 + 0011 = 1010
8	1000	1000 + 0011 = 1011
9	1001	1001 + 0011 = 11000

Eg: (i) $(24)_{10}$ [* for every digit, you have add 0011]

0100 0011 → $(57)_{Ex-3}$

(ii) 5 7

0110 0101 1000 0011

1000 1000 1011

9 8 11

Double

Imp* Ex-3 code is the only unweighted code which is self complementing

* $w_3\ w_2\ w_1\ w_0$

$w_3+w_2+w_1+w_0 = 9$

$\neq 9$

2-4-2-1

$2+4+2+1 = 9$ ✓

$8+4+2+1 = 15 \neq 9$

$3-3-1-1 = 8$ ✓ < 9

$4+3+1+1 = 9$ ✓

Eg: i) 16
ii) 39
iii) 1208
iv) 5-2-1-1

} obtain excess-3

↓ Yes, = 9 self complimentary

1) 16 = 01001001
2) 39 = 01101100
3) 1208 = 0100 0101 0011 1011

Excess-3 code addition:

eg: $(2)_{10} + (5)_{10}$

	BCD	ex-3
$(2)_{10}$ →	0010 →	0101
$(5)_{10}$ →	0101 →	1000

→ 0101 → Ex-3
1000 → Ex-3
1101 → Ex-6
↓
wrong XD sub Ex-3
0 2
1101
−0011
1010 Ans
(10)

imp: $(27)_{10} + (39)_{10}$ +0011 +0011 Excess-3

$(27)_{10}$ → 0010 0111 → 0101 1010
$(39)_{10}$ → 0011 1001 → 0110 1100

G_2 ← 0 0 0 1 0 → Fc of G1 = 1
0101 1010 G1
Fc of G_2=0

G_1 = FC = 1 → we will add 3(0011) to it
G_2 = FC = 0 → we will sub 3(.

(+) 0110 1100
1110 0110
0 1 2 0 1
1110 0110
−0011 +0011
1001 1001
9 9

27
39
66
+33
99

Gray code: (Reflected Binary code) / cyclic code.

→ unweighted code
→ Unit distance code & minimum error code } other names
* → two successive values differ in only 1 bit
* Binary no is converted to Gray code to reduce switching o/p
used to error correction.

Decimal	Binary $b_3 b_2 b_1 b_0$	Gray code $g_3 g_2 g_1 g_0$
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	0000

similar last switching pair

only one time high switching speed

Binary to Gray code:

S1: Record MSB as it is

S2: Add the MSB to the next bit, record the sum & neglect the carry (X-OR)

S3: Repeat the process

Eg:- Convert 1011 to gray code.

(i) 1011

S1 1011

S2 1 1 1 0 ← 1+1

1+0 0+1

AB	Y
00	0
01	1
10	1
11	0

(ii) 1 1 1 0

1 0 0 1

$g_3 = b_3$

$g_2 = b_2 \oplus b_3$

$g_1 = b_1 \oplus b_2$

$g_0 = b_0 \oplus b_1$

(iii) 1 1 1 1

1 0 0 0

Gray code to Binary:

S1: Record the MSB as it is

S2: Add MSB to the next bit of Gray code, record the sum & neglect the carry

S3: Repeat the process

Eg: (i) 1 1 1 0

1 0 0 1

B to G

10 now G to B ↓

1 1 1 0

1 0 1 1

(ii) 1 0 0 1

1 1 1 0

$b_3 = g_3$

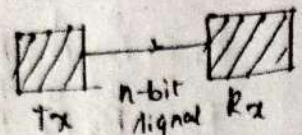
$b_2 = b_3 \oplus g_2$

$b_1 = b_2 \oplus g_1$

$b_0 = b_1 \oplus g_0$

What is parity:-

→ It is concept to detect errors
→ A "single bit error" is "detected" by it.
→ If there is error in 2 or more bits then it donot going to detect error.

→ 

we send a bit n-bit stream & also an extra bit (tells total 1's)

Eg 0100 → odd no of 1
0101 → even no of 1

two types of parity < Even / Odd.

* Even parity

original signal + parity bit
0100 + 1
1100 + 0
↗ even no of 1's unnai kabbati parity bit = 0

0100 + 1 = 0101
original signal +1 = 0101 → receiver detect (but received odd, so there is an error)

we sent even parity (transmitted)

* odd parity

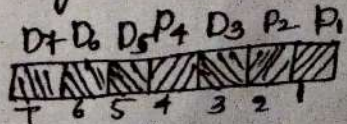
original signal + parity bit
0100 + 0
1100 + 1
↗ even no of 1's unnai kabbati parity bit = 1

Hamming code:-

→ easy to implement
→ 7-bit hamming code is used commonly.

Data bits - 4 } 7-bit hamming code
parity bits - 3

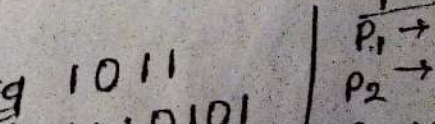
→ 2^n { where n=0,1, ... n }



for where parity bits should lie.

$2^0 = 1$
$2^1 = 2$

Eg 1011
↳ 1010101



even 0 1 0
$P_1 \rightarrow D_3 D_5 D_7$
$P_2 \rightarrow D_3 D_6 D_7$
$P_4 \rightarrow D_5 D_6 D_7$

$P_2 = D_3 D_6 D_7$
$= 1\ 0\ 1$
$P_2 = 0$

$P_1 = D_3 D_5 D_7$
$= 1\ 1\ 1$

$P_4 = D_5 D_6 D_7$
$= 1\ 0\ 1$
$P_4 = 0$ for even parity → $P_1 = 1$

eg: If the 7-bit hamming code word received by a receiver is 1011011. Assuming the even parity. (Imp) State whether the received code word is crt or wrong. If wrong locate the bit having error.

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1	1	0	1	1

Sol: 1 0 1 1 0 1 1

D_5 D_6 D_7 P_4

1 1 0 1 → total no. of 1's P_4 3 odd

$P_4 = 1$

for even parity 10 wrong, odd parity error.

$P_1 = D_3 D_5 D_7$ ✗

$P_2 = D_3 D_6 D_7$ ✗

$P_4 = D_5 D_6 D_7$ ✗

~~$P_4 = 0$~~ ↳ *If we get odd parity, but mentioned even parity then just Parity bit = 1

Why we get ep, but min op, Parity bit = 1

P_2 D_3 D_6 D_7 2 no. of 1's

1 0 0 1 → even parity

$P_2 = 0$ no error

P_1 D_3 D_5 D_7

1 0 1 1 → 3 1's odd parity

$P_1 = 1$

P_4 P_2 P_1

$(1\ 0\ 1)_2 = (5)_{10}$

So crt code = 1001011

Logic gates

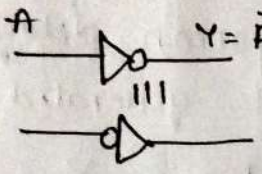
Logic gates: It is a physical device which performs logic operation on one or more logical inputs, & produces a single logical output. [Inversion, logical multiplication, logical sum etc...]

(i) Basic gates : NOT, AND, OR [Because we can implement any logic by using these gates]

(ii) Universal gates : NAND, NOR [Because, by using only NAND or only NOR we can implement and digital system]

(iii) Arithmetic gates : X-OR, X-NOR [Because, we use them in arithmetic gate]

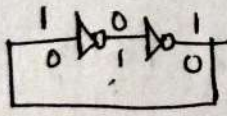
NOT Gate:-



Truth table:

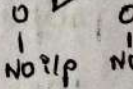
↳ table having o/p from all possible combinations of i/p's

A	Y
0	1
1	0

a) 

→ 1 if i/p = 1
0 if i/p = 0

gate

~~a)~~ Buffer [i/p = o/p] → 

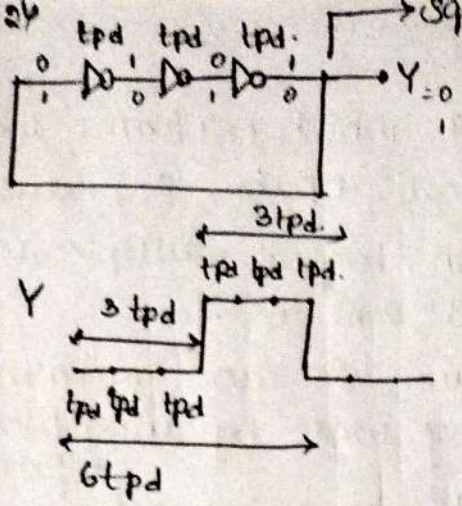
because of feedback it is not buffer

b) Astable MV → [Astable, in which the ckt is not stable in either state, it continuously switches from one state to other]

c) Bistable MV

d) Sq wave gen → [Bistable, in which the ckt is stable in either state, It can be flipped from one state to the other by an external trigger pulse.]

06-24



Sq. wave generator / Astable MV / Clk gen / ring oscillator

tpd = propagation delay

[to get o/p we need some time that delay is called as propagation delay]

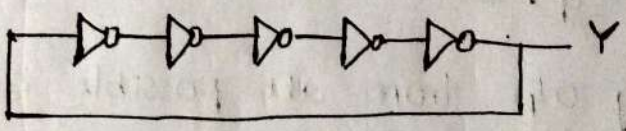
time period = the time that signal / function repeats the same

$= 6t_{pd}$

$$\boxed{T = 2nt_{pd}}$$

n: no. of NOT gates

3) In the ckt shown, the propagation delay of each NOT gate is 100 psec. Then freq of generated square wave is



$T = 2 \times 5 \times 100 \times 10^{-12}$ (psec)

$T = 1000 \times 10^{-12}$

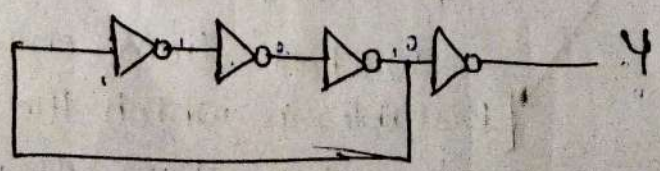
$T = 10^{-9}$

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

$f = \frac{1}{10^{-9}}$

$f = 10^{9} \Rightarrow f = 1\,GHz$.

4) In the circuit shown, the propagation delay of each NOT gate is 8nsec, then the time period of generated square wave is

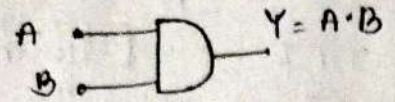


$T = 2 \times 3 \times 2$ nsec

$T = 12$ nsec ✱ . {only considered within FB}

AND:- output is high (1) when all the inputs are high

2 i/p AND :-

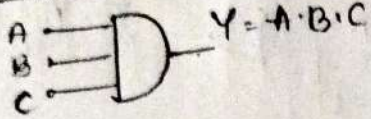


T.T :-

AB	Y
00	0
01	0
10	0
11	1

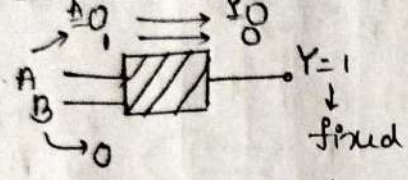
* Associative law:
  $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
* Commutative law:
  $A \cdot B = B \cdot A$

3 i/p AND :-

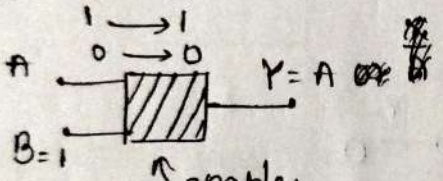


ABC	Y
000	0
001	0
010	0
011	0
100	0
101	0
110	0
111	1

→ Enable & Disable:

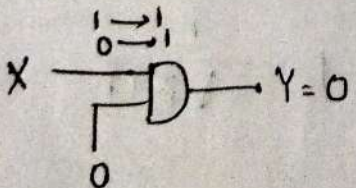


here 'B' acts as disable

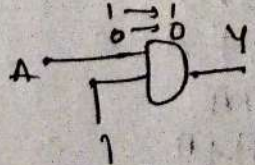


	A	B	Y
disable o/p=0	0	0	0
	0	1	0
enable buffer	1	0	1
	1	1	1

"0" → disable with o/p = 0



"1" enable / buffer

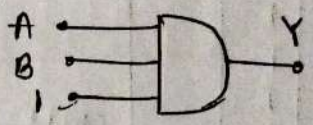


→ Unused i/p:

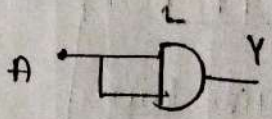
Vimp ** In TTL logic, if any i/p is open or floating it will act as "1"

E " ECL " it will act as "0"

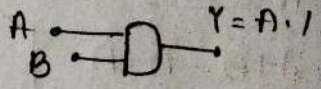
(i)



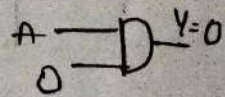
(ii)



(iii) TTL logic

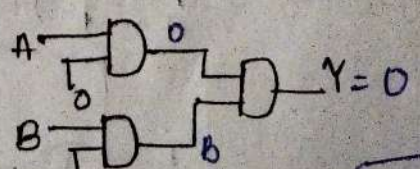


ECL: Disable.

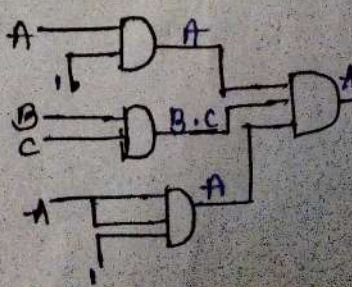


HW:

1)



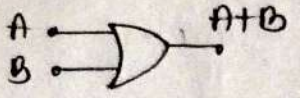
2)



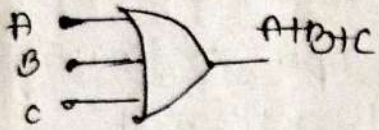
for TTL: (unused i/p = 1)
F = A·B·C

for ECL (unused = 0)
F = 0

OR:- O/p high, if any one or all inputs are high



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



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

* Associative law:

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

* Commutative law:

$A+B = B+A$

Enable & Disable

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

Disable

here "B" acts as enable if A=0 & "B" acts as disable if A=1

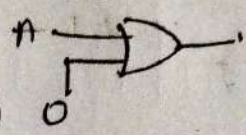
unused i/p

TTL → '1' [open or floating i/p] (ii)

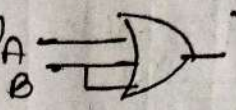
ECL → '0' (i)

* connect to B

TTL=1
ECL=0

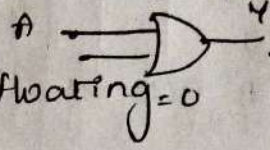


(i)

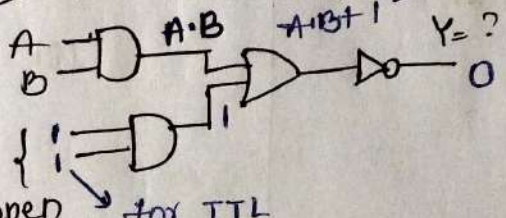


(ii)

(iii) ECL logic:

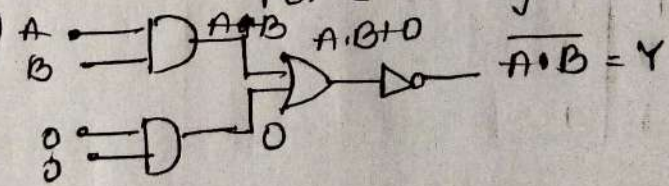


gate: 1) For TTL logic

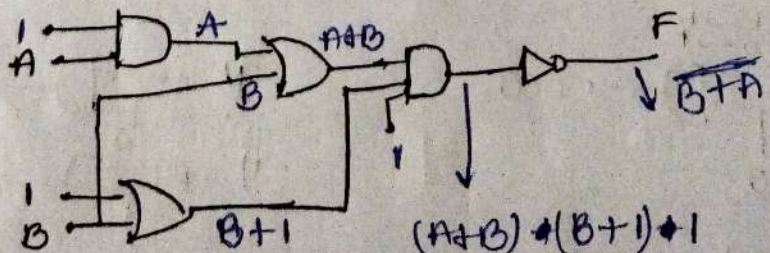


for ECL logic

HW) 1)



2) for TTL logic



$= (A+B) \cdot (B+1) \cdot 1$

$= (A+B) \cdot (B+1)$

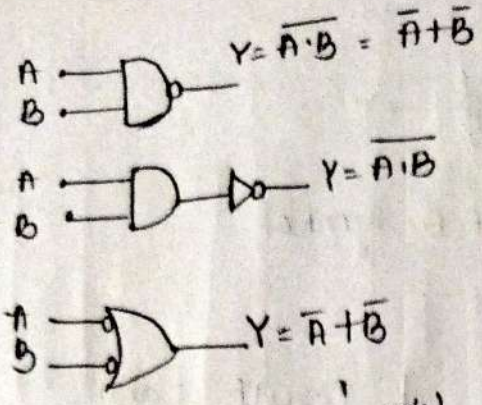
$= AB + BB + A + B$

$= AB + B + A$

$= B(A+1) + A$

$= B + A$

NAND Gate:-



(a/c i/p's)

NAND gate = Bubble OR gate

T.T:

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

disable o/p = 1

enable inverter

Associative law: x

$\overline{(\overline{AB})\bar{C}} = \overline{A(\overline{BC})}$

$\overline{\overline{AB}} + \bar{C} = \bar{A} + \overline{\overline{BC}}$

$AB + \bar{C} = \bar{A} + BC$

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

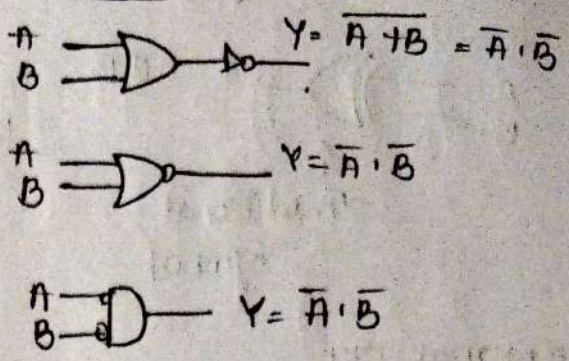
Commutative law:-

$\overline{AB} = \overline{BA}$

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

unused i/p: same as AND gate

NOR Gate:



NOR gate = bubble AND

T.T:

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

o/p

enable inverter

disable

Associative law:x

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

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

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

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

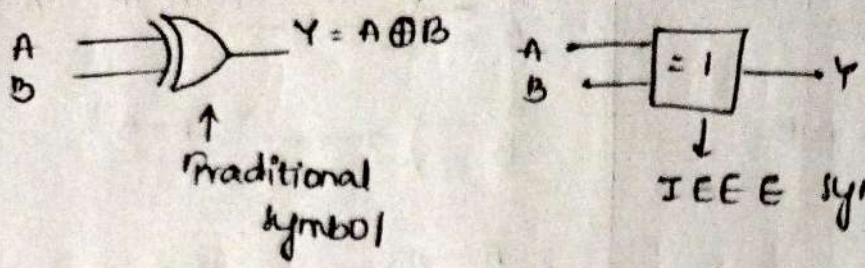
Commutative law:

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

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

unused i/p: same as OR gate

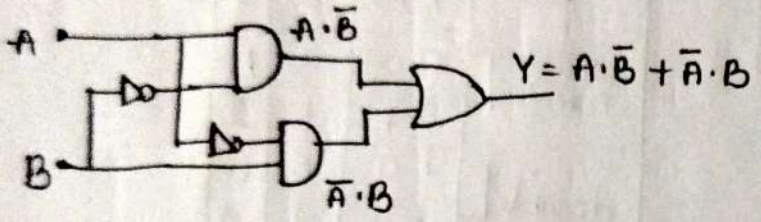
Exclusive OR gate : EX-OR / X-OR :-



Internal CKt

$Y = A \oplus B$

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



Truth table

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

* ODD is Detector

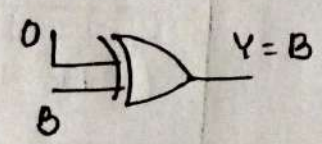
Enable & disable

A = 0 Enable buffer

A = 1 Enable inverter

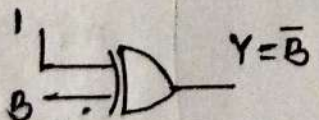
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Controlled inverter



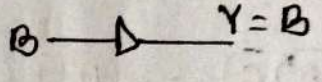
$0 \to 0$

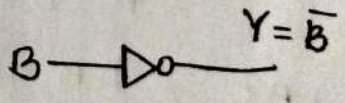
$1 \to 1$



$0 \to 1$

$1 \to 0$





properties:

1) (i) $A \oplus A = 0$

(ii) $A \oplus \bar{A} = 1$

(iii) $A \oplus 0 = A$

(iv) $A \oplus 1 = \bar{A}$

2) If $A \oplus B = C$ then

$B \oplus C = A$

$A \oplus C = B$

$A \oplus B \oplus C = 0$

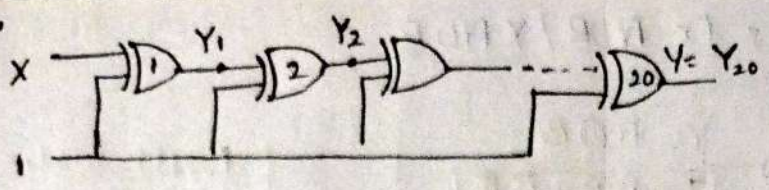
iii) $A \oplus A = 0$

$A \oplus A \oplus A$

$0 \oplus A = A$

$A \oplus A \oplus A \oplus \cdots n = A$, n = odd

$= 0$, n = even

Q) 

Cascading of 20 XOR gates
Y=?

Sol: $Y_1 = X \oplus 1$ | $Y_2 = \bar{X} \oplus 1$ | $Y_3 = \bar{X}$

$Y_1 = \bar{X}$ | $Y_2 = X$ | $Y_4 = X$

$Y_{odd} = \bar{X}$

$\boxed{Y_{even} = X = Y_{20}}$

Q) $Y = A \oplus \bar{A} \oplus \bar{A} \oplus A \oplus A \oplus \bar{A} \oplus \bar{A} \oplus A \oplus A$

$1 \oplus 1 \oplus 1 \oplus 1 \oplus A$

$\Rightarrow 0 \oplus 0 \oplus A$

$= \boxed{A}$

Q) $f(A,B) = A \oplus B$

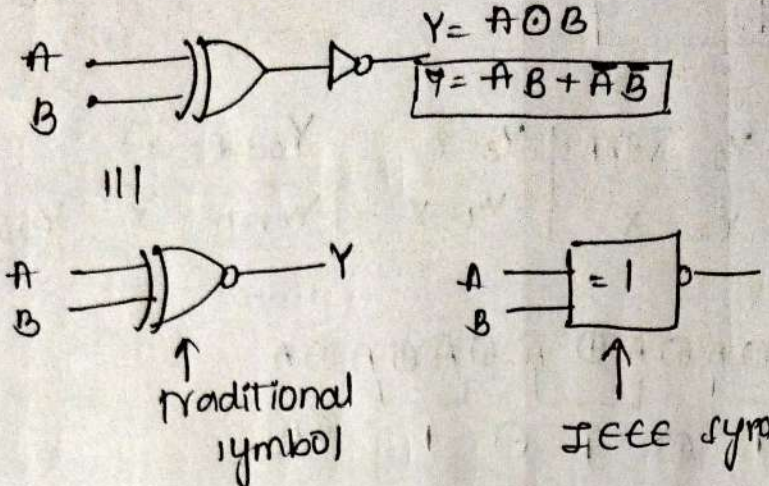
simplified form of the func. $f(f(x \oplus y, z), w) = ?$

Sol: $f(x \oplus y, z) = (x \oplus y) \oplus z$

$= x \oplus y \oplus z$

$f(f(x \oplus y, z), w) = x \oplus y \oplus z \oplus w$ //

Exclusive NOR gate : Ex-NOR / XNOR



Truth table

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

When A = B
o/p = 1
When A ≠ B
o/p = 0

Same i/p detector

Enable & disable:

	A	B	Y
enable inverter	0	0	1
	0	1	0
enable buffer	1	0	0
	1	1	1

Properties

1) (i) $A \odot A = 1$
$A \odot \bar{A} = 0$
$A \odot 0 = \bar{A}$
$A \odot 1 = A$

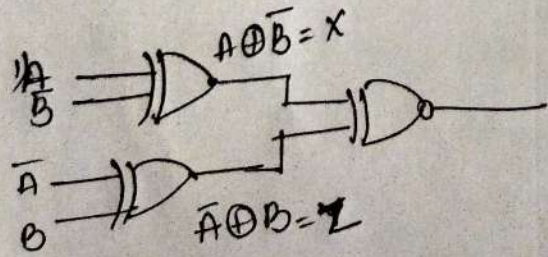
2) $A \odot A = 1$
$A \odot A \odot A = 1$
$1 \odot A = A$
$\Rightarrow A \odot A \odot A \odot \cdots n = A$, n = odd
1, n = even

3) ✱ Imp
EXOR & EX-NOR are compliment → even i/p
EXOR & EX-NOR are same → odd i/p

$A \oplus B \oplus C = A \odot B \odot C$ → odd i/p

$A \oplus B \oplus C \oplus D = \overline{A \odot B \odot C \odot D}$ → even i/p

HW:



$Y = X \odot Z$
$Y = (A \oplus \bar{B}) \odot (\bar{A} \oplus B)$
$= (A \oplus \bar{B})(\bar{A} \oplus B) + \overline{(A \oplus \bar{B})}\,\overline{(\bar{A} \oplus B)}$
$= (AB + \bar{A}\bar{B})(\bar{A}\bar{B} + AB) + \overline{(AB + \bar{A}\bar{B})}\,\overline{(\bar{A}\bar{B} + AB)}$
$= (AB + A\bar{B}) + (\overline{AB + \bar{A}\bar{B}})$
$= 1$

$(x \cdot x = x)$ $(x + \bar{x} = 1)$

a) $Y = A \odot A \odot A \odot A \odot A \odot \bar{A} \odot A$
$1 \odot 1 \odot 0 \odot A$
$1 \odot \bar{A} = \bar{A}$

NAND as universal gate :-

→ AND, OR & NOT are sufficient to implement any digital system.

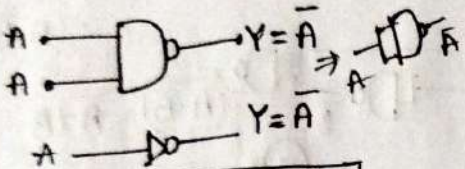
→ If we can convert NAND or NOR to these three, we can say that any ckt can be implemented

NAND as NOT :

$Y = \overline{A \cdot B}$

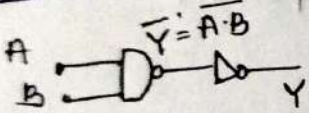
$Y = \overline{A \cdot A}$ if i/p's are same

$Y = \overline{A}$

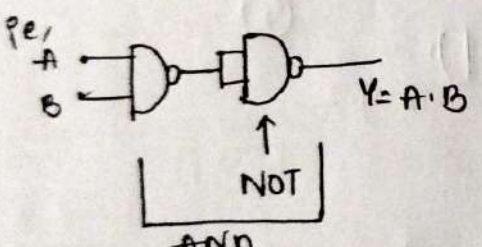


1 NAND Gate

NAND as AND : (complement o/p)



ie,

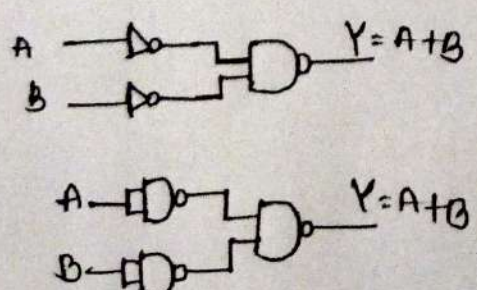


2 NAND GATE

NAND as OR : (complement o/p)

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

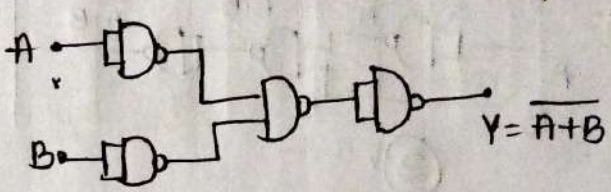
OR operation, $Y = A + B$



3 NAND Gate

NAND as NOR :

→ Complementing NAND as OR o/p to get NOR



4 NAND Gates

NAND as EXOR :-

imp

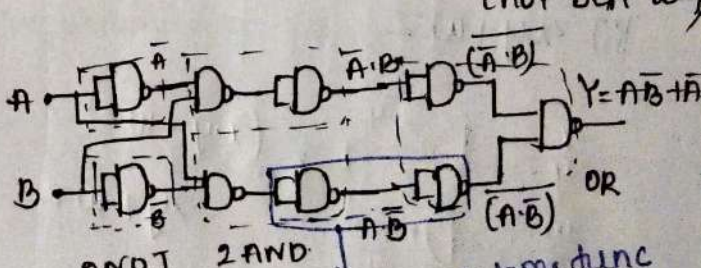
$Y = A\bar{B} + \bar{A}B$

2 NOT → 2 NAND gates

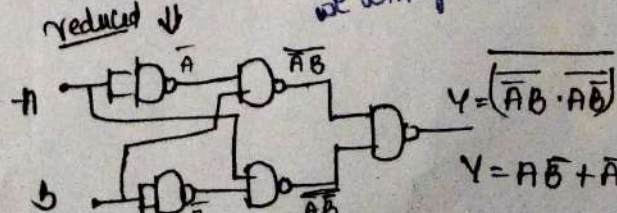
2 AND → 4 NAND gates

1 OR → 3 NAND gates

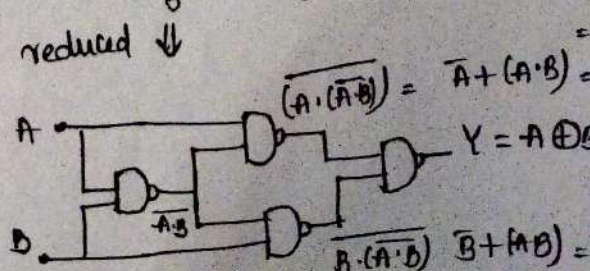
total 9 NAND (not best way)



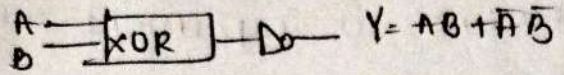
reduced ⇓

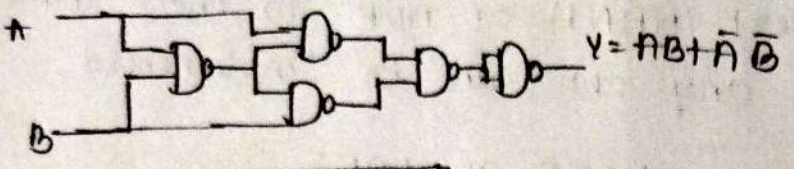


reduced ⇓



NAND & EXNOR :-



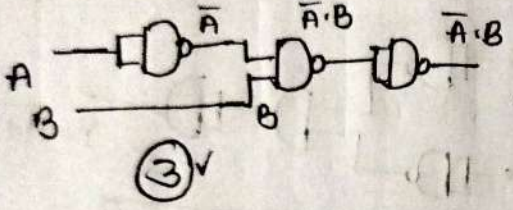


5 NAND Gate

Gate	No of NAND req
NOT	1
AND	2
OR	3
NOR	4
EXOR	4
EXNOR	5

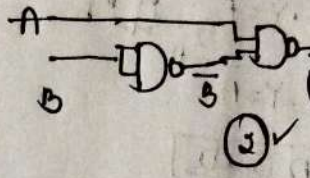
Q) Min no. of 2 i/p NAND gate required to implement $Y = \bar{A} \cdot B$

NOT:



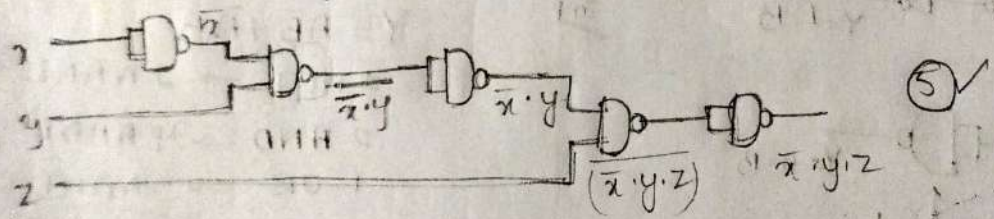
$Y = \bar{A} + B = \overline{\bar{A} + B} = \overline{(\bar{A} \cdot \bar{B})}$

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

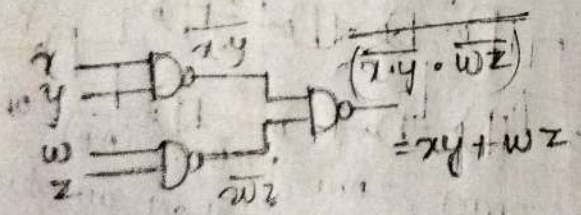


Q) Min. of 2 i/p NAND gate req. to implement

(i) $\bar{x} \cdot y \cdot z$



(ii) $xy + wz$



NOR gate:-(as universal gate)

NOR as NOT:-

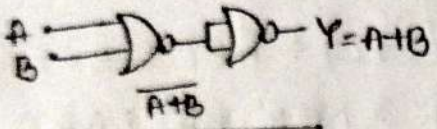
$Y = \overline{A+B}$

$Y = \overline{A+A}$

$Y = \overline{A}$

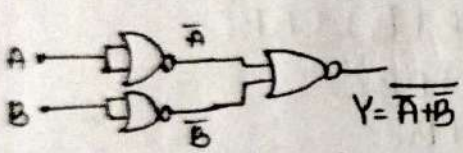
1 NOR gate

NOR as OR:-



2 NOR gates

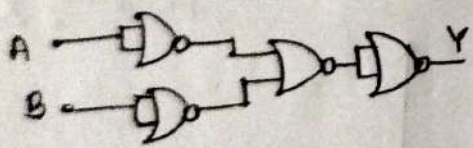
NOR as AND



3 NOR gate

NOR as NAND:-

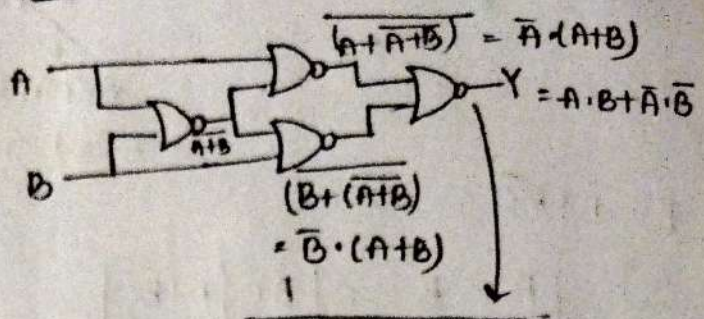
$Y = \overline{A \cdot B}$



4 NOR gates

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

NOR as XNOR $Y = A \cdot B + \bar{A} \cdot \bar{B}$



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

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

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

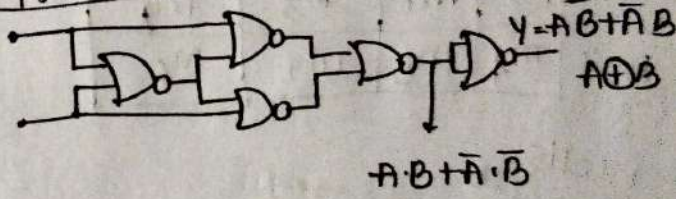
4 NOR gate $= A \cdot B + \overline{(A+B)}$.

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

NOR as XOR:-

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

5 NOR gates



Q4 No. of 2 i/p NOR gates req to implement (i) $Y = (\bar{A}+B) \cdot (\bar{C}+D)$

K-Map - (Karnaugh Map):-

→ for logic minimisation < Boolean Algebra → result is not always minimum because of rules & formula not perfect result & unique
K-MAP → minimum, But result is not same

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

Q.1:

	A	B	C	$B\bar{C}$	$A+B\bar{C}$	
m_0	0	0	0	0	0	0
m_1	0	0	1	0	0	1
m_2	0	1	0	1	1	2
m_3	0	1	1	0	0	3
m_4	1	0	0	0	1	4
m_5	1	0	1	0	1	5
m_6	1	1	0	1	1	6
m_7	1	1	1	0	1	7

must be change = 1 bit only

	BC 00	BC 01	BC 11	BC 10
A=0	m_0	m_1	m_3	m_2
A=1	m_4	m_5	m_7	m_6

Boolean algebra

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

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

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

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

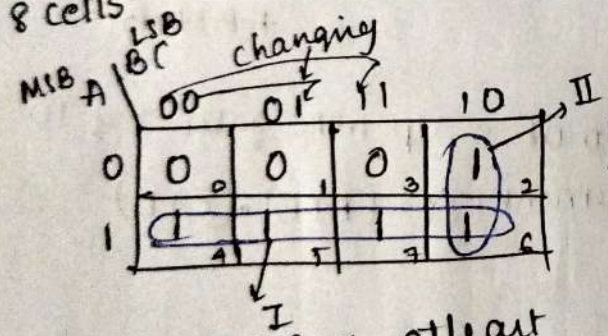
$= AB + A\bar{B} + \bar{A}B\bar{C}$

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

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

$= A + B\bar{C}$

8 cells



While pairing atleast one new cell must be there

Pairing rule

2 '1's
4 '1's
8 '1's
16 '1's

priority order high to low
16 → 8 → 4 → 2

* Diagonal pairing X

$F = I + II$ (implicants)

F = non changing variables

$F = A \cdot 1 \cdot 1 + 1 \cdot B\bar{C}$

A changing

B1 is changing

$B\bar{C}$ not changing (10)

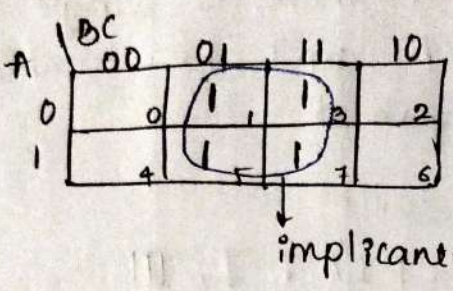
eg 1: $f(A,B,C) = \Sigma m(1,3,5,7)$ (M1, M3, M5, M7)

MSB — LSB — SOP

(i) Find out no. of variables = $n = 3$ (A, B, C)

(ii) No. of cells in K-map = $2^n = 2^3 = 8$

Sol

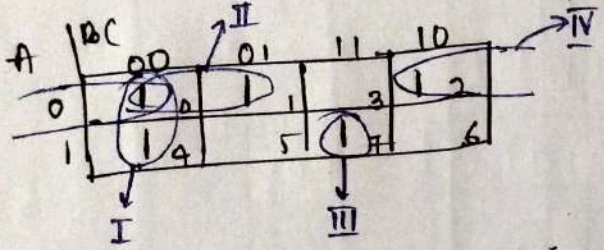


implicant - I

$f = C$ ✓

eg 2: $f(A,B,C) = \Sigma m(0,1,2,4,7)$

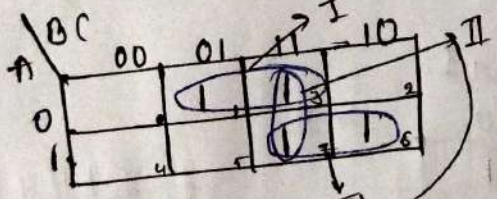
Sol: $n = 3$, 8 cells



$F = I + II + III + IV$

$f = \bar{B}\bar{C} + \bar{A}\bar{B} + \bar{A}\bar{C} +$ ABC ✓ Any

eg 3 $F(A,B,C) = \Sigma m(1,3,6,7)$



$f = I + II + III$

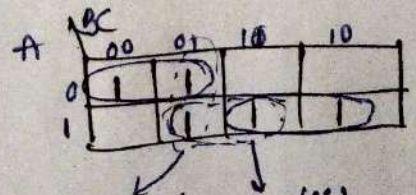
$= \bar{A}C + AB + \cancel{BC}$ ↓ redundant

redundant because no new cell while pairing.

Redundancy theorm

- 3 variable
- each variable repeated twice
- 1 variable is complemented

eg: $F(A,B,C) = \Sigma m(0,1,5,6,7)$



Case i: $f = A\bar{B} + \bar{A}\bar{B} + AC$

Case ii: $f = A\bar{B} + \bar{A}\bar{B} + \bar{B}C$

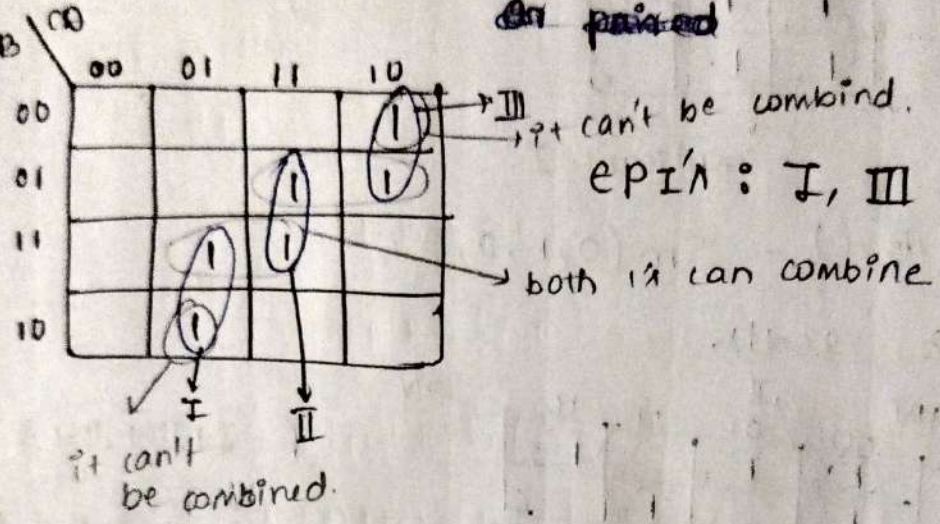
Fine result is min but not same values

Implicants:- The group of 1's is called as implicants
eg: 1, 2, 4, 8, 16 - - - etc.

Prime Implicants:- It is the largest possible group of 1's in K-map

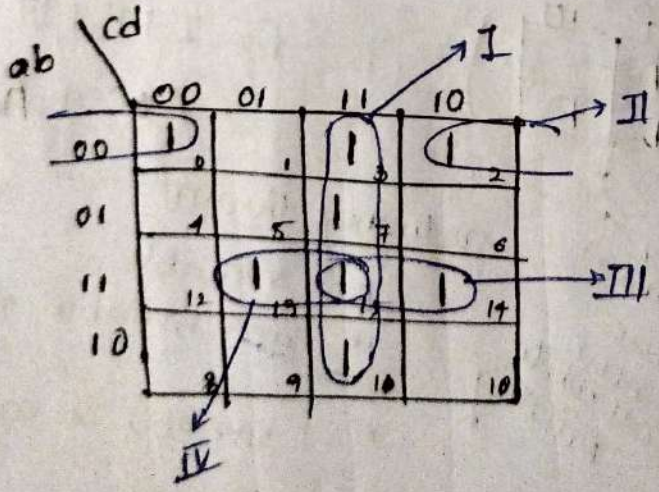
Essential prime implicants:- Atleast there is single one which can't be combined
(EPI) ~~or paired~~

eg :-



Eg: 4-Map with 4 variables :

$f(a,b,c,d) = \Sigma m(0,2,3,7,11,13,14,15)$

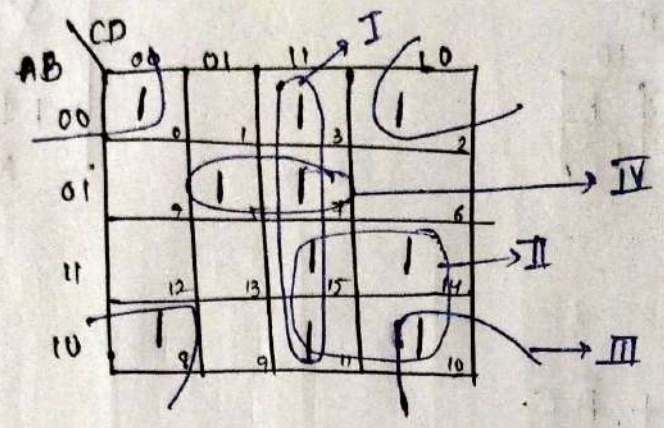


$f = I + II + III + IV$

$f = cd + abd + abc + \bar{a}\bar{b}\bar{d}$

eg(ii) $f=(A,B,C,D) = \Sigma m(0,2,3,5,7,8,10,11,14,15)$

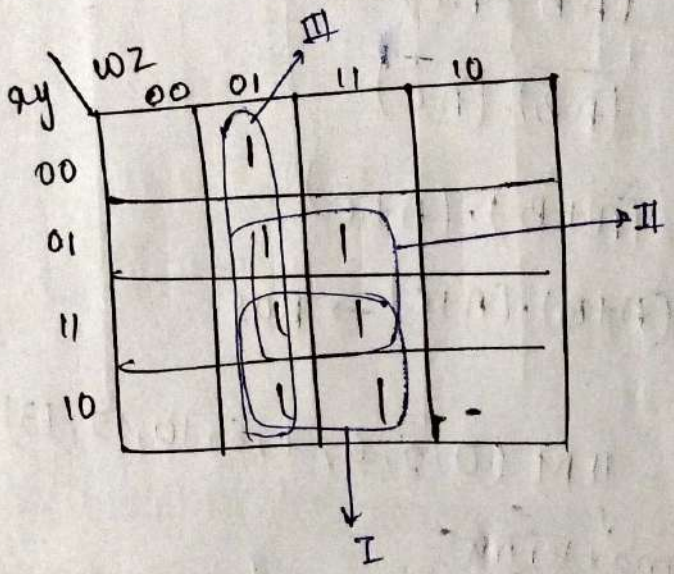
sol:



$f = I + II + III + IV$

$f = CD + AC + \bar{A}B\bar{D} + \bar{B}\bar{D}$

eg(iii) $f(w,x,y,z) = \Sigma m(1,5,7,9,11,13,15)$

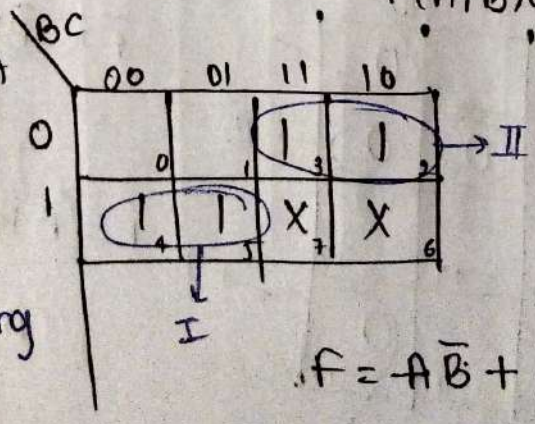


$f = \bar{w}z + yz + xz$

Don't care in K-MAP:-

$f(A,B,C) = \Sigma m(2,3,4,5) + \Sigma d(6,7)$

(i)



without ~~without~~ considering 'X'

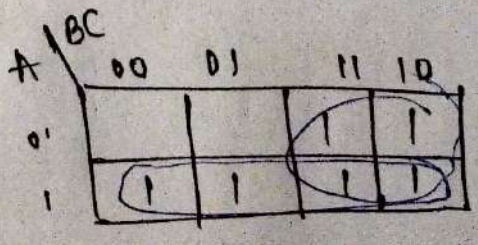
'dontcare can be '0' or '1'

max

min

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

(ii) ✓



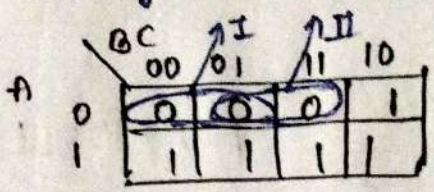
with $X = 1$

$f = A + B$

K-Map using Max terms: (grouping 0's)

* Min terms
  X = 1
* Max terms
  X = 0

eg1:



~~F = $\bar{A}\bar{B} + A\bar{B}$~~

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

↳ Apply demorgans law

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

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

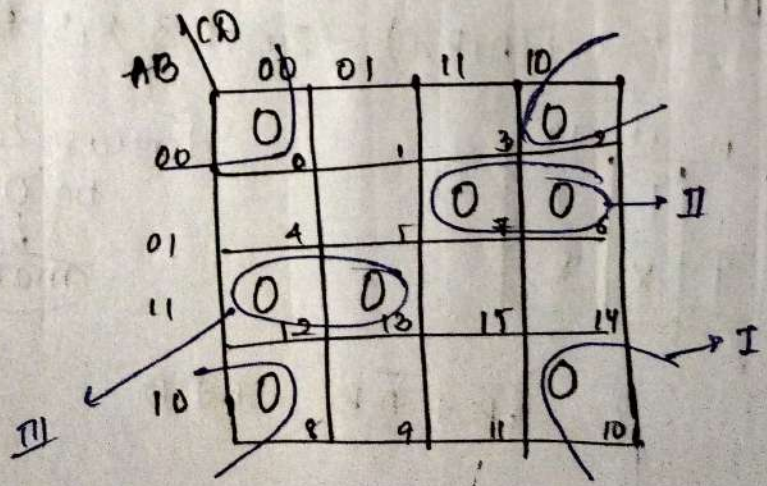
$= (\bar{\bar{A}} + \bar{\bar{B}}) \cdot (\bar{\bar{A}} + \bar{C})$

$F = (A+B) \cdot (A+\bar{C})$ → POS

eg2:

$f(A,B,C,D) = \Pi M\,(0,2,6,7,8,10,12,13)$

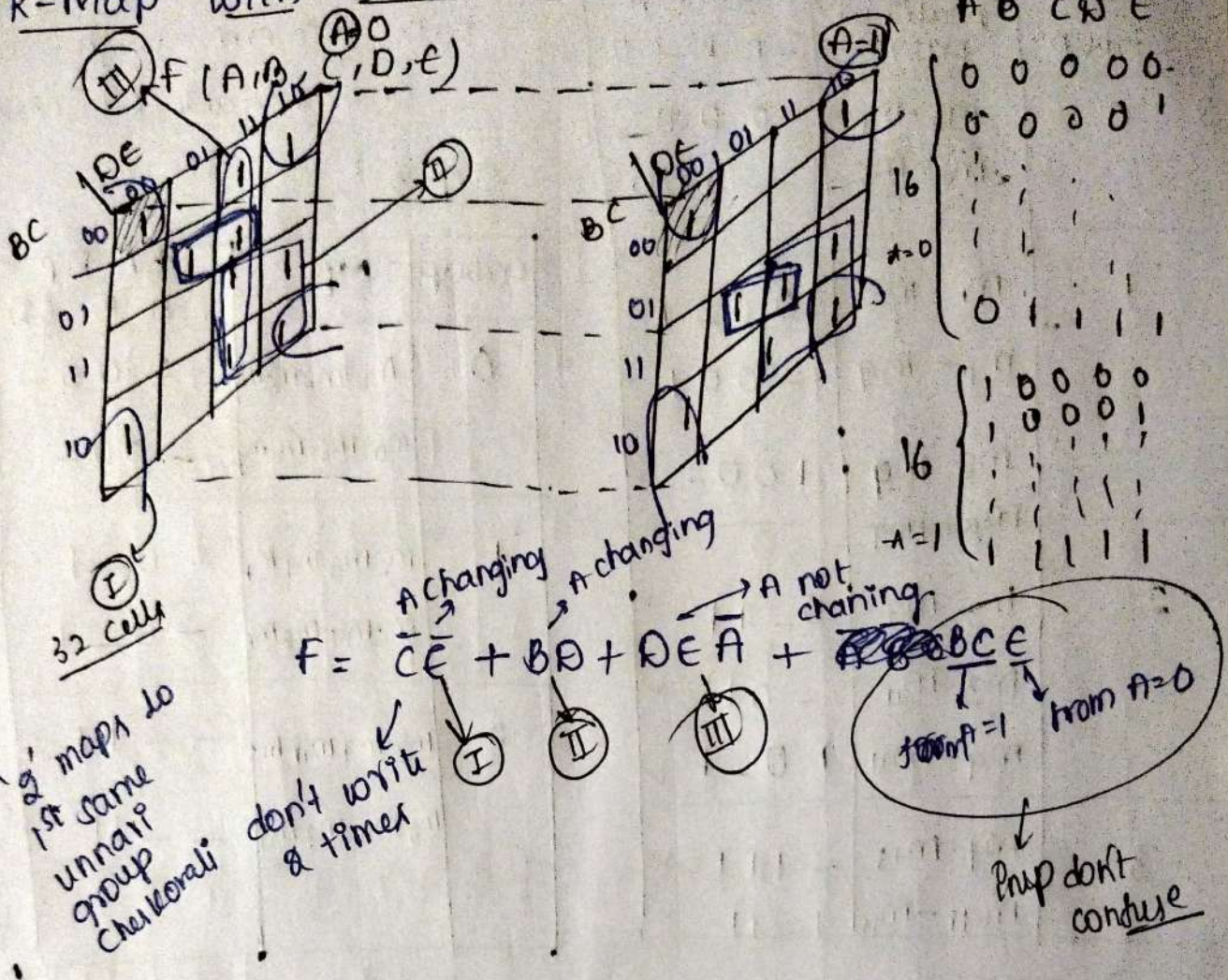
↓ max terms



$\bar{F} = \bar{B}\bar{D} + AB\bar{C} + \bar{A}BC$

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

K-Map with 5-variable.



A B C D E
0 0 0 0 0
0 0 0 0 1
⋮
0 1 1 1 1
1 0 0 0 0
1 0 0 0 1
⋮
1 1 1 1 1

Quine - McCluskey Minimization Technique (tabular method)

↳ for more variables → for complex designs

eg: $Y(A,B,C,D) = \Sigma m(0,1,3,7,8,9,11,15)$

A B C D
0 - 0000
1 - 0001
3 - 0011
7 - 0111
8 - 1000
9 - 1001
11 - 1011
15 - 1111

Step 1:-

Group	minterm	Bin Rep A B C D
0	m0	0000 ✓
1	m1 m8	0001 ✓
		1000 ✓
2	m3, m9	0011 ✓
		1001 ✓
3	m7, m11	0111 ✓
		1011 ✓
4	m15	1111 ✓

Step-2:

Group	matched pair	Bin-Rep A B C D
0	$m_0 - m_1$	0 0 0 _
	$m_0 - m_8$	_ 0 0 0
1	$m_1 - m_3$	0 0 _ 1
	$m_1 - m_9$	_ 0 0 1
	$m_8 - m_9$	1 0 0 _
2	$m_3 - m_7$	0 _ 1 1
	$m_3 - m_{11}$	_ 0 1 1
	$m_9 - m_{11}$	1 0 _ 1
3	$m_7 - m_{15}$	_ 1 1 1
	$m_{11} - m_{15}$	1 _ 1 1

compare n^{th} & $(n+1)^{th}$ group take minterm with one-variable/bit change

Step 3:

Group	MP	Bin-Rep A B C D	
0	$m_0 - m_1 - m_8 - m_9$	_ 0 0 _	$\bar{B}\bar{C}$
	$m_0 - m_8 - m_1 - m_9$	_ 0 0 _	
1	$m_1 - m_3 - m_9 - m_{11}$	_ 0 _ 1	$\bar{B}D$
	$m_1 - m_9 - m_3 - m_{11}$	_ 0 _ 1	
2	$m_3 - m_7 - m_{11} - m_{15}$	_ _ 1 1	CD
	$m_3 - m_{11} - m_7 - m_{15}$	_ _ 1 1	

further no matched pair

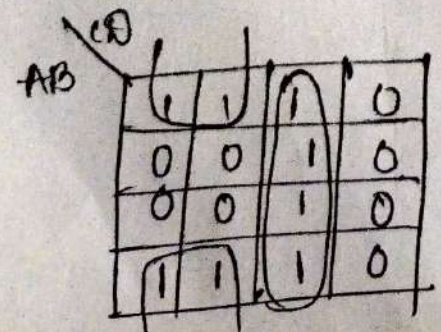
Prime implicant table:-

EPI	P-I	minterms involved	0	1	3	7	8	9	11	15
		(given minterms)								
EPI	$\bar{B}\bar{C}$	0, 1, 8, 9	(X)	X			(X)	X		
not EPI	$\bar{B}D$	1, 3, 9, 11		X	X			X	X	
EPI	CD	3, 11, 7, 15			X	(X)			X	(X)

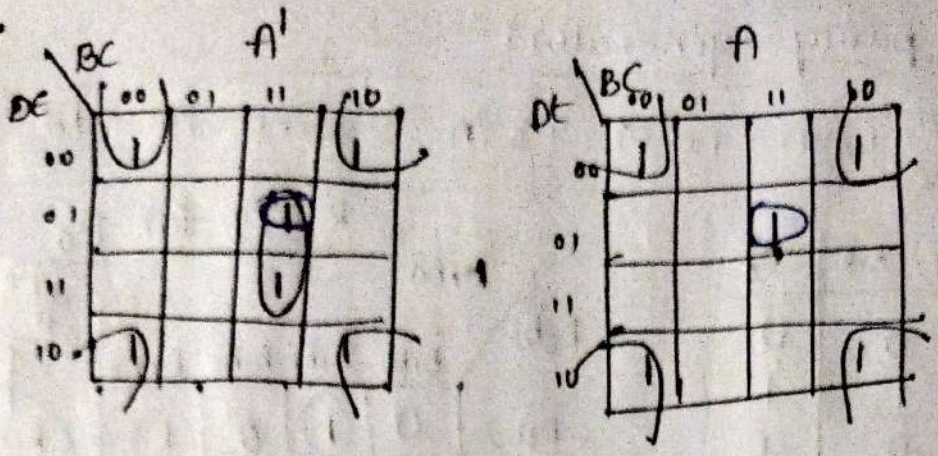
circle alone

EPI = essential prime implicant

to verify $Y = \bar{B}\bar{C} + CD$



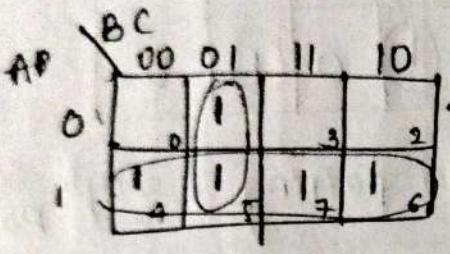
Q6



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

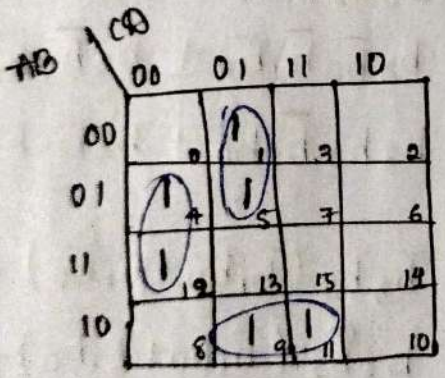
Q7

SOP of the func $f = A + \bar{B}C$ is



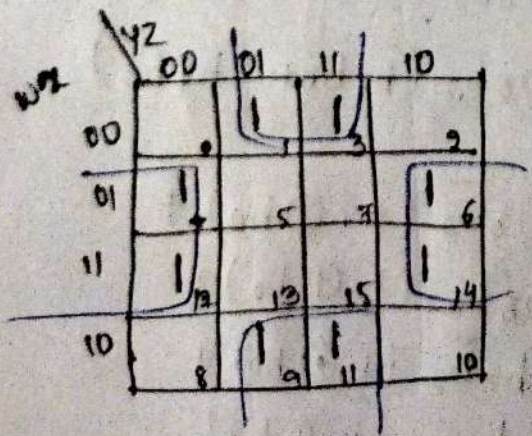
$$f = \Sigma m(1,4,5,6,7)$$

Q8 The switching expression corresponding to $f(A,B,C,D) = \Sigma(1,4,5,9,11,12)$ is



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

Q9 $f(w,x,y,z) = \Sigma(1,3,4,6,9,11,12,14)$ → independent of how many variables ↓ a variable



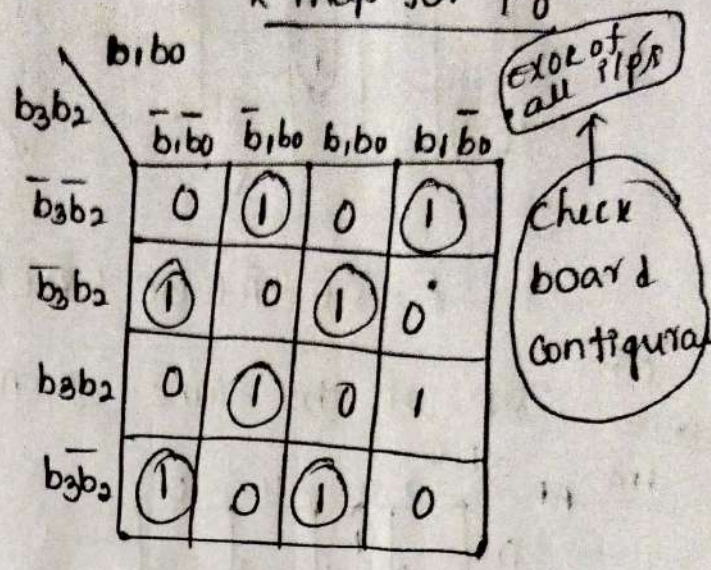
$$f = \bar{x}z + x\bar{z}$$

4-bit even parity generator:-

Parity is '1' when no. of 1's in sequence is odd.

b_3	b_2	b_1	b_0	P_0
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

K-map for P_0



no pairing, single cell pair.

$$P_0 = \bar{b}_3\bar{b}_2\bar{b}_1b_0 + \bar{b}_3\bar{b}_2b_1\bar{b}_0 + \bar{b}_3b_2\bar{b}_1\bar{b}_0 + \bar{b}_3b_2b_1b_0 + b_3\bar{b}_2\bar{b}_1\bar{b}_0 + b_3\bar{b}_2b_1b_0 + b_3b_2\bar{b}_1b_0 + b_3b_2b_1\bar{b}_0$$

$$P_0 = b_1 \oplus b_0 (\bar{b}_3\bar{b}_2 + b_3b_2) + b_1 \odot b_0 (b_3\bar{b}_2 + \bar{b}_3b_2)$$

$$P_0 = b_1 \odot b_0 (b_3 \odot b_2) + b_1 \odot b_0 (b_3 \oplus b_2)$$

$$P_0 = (b_1 \overset{A}{\oplus} b_0)(\overline{b_3 \overset{B}{\oplus} b_2}) + \overline{(b_1 \overset{A}{\oplus} b_0)}(b_3 \overset{B}{\oplus} b_2)$$

$$P_0 = A\bar{B} + \bar{A}B$$

$$P_0 = A \oplus B$$

$$P_0 = (b_1 \oplus b_0) \oplus (b_3 \oplus b_2)$$

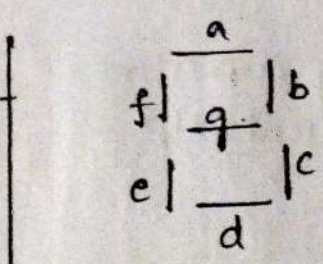
$$P_0 = b_0 \oplus b_1 \oplus b_2 \oplus b_3$$

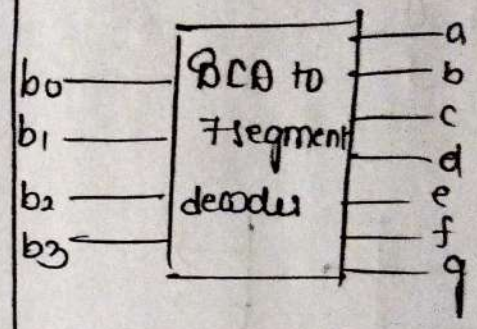
$$P_0 = \bar{b}_3\bar{b}_2(\bar{b}_1b_0 + b_1\bar{b}_0) + \bar{b}_3\bar{b}_2(\bar{b}_1b_0 + b_1b_0) + b_3\bar{b}_2(\bar{b}_1b_0 + b_1b_0) + b_3b_2(\bar{b}_1b_0 + b_1\bar{b}_0)$$

$$P_0 = \bar{b}_3\bar{b}_2(b_1 \oplus b_0) + b_3b_2(b_1 \oplus b_0) + \bar{b}_3\bar{b}_2(b_1 \odot b_0) + b_3\bar{b}_2(b_1 \odot b_0)$$

Seven Segment Display decoder : (0 to 9)

b_3	b_2	b_1	b_0	a	b	c	d	e	f	g	rep
0	0	0	0 (0)	1	1	1	1	1	1	0	0
0	0	0	1 (1)	0	1	1	0	0	0	0	1
0	0	1	0 (2)	1	1	0	1	1	0	1	2
0	0	1	1 (3)	1	1	1	1	0	0	1	3
0	1	0	0 (4)	0	1	1	0	0	1	1	4
0	1	0	1 (5)	1	0	1	1	0	1	1	5
0	1	1	0 (6)	1	0	1	1	1	1	1	6
0	1	1	1 (7)	1	1	1	0	0	0	0	7
1	0	0	0 (8)	1	1	1	1	1	1	1	8
1	0	0	1 (9)	1	1	1	1	0	1	1	9
1	0	1	0	1	1	1	0	1	1	1	
:	:	:	:	} Don't care.							
1	1	1	1								

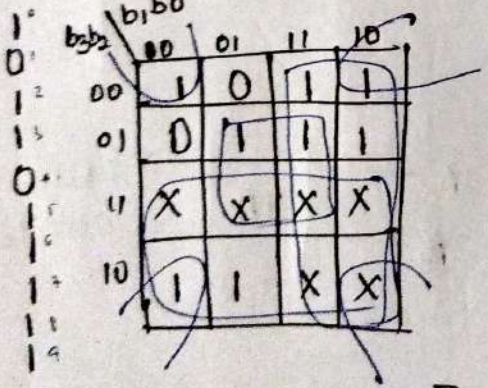




for 14 | 14

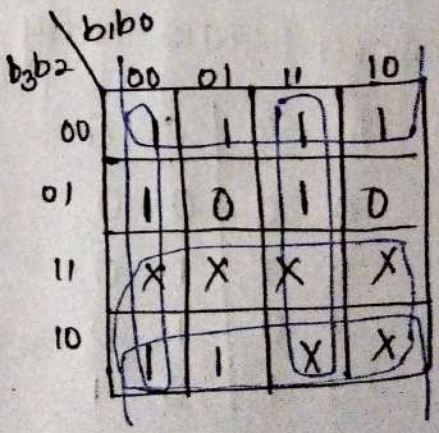
2 seven segment display decoder we need

Now, for a



$$a = b_3 + b_1 + \bar{b}_2\bar{b}_0 + b_0 b_2$$

for b



$$b = b_3 + b_1b_0 + \bar{b}_2 + \bar{b}_1\bar{b}_0$$

Ny,

$$c = b_3 + b_2 + \bar{b}_1 + b_0$$

$$d = b_3 + b_1\bar{b}_0 + \bar{b}_2\bar{b}_0 + \bar{b}_2 b_1 + b_2\bar{b}_1 b_0$$

$$e = b_1\bar{b}_0 + \bar{b}_2\bar{b}_0$$

$$f = b_3 + b_2\bar{b}_1 + b_2\bar{b}_0 + \bar{b}_1\bar{b}_0$$

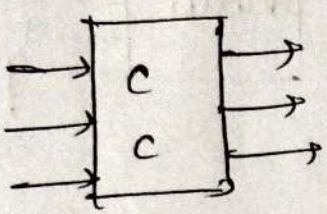
$$g = b_3 + b_2\bar{b}_1 + \bar{b}_2 \cdot b_1 + b_1\bar{b}_0$$

Combinational Vs Sequential:-

↓
o/p is only dependent on present i/p

eg: Adder

```
  1
+ 0
---
  1
```



↘ o/p is dependent on present i/p & past o/p's

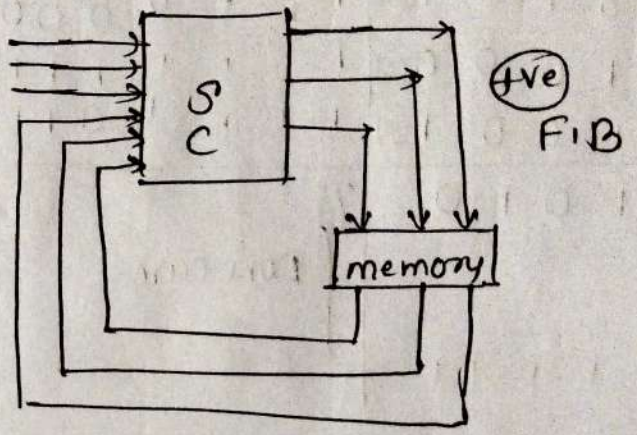
eg: Counter

+1 previous o/p

prev o/p → 4 +1 → 5 uses memory

3+1 → 4

prev o/p



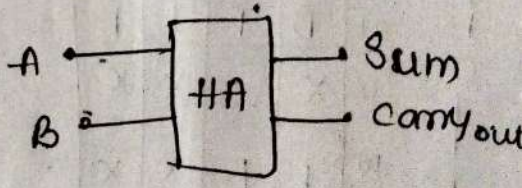
HAIF ADDER:-

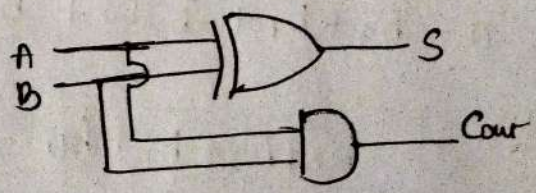
* used to add single bit no.
* Doesn't take carry from prev sum

A	B	S	Co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = A \oplus B$

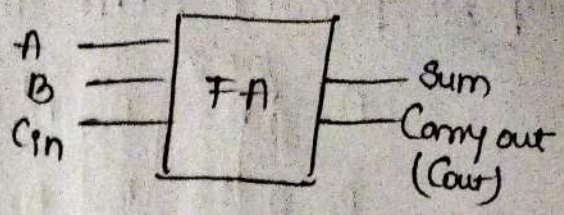
$C_o = A \cdot B$



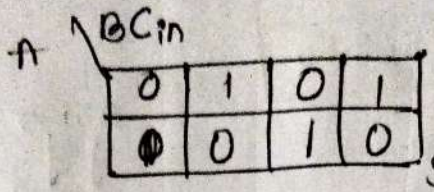


Full adder

A	B	C	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



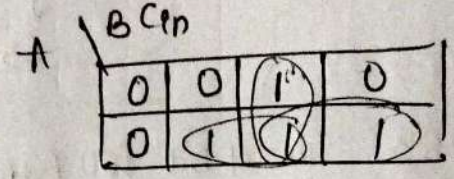
for sum:



check board cont

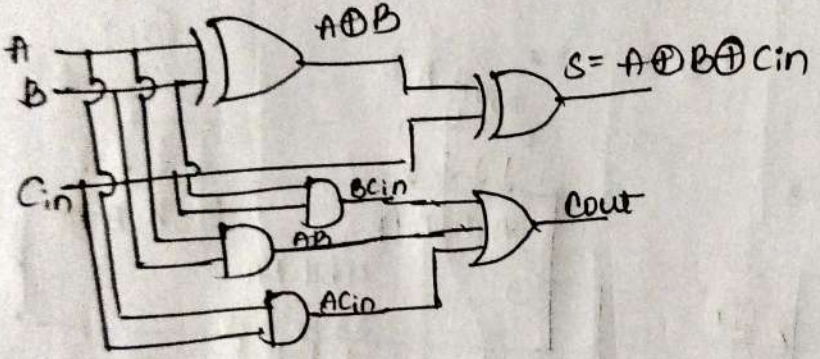
$S = A \oplus B \oplus C_{in}$

for Cout



$Cout = BCin + ACin + AB$

Logic gate



$Co = AB + Cin(A \oplus B)$

$Co = AB + A\bar{B}Cin + \bar{A}BCin$

$= A(B + \bar{B}Cin) + \bar{A}BCin$

$= A(B + Cin) + \bar{A}BCin$

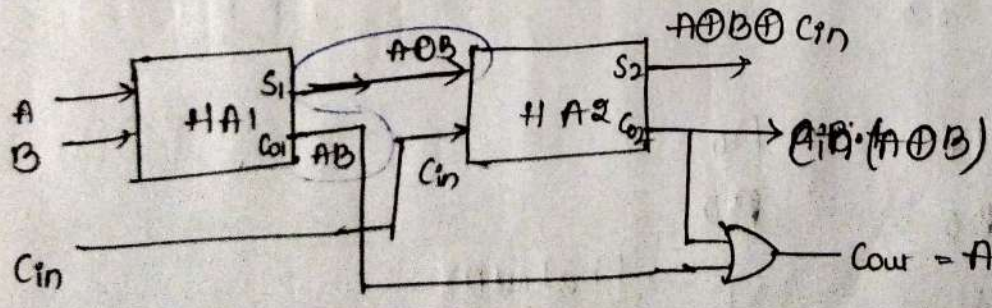
$= AB + ACin + \bar{A}BCin$

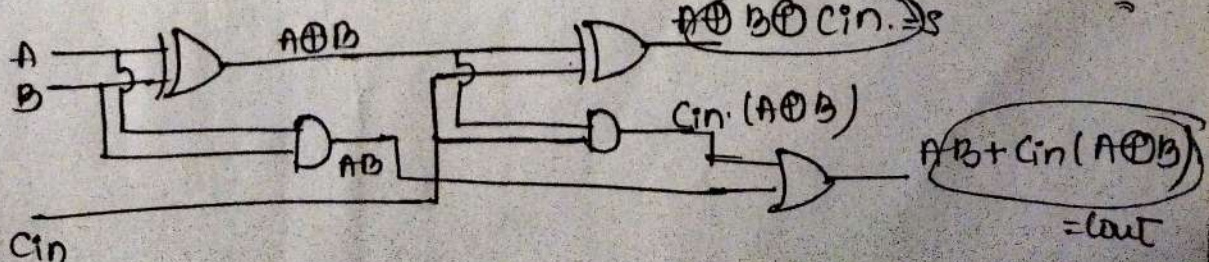
$= AB + (A + \bar{A}B)Cin$

$= AB + (A + B)Cin$

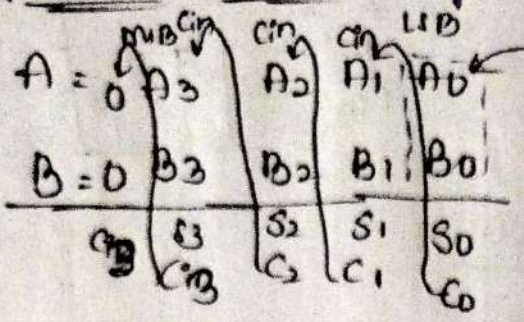
$= AB + ACin + BCin$ //.

Full adder using HA:- 2 HA = 1 FA





4-bit parallel Adder using FA :-

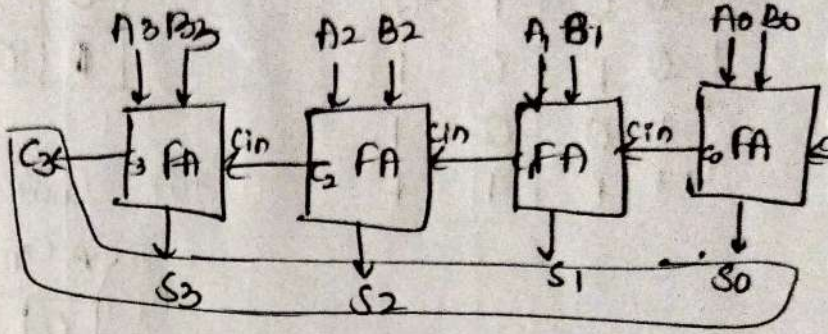


sum may be 4bit or 5bit

A+B = C3 S3 S2 S1 S0

C3 = 0 → 4bit

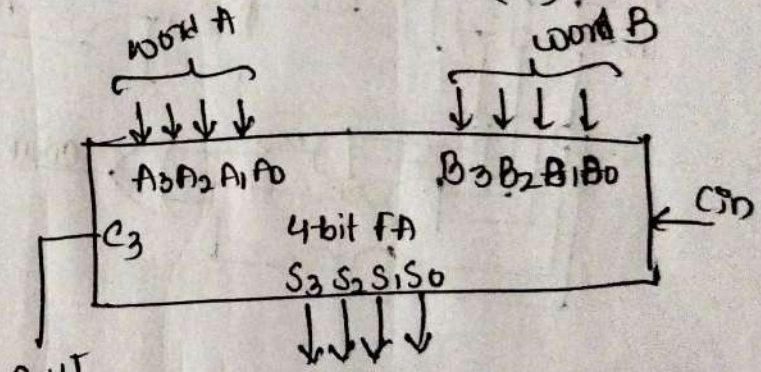
C3 = 1 → 5bit



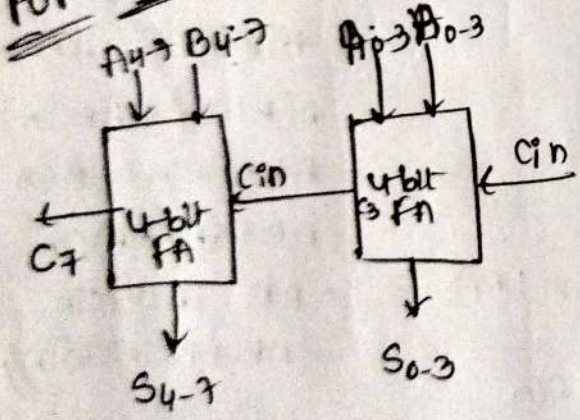
IC = 74HC283 for 4-bit full Adder

final sum

(or)



For 8-bit FA:-



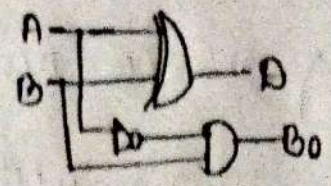
A = A7 A6 - - - A0

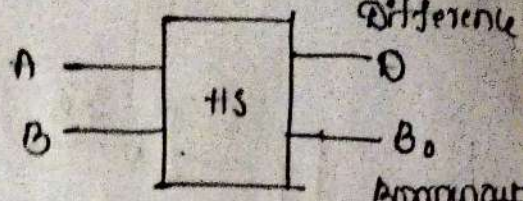
B = B7 B6 - - - B0

A+B = C7 S7 S6 - - - S1 → final sum

HALF SUBTRACTOR:-

1	0		$1-0=1$
(-) 0	1		$0-1=1$ (borrow)
0	1		

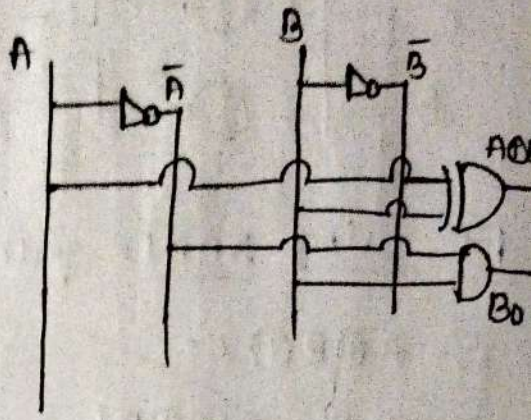




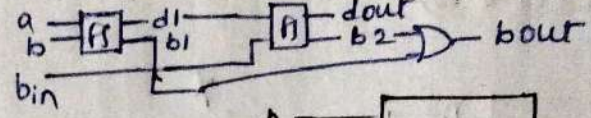
A	B	D	Bo
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$D = A \oplus B$

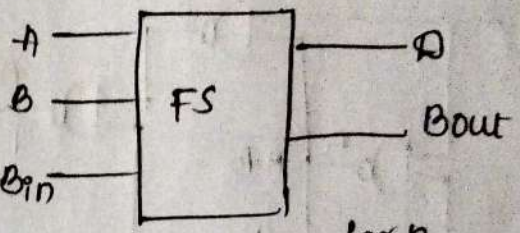
$B_o = \bar{A}B$



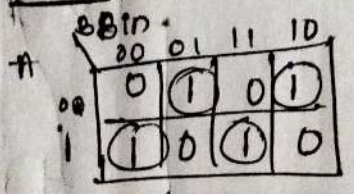
Full Subtractor:-



A	B	Bin	D	Bo
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



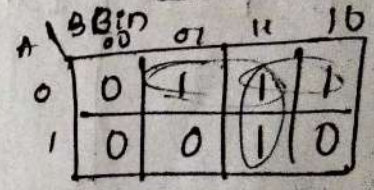
for D



Check board configuration

$D = A \oplus B \oplus Bin$

for B_o



$B_o = B\bar{B}_{in} + \bar{A}\bar{B}_{in} + \bar{A}B$

HA using NAND only:- $S = A \oplus B$, $Cout = AB$

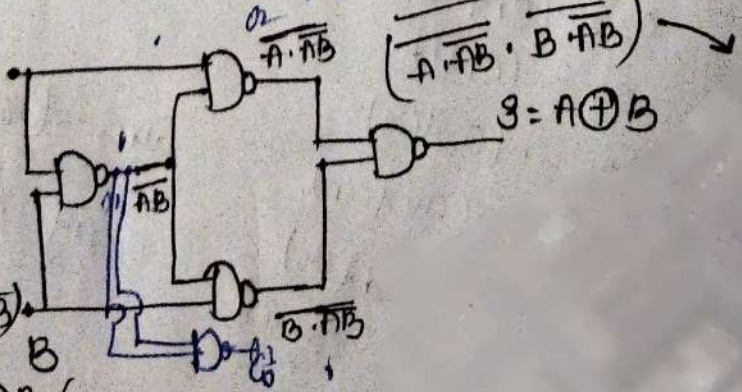
$= \overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}$

$= \overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}$

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

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

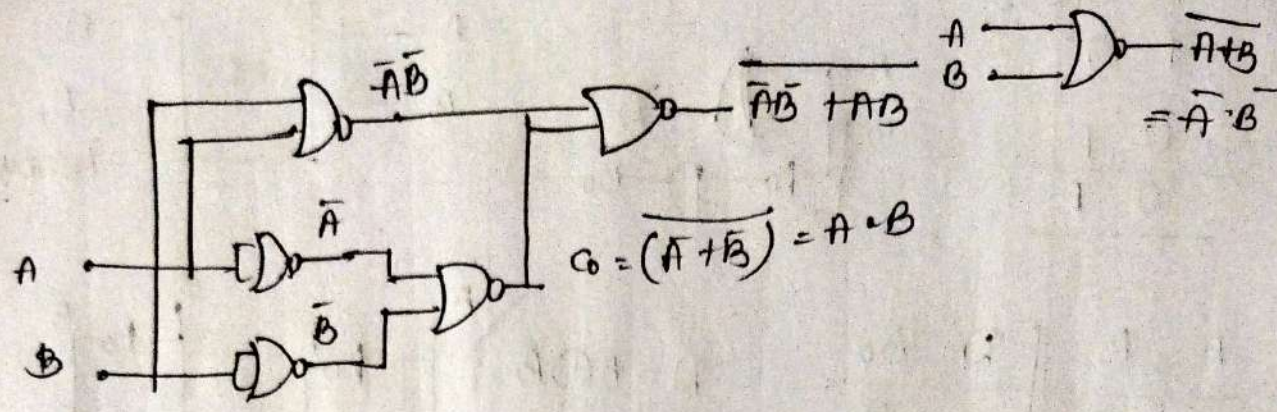
$= A\bar{B} + \bar{A}B = A \oplus B$ ✓



HA using NOR gate:-

$S = A \oplus B = A\bar{B} + \bar{B}A = \overline{\overline{A\bar{B}} + AB}$

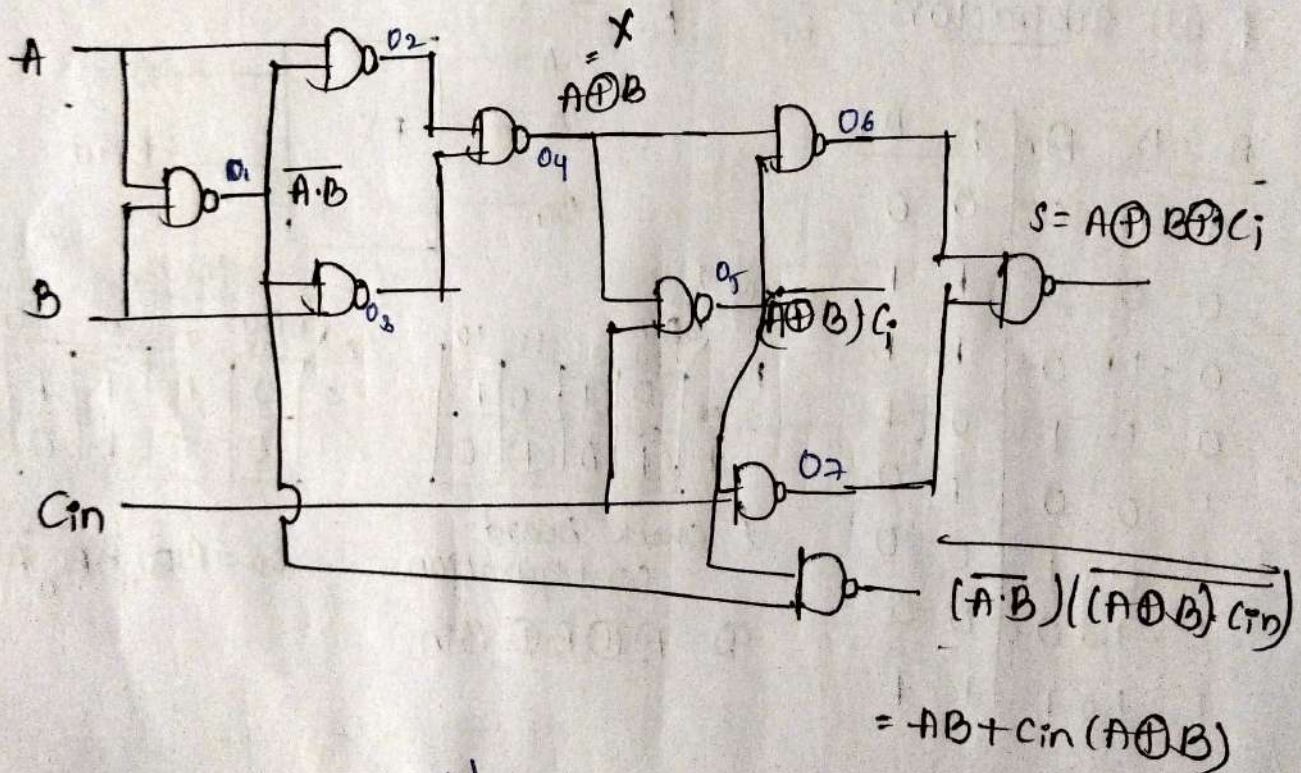
$C_0 = A \cdot B$



FA using NAND gate:-

$S = A \oplus B \oplus C_i$

$C_0 = A \cdot B + C_i(A \oplus B)$



$= AB + C_{in}(A \oplus B)$

$O1 = \sim(a \& b)$

$O_2 = \sim(O_1 \& a)$

$O_3 = \sim(O_1 \& b)$

$O_4 = \sim(O_2 \& O_3)$

$O_5 = \sim(O_4 \& Cin)$

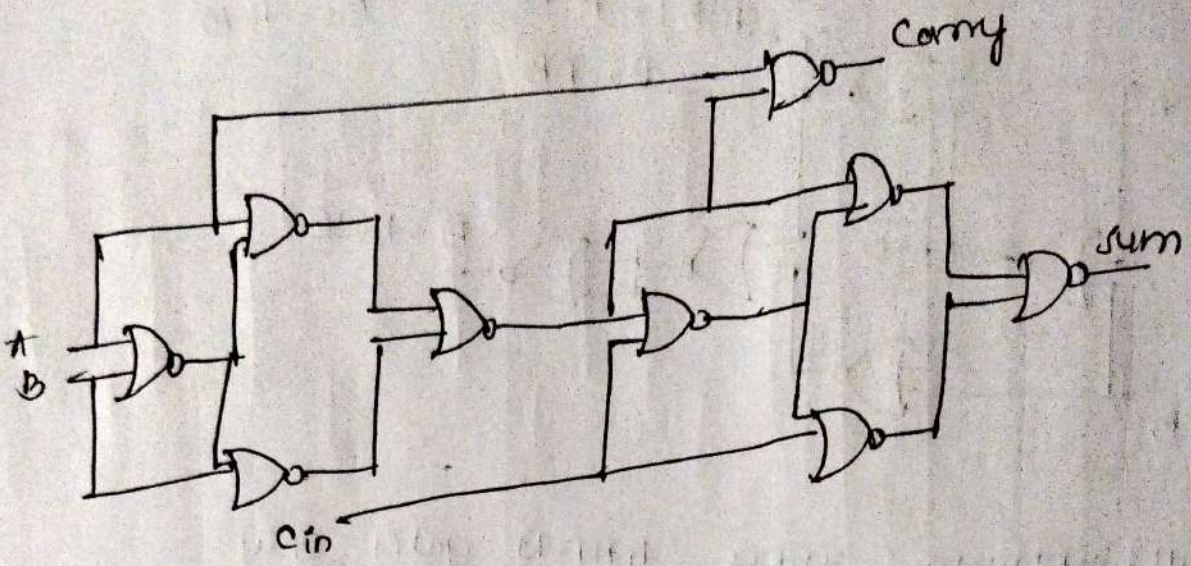
$O_6 = \sim(O_4 \& O_5)$

$O_7 = \sim(O_5 \& Cin)$

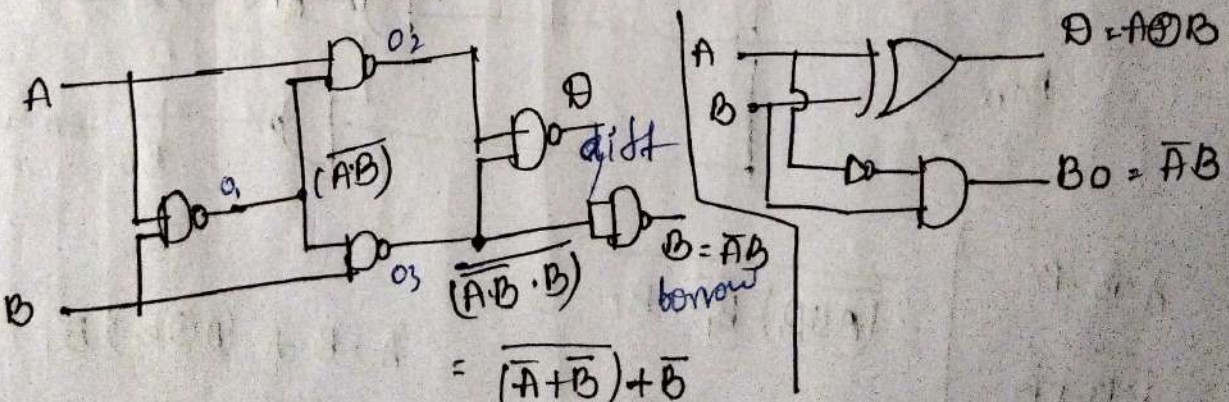
$sum = \sim(O_6 \& O_7)$

$carry = \sim(O_1 \& O_5)$

FA using NOT gate :-



HS using NAND gate :-



$O_1 = \sim(a \& b)$

$O_2 = \sim(a \& O_1)$

$O_3 = \sim(b \& O_2)$

$diff = \sim(O_2 \& O_3)$

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

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

$= AB+\overline{B}$

$= A+\overline{B}$

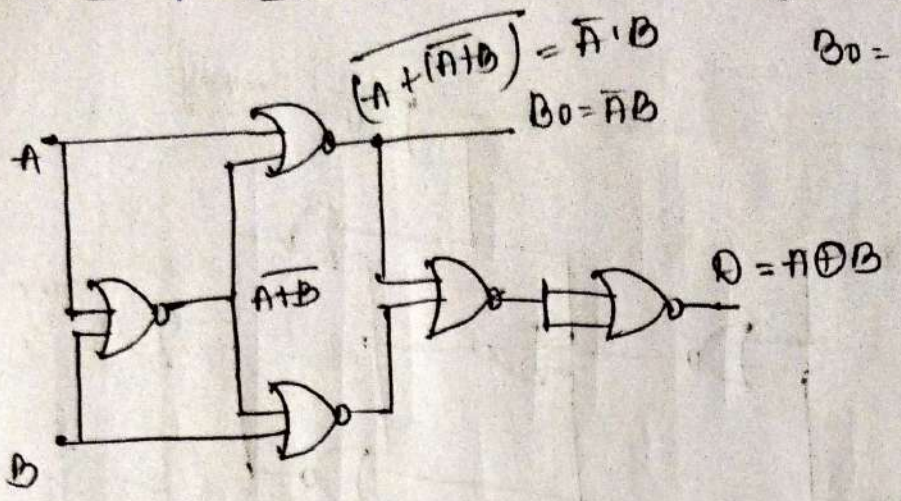
$= \overline{\overline{A+\overline{B}}}$

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

HS using NOR gate only:-

$D = A \oplus B$

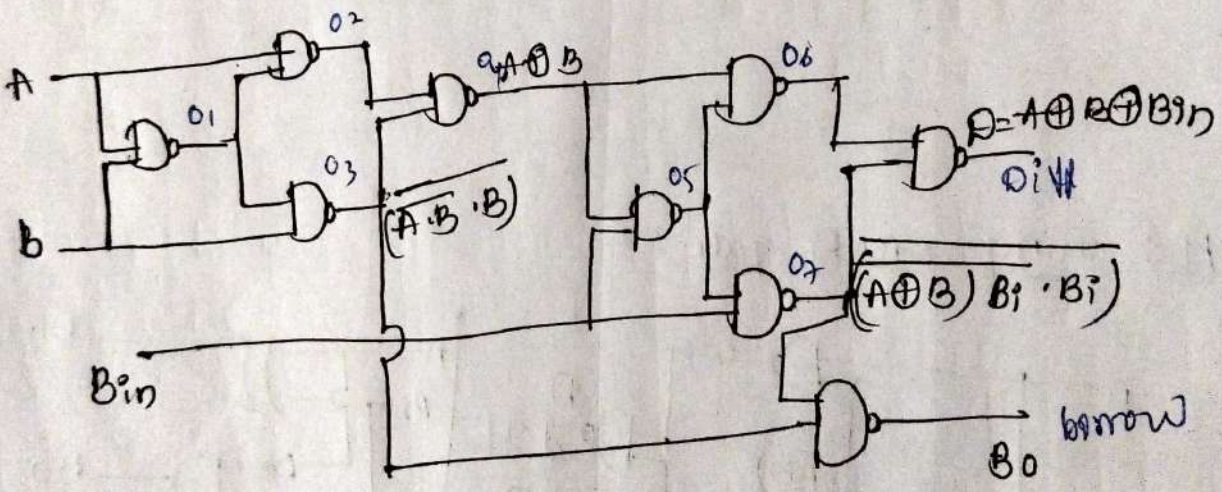
$B_0 = \bar{A}B$



Full subtractor using NAND gates only:-

$D = A \oplus B \oplus B_{in}$

$B_{out} = \bar{A}B + \bar{A}B_{in} + BB_{in}$



$\overline{\overline{A \cdot B} \cdot B}$

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

$= AB + \bar{B}$

$= A + \bar{B}$

$= \overline{\overline{(A \oplus B) B_i} \cdot B_i}$

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

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

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

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

$= \bar{A}B + ABB_i + \bar{A}\bar{B}B_i$

$B_0 = \overline{(A+\bar{B}) \cdot (A \oplus B + \overline{B_i})}$

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

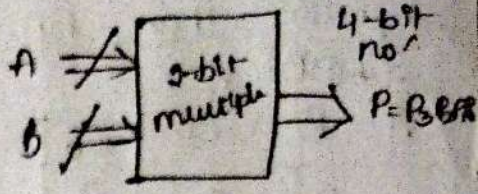
$= (\bar{A} \cdot B) + \overline{(A \oplus B) \cdot B_i}$

2-Bit Multipliers:- multiplies 2 bit no's

A = A_1 A_0

B = B_1 B_0

		A_1B_0	A_0B_0
	C_1		X
(+)	A_1B_1	A_0B_1	
C_2	$A_1B_1+C_1$	$A_1B_0+A_0B_1$	A_0B_0
	C_2	C_1	

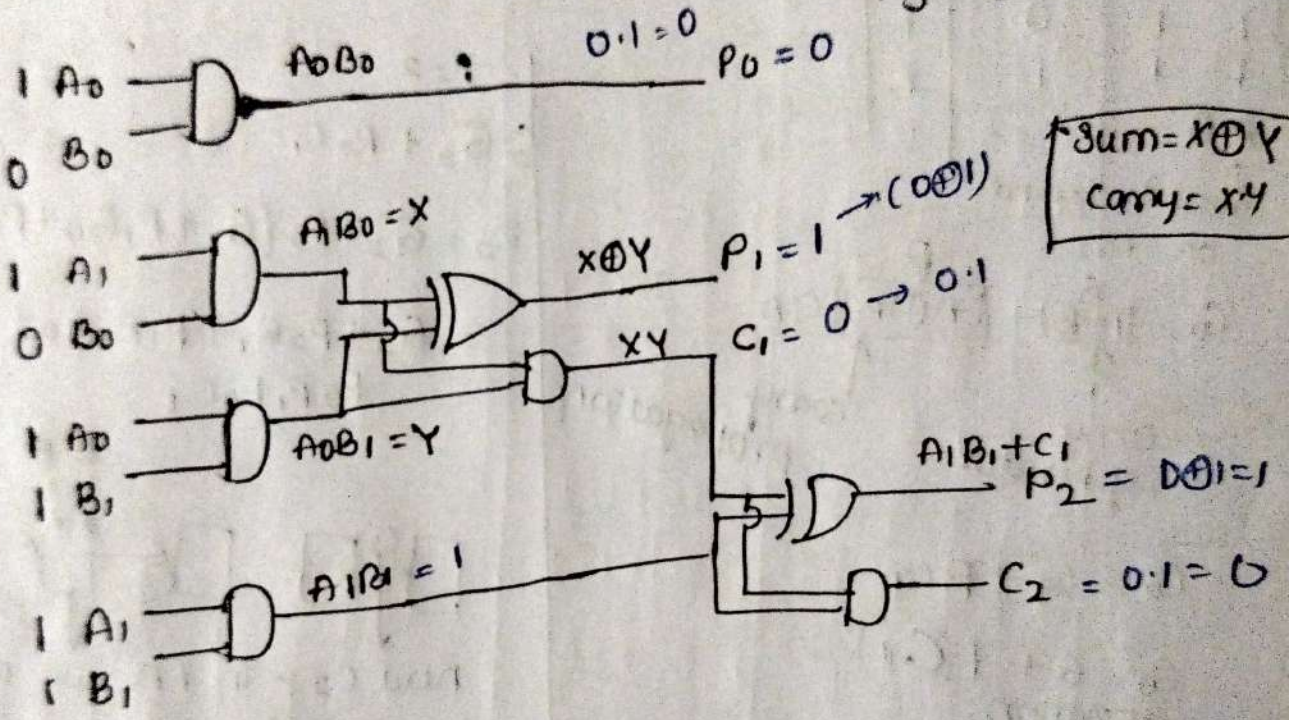


$P_0 = A_0B_0$

$P_1 = \underbrace{A_1B_0 + A_0B_1}_{HA}$

$P_2 = \underbrace{A_1B_1 + C_1}_{HA}$

$P_3 = C_2$



Sum = X⊕Y

Carry = XY

A_1A_0

A = 1 1

B_1B_0

B = 1 0

---

0 0

1 1 X

---

1 1 0

Carry look Ahead adder (CLA adder) predict the carry

* Superior than FA due to speed

A	B	Cin	C_0
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$(A \oplus B)C_{in}$

A·B doesn't depend on Cin

$C_0 = \underbrace{A \cdot B}_{G} + \underbrace{(A \oplus B)}_{P} C_{in}$

$A \cdot B$ → carry generator

$(A \oplus B)C_{in}$ → carry propagation

$C_0 = G + P C_{in}$

$C_0 = G + P(C_{-1})$

Generalization:

$C_i = G_i + P_i C_{i-1}$

$i = 0$

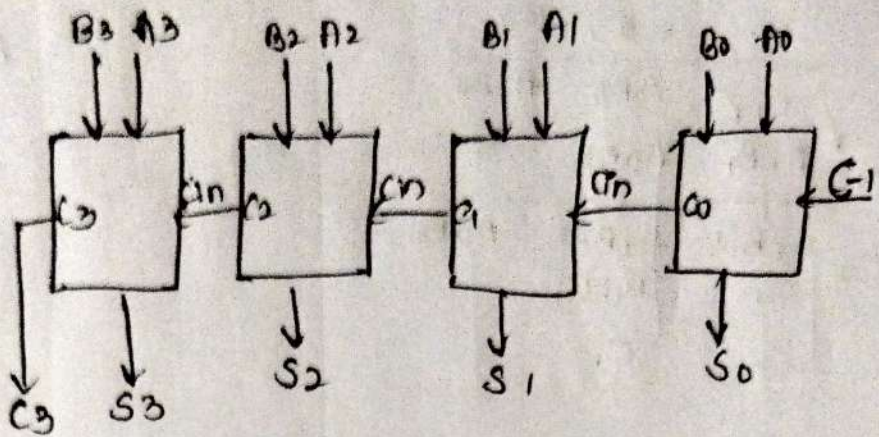
$C_0 = G_0 + P_0 C_{-1} \rightarrow (1)$

$i = 1$

$G_1 = G_1 + P_1 C_0$

$C_1 = G_1 + P_1 (G_0 + P_0 C_{-1}) \rightarrow (2)$

$C_1 = G_1 + P_1 G_0 + P_1 P_0 C_{-1}$ → we got C_1 in RHS so no need to wait for carry to generate

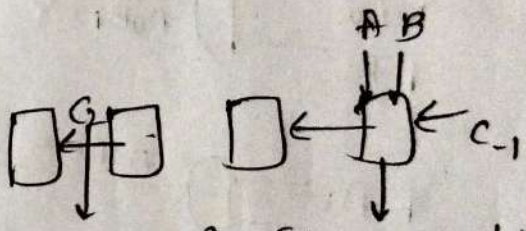


$i = 2$

$C_2 = G_2 + P_2 C_1$

$C_2 = G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_{-1})$

$= G_2 + P_2 G_1 + P_1 P_2 G_0 + P_2 P_1 P_0 C_{-1}$



need C_2, in FA we need to wait for C_0, C_1

but here C_2 = 2 terms of C_{-1}

time will saved

$i = 3$

$C_3 = G_3 + P_3 C_2$

$C_3 = G_3 + P_3(G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1})$

$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{-1}$

$G_3 = A_3 \cdot B_3$

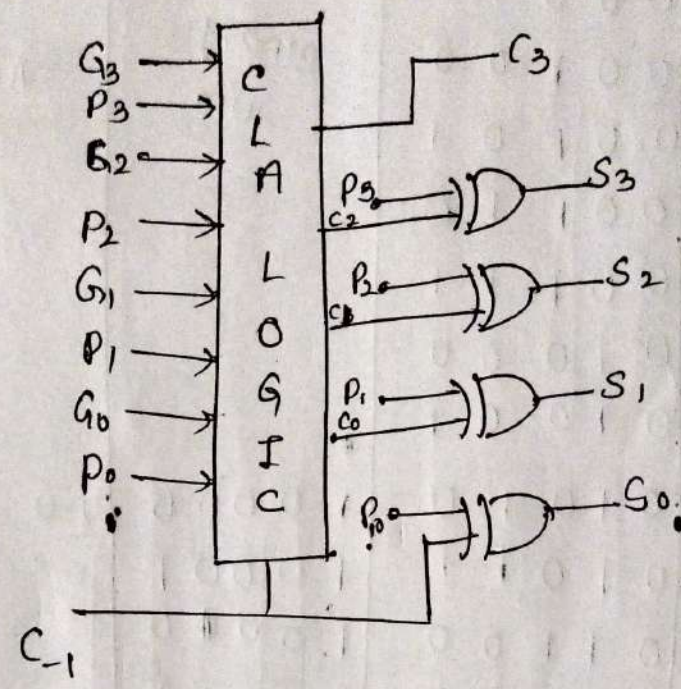
$P_3 = A_3 \oplus B_3$

$P_2 = A_2 \oplus B_2$

$P_1 = A_1 \oplus B_1$

$P_0 = A_0 \oplus B_0$

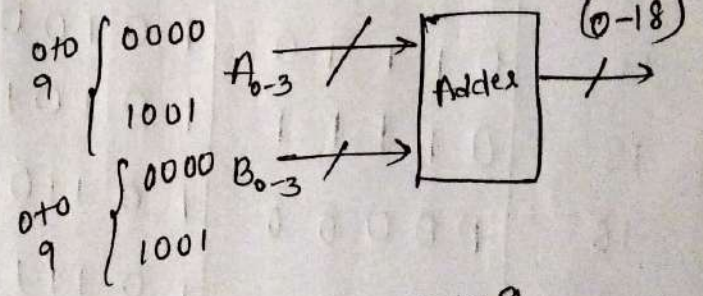
$S_3 = A_3 \oplus B_3 \oplus C_2$



BCD Adder

from 0 to 9

Binary sum = BCD sum



why 0 to 9

coz we are adding decimal which is 0 to 9

* here we don't consider carry prev sum

if we consider carry then sum = 0 to 19.

Decimal	Binary sum					BCD sum				
	C^*	S_3^*	S_2^*	S_1^*	S_0^*	C	S_3	S_2	S_1	S_0
0	0	0	0	0	0	Same as Binary sum				
1	0	0	0	0	1					
2	0	0	0	1	0					
3	0	0	0	1	1					
4	0	0	1	0	0					
5	0	0	1	0	1					
6	0	0	1	1	0					
7	0	0	1	1	1					
8	0	1	0	0	0					
9	0	1	0	0	1					
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

each digit BCD: (15) → 0001 0101 → 00010101

but 15 → binary 1111

Add '0110' ('6') when

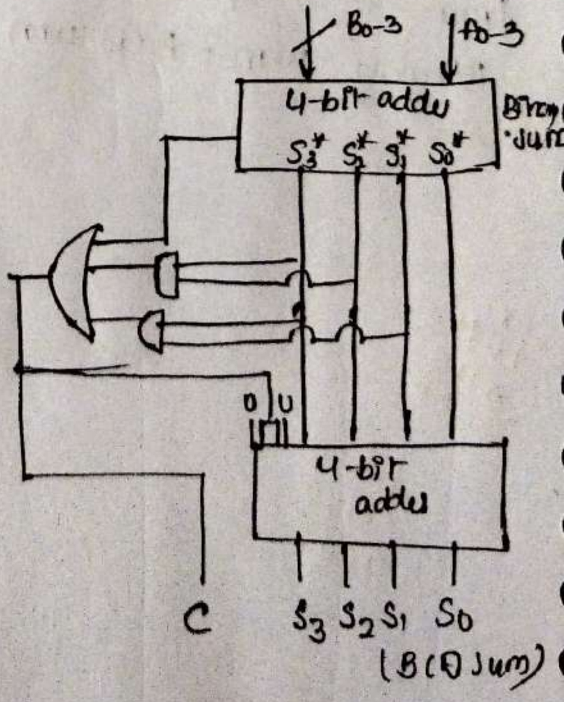
(i) $C^* = 1$

(ii) $S_3^* \cdot (S_2^* + S_1^*)$

(iii) $S_3^* \cdot S_1^*$

$= C^* + S_3^* \cdot (S_2^* + S_1^*) + S_3^* \cdot S_1^*$

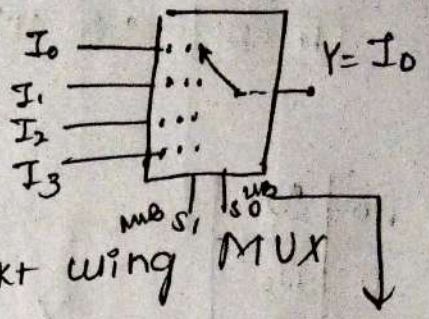
$= C^* + S_3^* S_2^* + S_3^* \cdot S_1^*$



Introduction to Multiplexers:- (MUX)

→ combinational ckt that selects binary info from one of many i/p lines & directs it to o/p line

→ Simple data selector

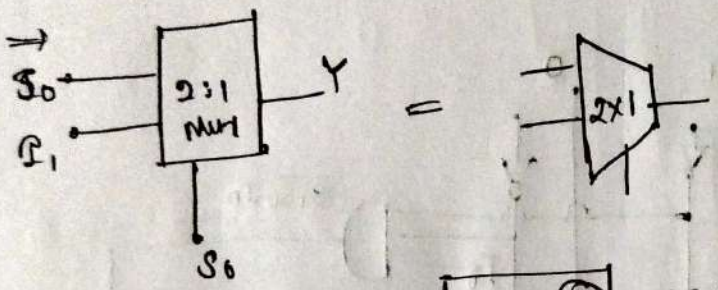


advantages / why mux:-

1. Reduce no of wires → because gates ↓
2. " ckt complexity & cost
3. Implementation of various ckt using universal logic ckt

if $S_1, S_0 = 0\ 0$ the I_0 will selected

if $S_1, S_0 = 1\ 1$ then I_3 will selected



$n = 2^m$ → no of Select lines

no of i/p's

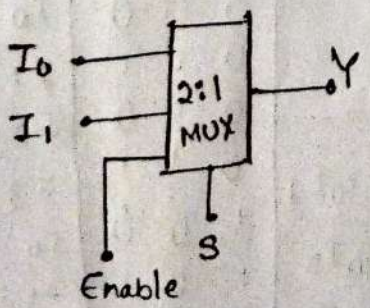
(or)

$m = \log_2 n$

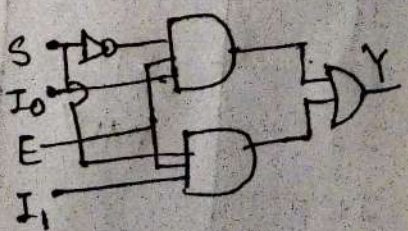
Types:

	Selected variable
2:1 MUX	1
4:1 MUX	2
8:1 MUX	3
16:1 MUX	4
32: MUX	5

2:1 MUX:-



E	S	Y
0	X	0
1	0	I_0
1	1	I_1



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

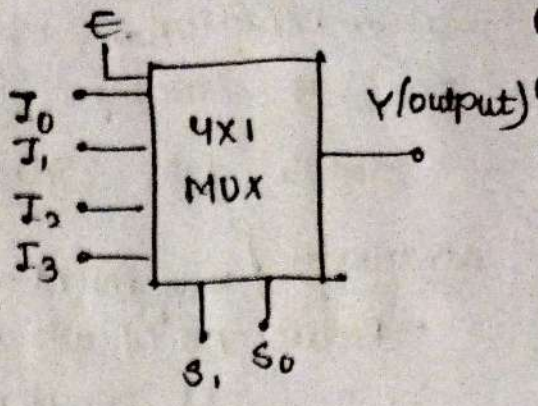
$Y = E(\bar{S} I_0 + S \cdot I_1)$

4X1 MUX:

$n=4 \rightarrow$ 4 i/p's

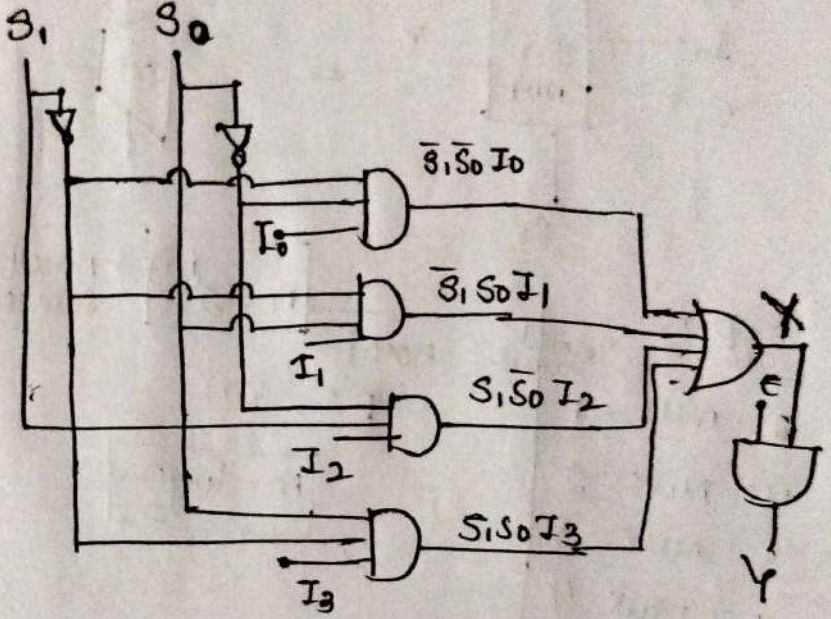
$m=\log_2 4$

$m=2 \rightarrow$ 2 selection variables

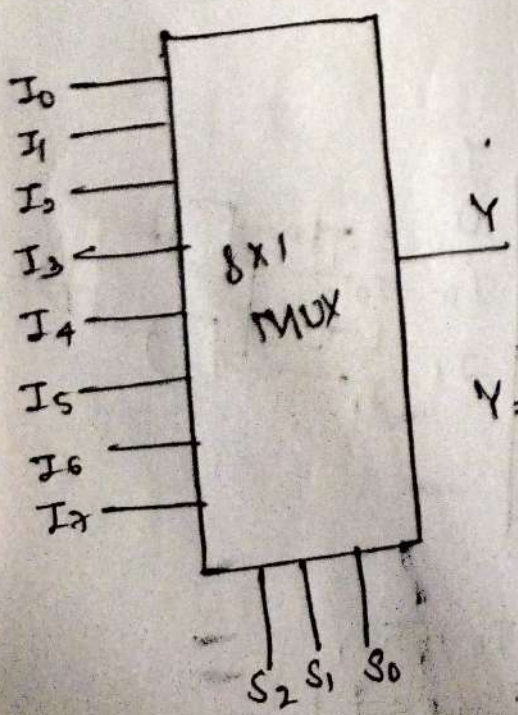


S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$Y = \bar{S}_1\bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1\bar{S}_0 I_2 + S_1 S_2 I_3$$



8X1 MUX:-



$n=8$

$m=\log_2 8$

$m=3$

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

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

↳ logic gate a/c to eqn

MUX Tree (higher MUX from lower MUX]

Implement 4x1 MUX using 2x1 MUX :-

4x1
n = 4

$\frac{4}{2} = 2$ → sequence : 2 - 2x1 MUX & then 1 2x1 MUX

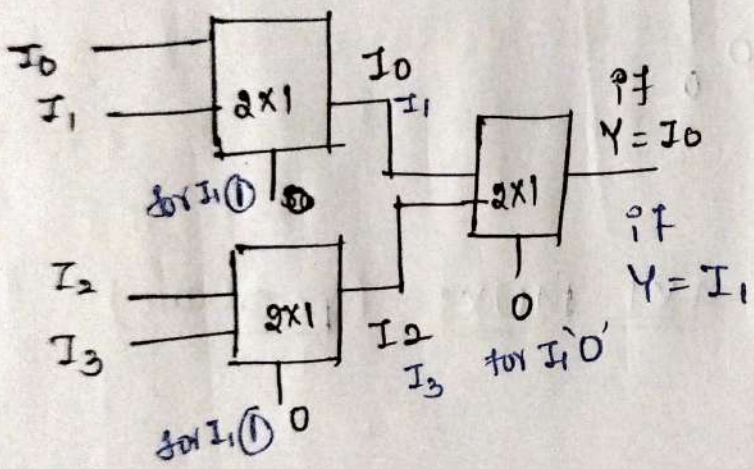
$\frac{2}{2} = 1$

$2 + 1 = 3$ → 2x1 MUX req to implement '1' 4x1 MUX

once we have '1' stop

4x1 MUX :-

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 :-

$n_1 = 8$ → req mux

$n_2 = 2$ → available mux

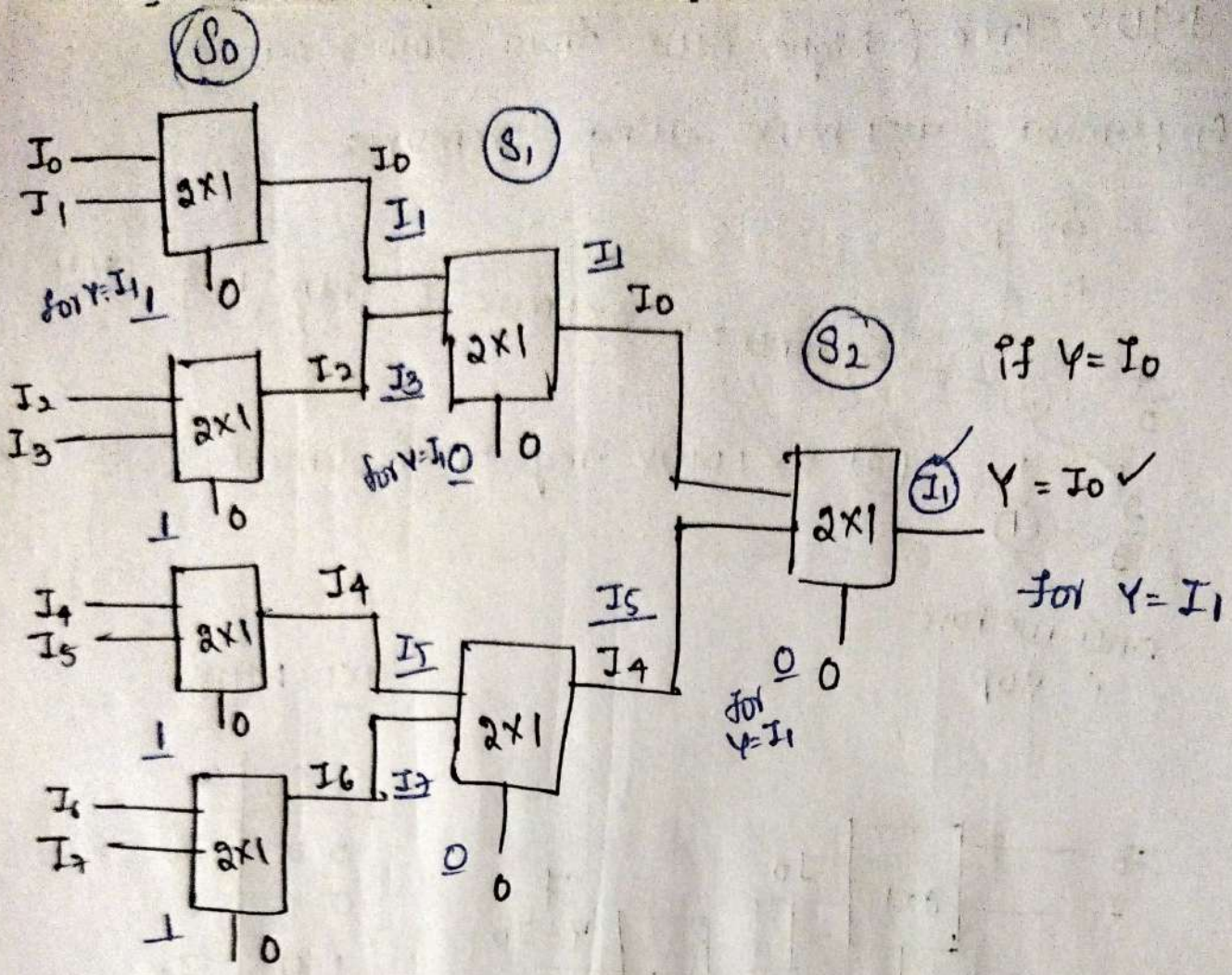
$\frac{8}{2} = 4$ → sequence 4, 2, 1

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

$4 + 2 + 1 = 7$ 2x1 MUX req to implement '1' 8x1 MUX

$\frac{2}{2} = 1$ → stop

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



8x1 MUX using 4x1 MUX:- (special case)

$n_1 = 8$

$n_2 = 4$

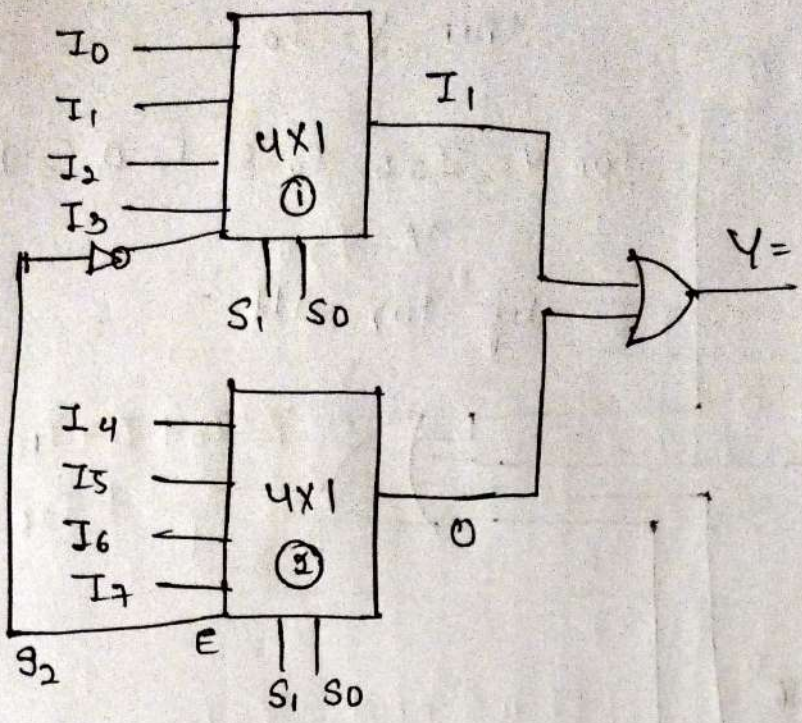
$\frac{8}{4} = 2$ + $\frac{2}{4} = \frac{1}{2}$

$= 2.5$ 4x1 MUX for 1 8x1 MUX

it means all of the i/p of the 3rd 4x1 MUX is not used.

So, we can't do it in normal way.

we can do this thing using Enable.



⇒ if we need $Y = I_1$ then

$S_2 = 0$

② 4MUX OFF

① 4X1 MUX ON

S_1 S_0 will be 01

then $Y = I_1$

if $S_2 = 0$ ② 4X1 OFF o/p = 0

But $S_2 = 0$ ① 4X1 ON o/p ✓

⇒ if we need $Y = I_7$ then $S_2 = 1$

② 4X1 MUX ON

① 4X1 MUX OFF

& S_1 S_0 will be 11

then $Y = I_7$

---

32 X1 MUX using 8X1 MUX

$n_1 = 32$

$n_2 = 8$

$\frac{32}{8} = 4$

$\frac{4}{8} = 1/2$

$x_1 \uparrow$ - $S_4 \downarrow S_3 \downarrow$

$x_2 \uparrow$ - $S_4 \downarrow S_3 \uparrow$

$x_3 \uparrow$ - $S_4 \uparrow S_3 \downarrow$

$x_4 \uparrow$ - $S_4 \uparrow S_3 \uparrow$

	S_4	S_3
x_1	0	0
x_2	0	1
x_3	1	0
x_4	1	1

$x_1 = \overline{S_4}\,\overline{S_3}$

$x_2 = \overline{S_4} S_3$

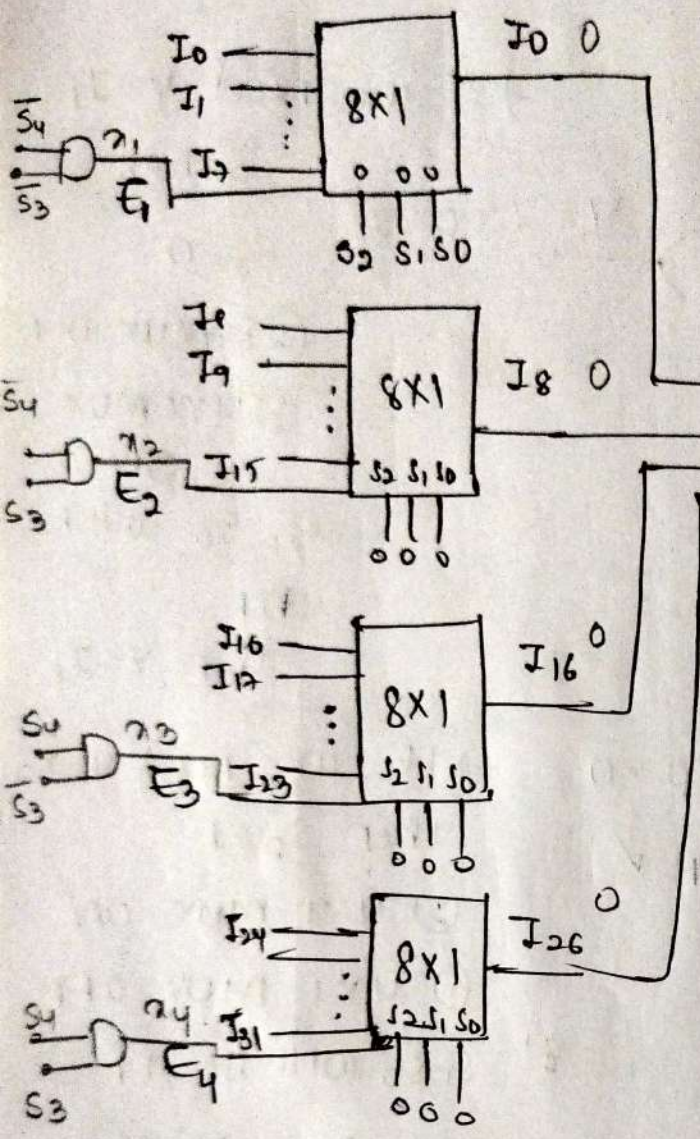
$x_3 = S_4 \overline{S_3}$

$x_4 = S_4 S_3$

S_4	S_3	S_2	S_1	S_0	Y	
0	0	0	0	0	I_0	x_1
0	0	0	0	1	I_1	
0	0	1	1	1	I_7	
0	1	0	0	0	I_8	x_2
0	1	0	0	1	I_9	
0	1	1	1	1	I_{15}	
1	0	0	0	0	I_{16}	x_3
1	0	0	0	1	I_{17}	
1	0	1	1	1	I_{23}	
1	1	0	0	0	I_{24}	x_4
1	1	0	0	1	I_{25}	
1	1	1	1	1	I_{31}	

for $Y = I_0$, $E_1 = 1$, $E_2 = E_3 = E_4 = 0$

then $Y = I_0$



for $Y = I_{26}$, $E_0 = E_1 = E_2 = 0$, $E_3 = 1$

$Y = I_{26}$

By for all

$Y = I_0 + I_8 + I_{16} + I_{26}$

$Y = I_{25}$

if we need $Y = I_{25}$

for I_{25}, $S_4 = 1$, $S_3 = 1$

so, $x_1, x_2, x_3 = 0$ ($E_1 = E_2 = E_3 = 0$)

$x_4 = 1$, that Enable = 1

& $S_2\ S_1\ S_0$ is

0 0 1

$Y = 0 + 0 + 0 + I_{25}$

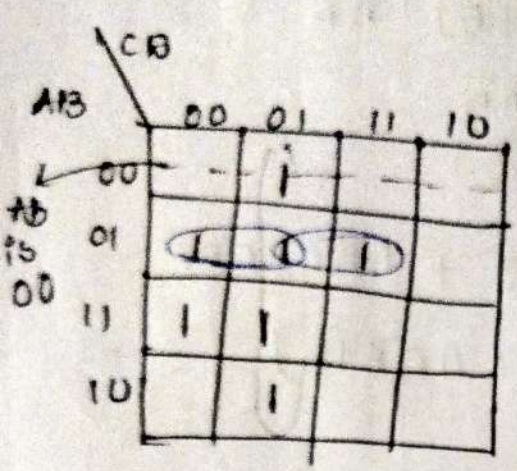
$Y = 25$

⇒ Implementation of boolean func using MUX:-

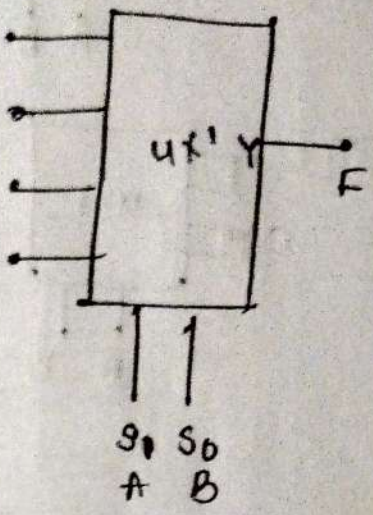
Compliment : $F(A,B,C,D) = \Sigma m(1,4,5,7,9,12,13)$

using 4x1 MUX

$S_0 = B$
$S_1 = A$



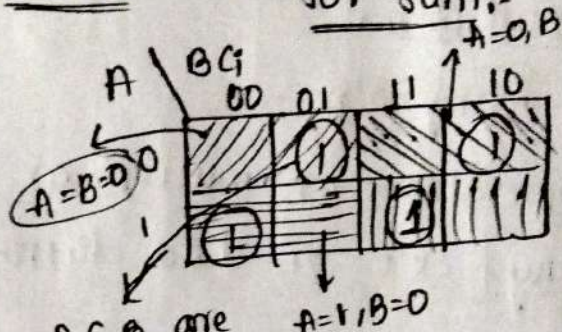
$S_1 S_0$	Y
0 0	$I_0 = \bar{C}D$
0 1	$I_1 = \bar{C}+D$
1 0	$I_2 = \bar{C}D$
1 1	$I_3 = \bar{C}$



1-Bit FA using MUX :-

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

for sum:-

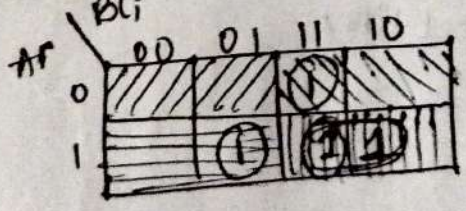


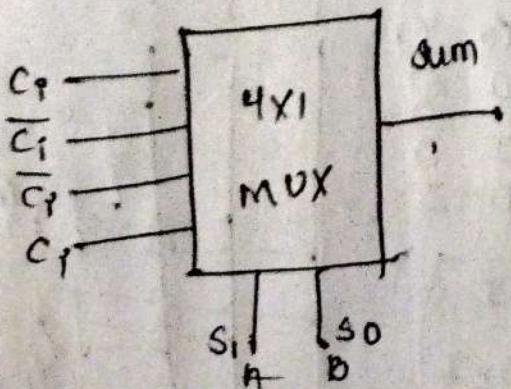
Check board cont

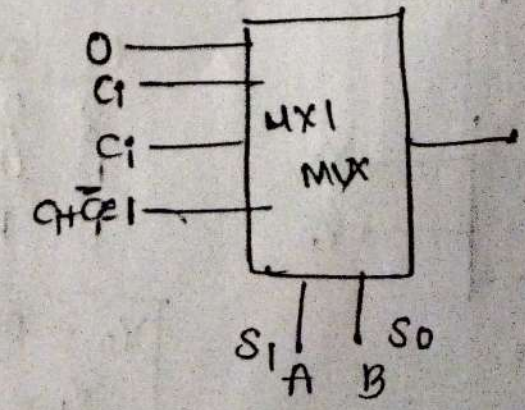
$S = A \oplus B \oplus C_i$

A B	S
0 0	$I_0 = C_i$
0 1	$I_1 = \bar{C}_i$
1 0	$I_2 = \bar{C}_i$
1 1	$I_3 = C_i$

for carryout:



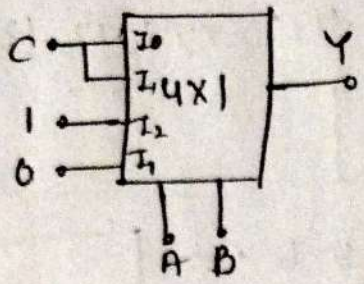




Logical expression from MUX :-

eg:

(i)



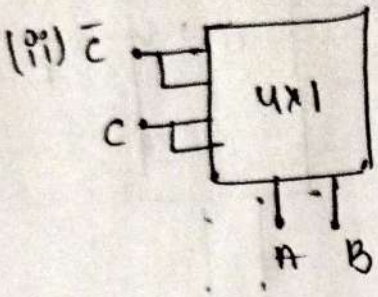
$Y = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\cdot 1 + 0$

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

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

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

(ii)



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

$Y = \bar{A}\bar{C}[\bar{B}+B] + AC[\bar{B}+B]$

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

$\boxed{Y = A \odot C}$

Demux : Demultiplexer

→ one input & many o/p

→ Reverse operation of multiplexer

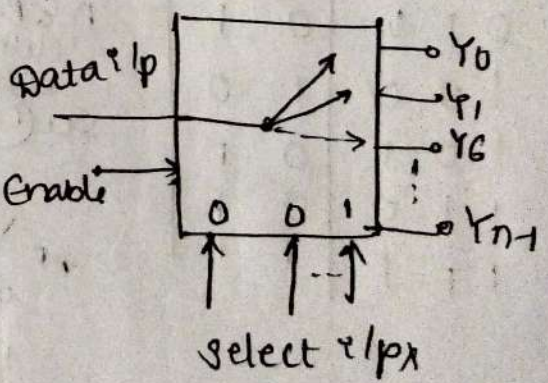
→ One to many ckt or data distributor

1 : Many

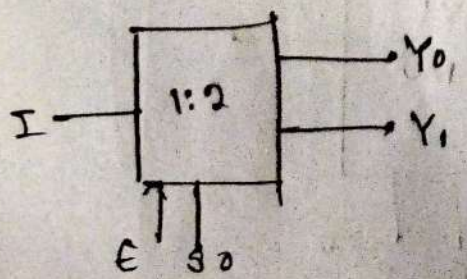
n → o/p lines

m → select lines

$\boxed{m = \log_2 n}$



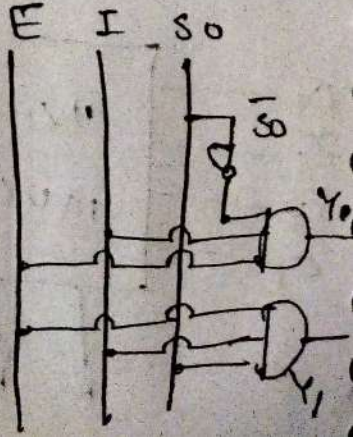
1:2 Demux:-



E	S_0	Y_0	Y_1
0	0	0	0
0	1	0	0
1	0	I	0
1	1	0	I

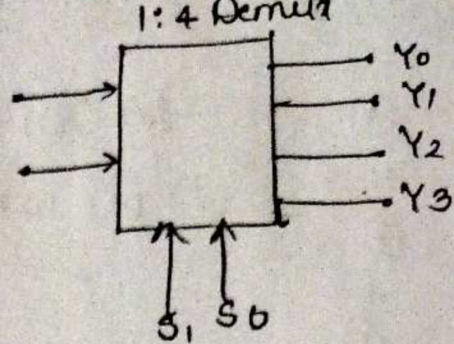
$\boxed{Y_0 = E\bar{S}_0 I}$

$\boxed{Y_1 = E S_0 I}$



1:4 Demultiplexer:-

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

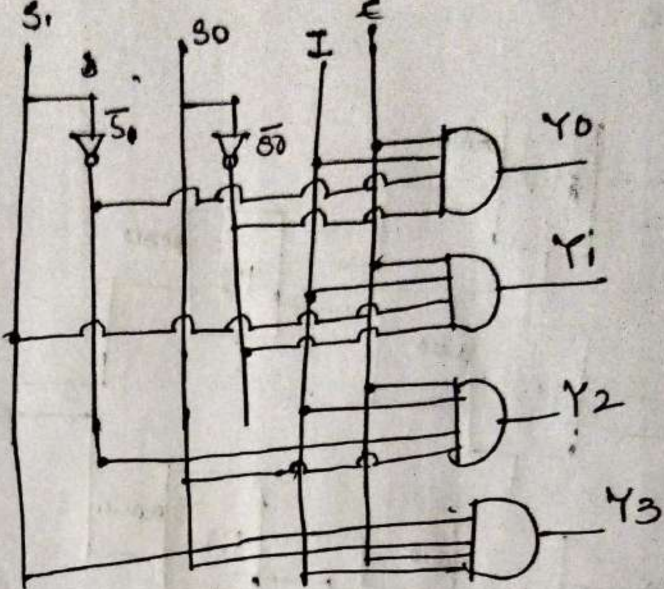


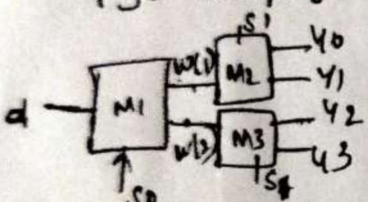
$Y_0 = E\bar{S_1}\bar{S_0}I$

$Y_2 = ES_1\bar{S_0}I$

$Y_1 = E\bar{S_1}S_0I$

$Y_3 = ES_1S_0I$





Full subtractor using 1:8 Demultiplexer:- (select variables as req i/p)

A	B	Bin	D	Bo
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

S_2 (A)	S_1 (B)	S_0 (Bi)	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

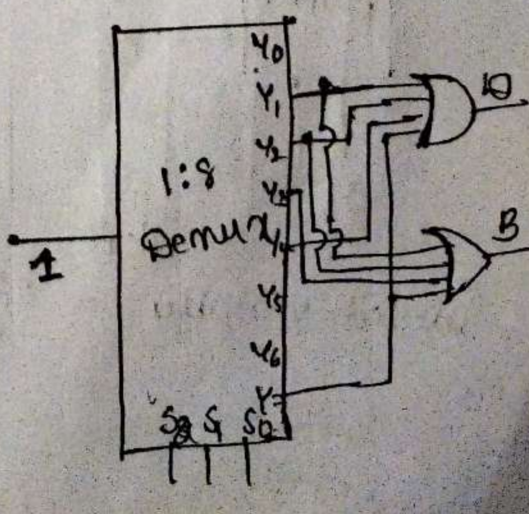
Standard SOP:-

$D(A, B, Bin) = \Sigma m(1,2,4,7)$

$Bo(A, B, Bin) = \Sigma m(1,2,3,7)$

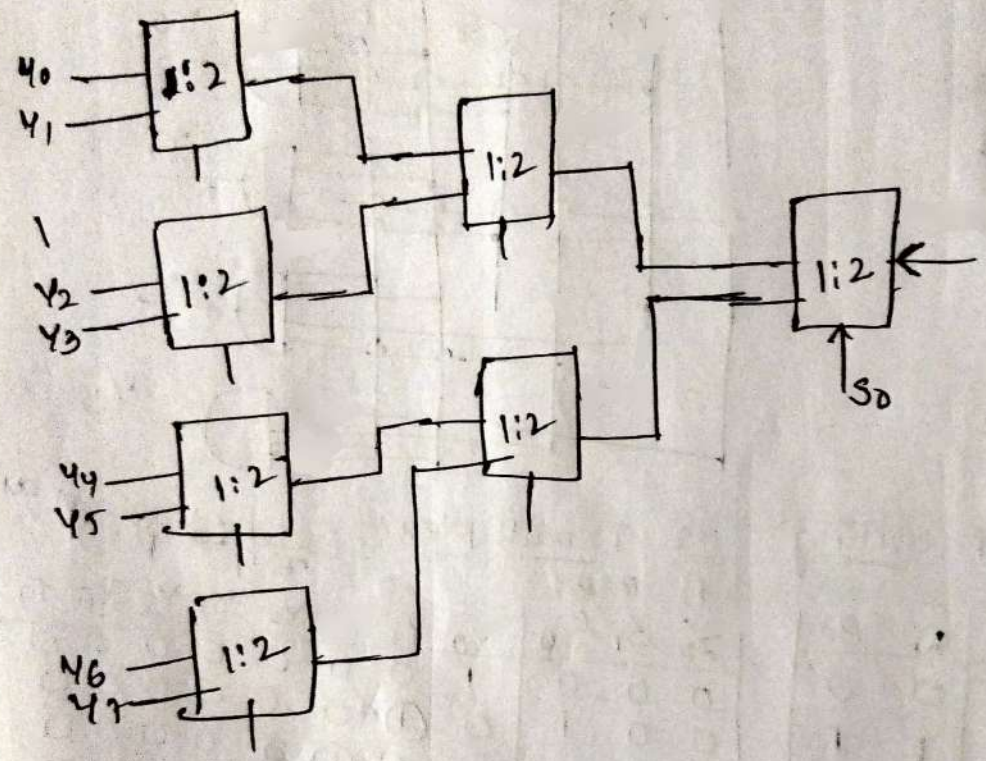
for $D = Y_1 + Y_2 + Y_4 + Y_7$

for $Bo = Y_1 + Y_2 + Y_3 + Y_7$



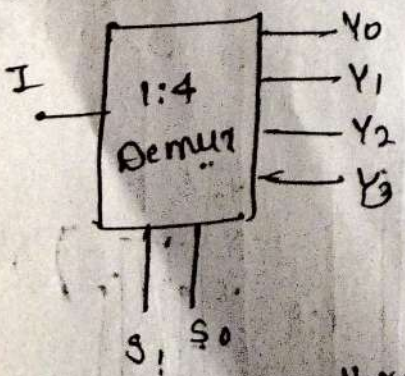
8:1 Demux using 2:1 Demux

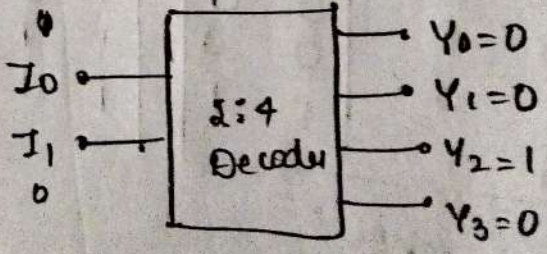
$\frac{8}{2} = 4$ + $\frac{4}{2} = 2$ + $\frac{2}{2} = 1$ = 7 for 8:1 Demux



Demux as Decoder:- 2:4 decoder using 1:4 Demux

[3:8 Decoder using 1:8 Demux]





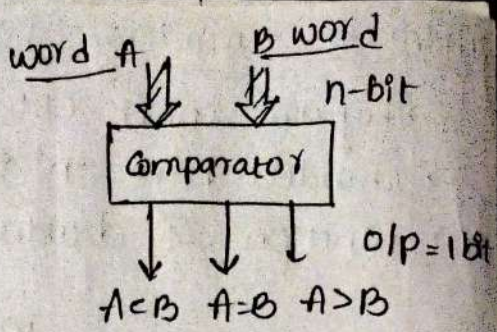
Here 1:4 Demux
selected variables = 2:4 Decoder o/p

$S_0 = I_0$
$S_1 = I_1$

S_1 S_0					
I_1	I_0	Y_0	Y_1	Y_2	Y_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

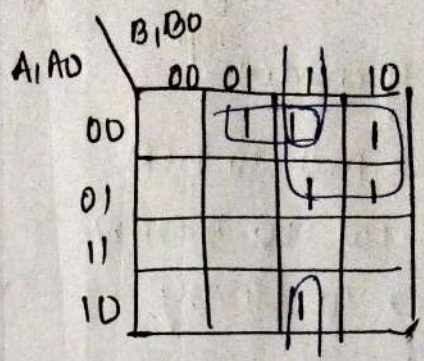
2 Bit Comparator :-

→ A digital comparator is a comb ckt designed to compare 2 n-bit binary words



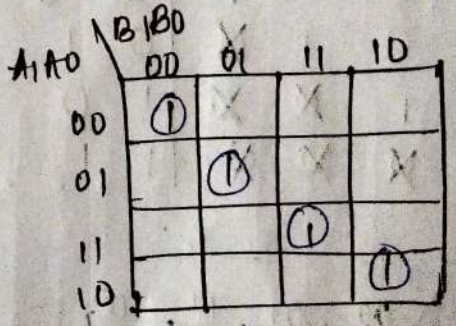
Inputs				o/p		
A_1	A_0	B_1	B_0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

for A<B :-



$A<B = \bar{A}_1 \bar{A}_1 B_0 + \bar{A}_1 B_1 + \bar{A}_0 B_1 B_0$

for A=B :-

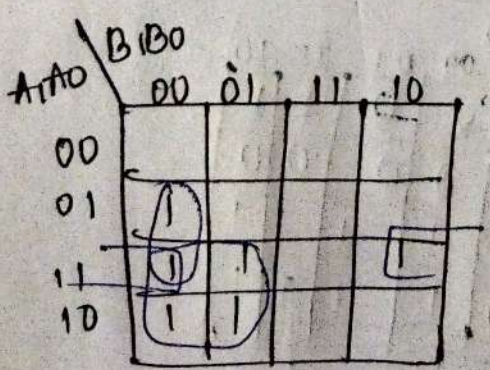


$A=B \Rightarrow \bar{A}_1 A_0 B_1 \bar{B}_0 + \bar{A}_1 A_0 \bar{B}_1 B_0 + A_1 A_0 B_1 B_0 + A_1 \bar{A}_0 B_1 \bar{B}_0$

$= A_0 B_0 [\bar{A}_1 \bar{B}_1 + A_1 B_1] + \bar{A}_0 \bar{B}_0 [\bar{A}_1 \bar{B}_1 + A_1 B_1]$

$= [A_0 B_0 + \bar{A}_0 \bar{B}_0] = (A_1 \odot B_1)$ $(A_0 \odot B_0)$
$[\bar{A}_1 \bar{B}_1 + A_1 B_1]$

for A>B :-



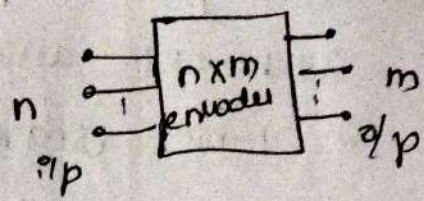
$A>B = A_0 \bar{B}_1 B_0 + A + A_1 \bar{B}_1$

Intro to Encoders & decoders :- (Medium scale IC) ∆MUX & DE

→ They are comb ckt

→ Encoders have "n" i/p & "m" o/p

→ Function of decoder is opp to encoder



$n = 2^m$

$m = \log_2 n$ → i/p

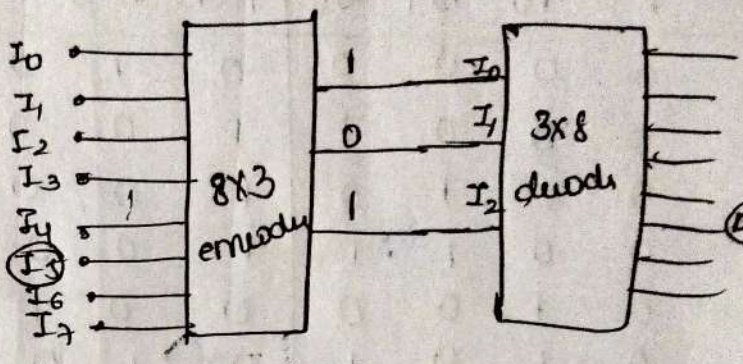
↓ o/p

priority encoder:-

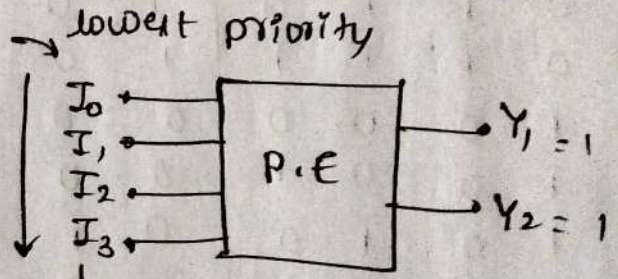
(ii) Decimal to BCD

(iii) Octal to Binary

(iv) Hex to binary



1) priority Encoder:-



If $I_3 = 1$ then we don't have I_0, I_1, I_2 due to H/p

I_3	I_2	I_1	I_0	Y_1	Y_0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

for Y_1 :-

I_3I_2 \ I_1I_0	00	01	11	10
00	X	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$Y_1 = I_2 + I_3$

for Y_2 :-

I_3I_2 \ I_1I_0	00	01	11	10
00	X	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

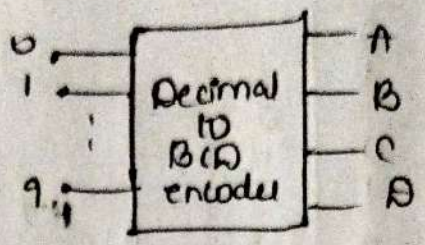
$Y_2 = I_3 + I_1\bar{I}_2$

Decimal to BCD encoder :-

Binary Coded

r = 10
0 to r-1
0 to 9

i/p	o/p D	C	B	A
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	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

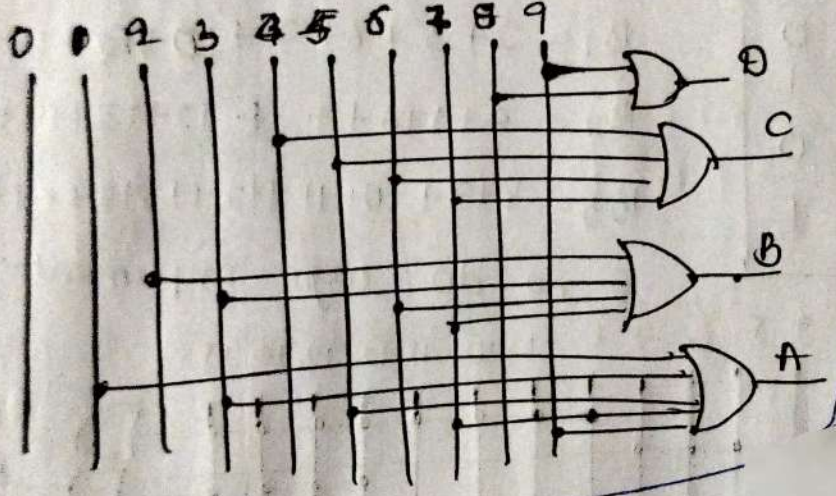


D = 8 + 9

C = 4 + 5 + 6 + 7

B = 2 + 3 + 6 + 7

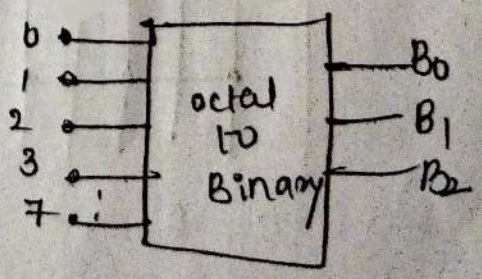
A = 1 + 3 + 5 + 7 + 9



Octal to Binary encoder

r = 8
0 to r-1
0 to 7

i/p	B_2	B_1	B_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1



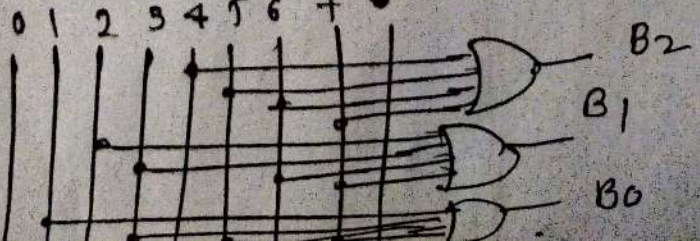
$(76)_8 \rightarrow (\quad)_2$

$(111\ 110)_2$

$B_0 = 4 + 5 + 6 + 7$

$B_1 = 2 + 3 + 6 + 7$

$B_0 = 1 + 3 + 5 + 7$



Hexadecimal to Binary Encoder

n = 16

0 to 15

ie

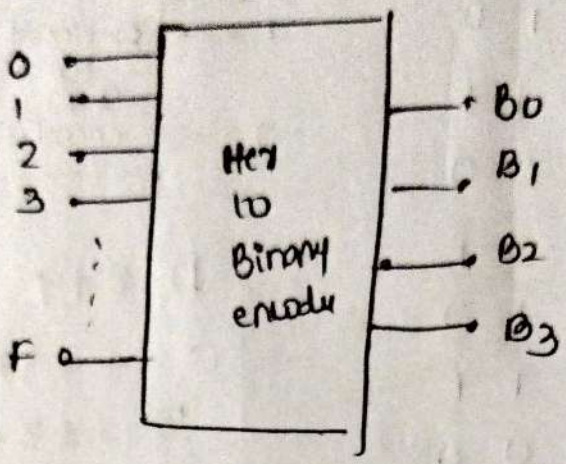
0 to F

n = 16

$m = \log_2 16$

$m = 4$

i/p	B_3	B_2	B_1	B_0
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	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1



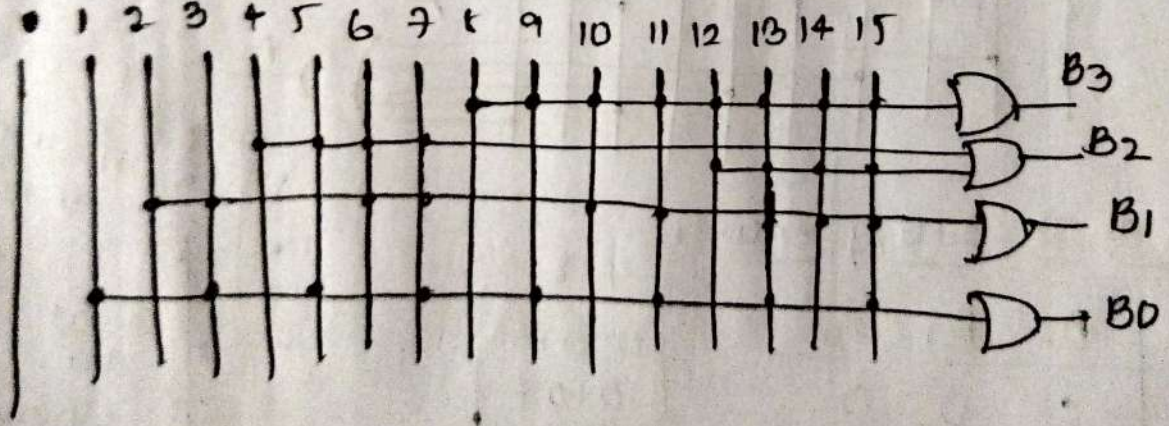
$B_0 = 1+3+7+9+11+13+15$

$B_1 = 2+3+6+7+10+11+14+15$

$B_2 = 4+5+6+7+12+13+14+15$

$B_3 = 8+9+10+11+12+13+14+15$

& no do logic implement



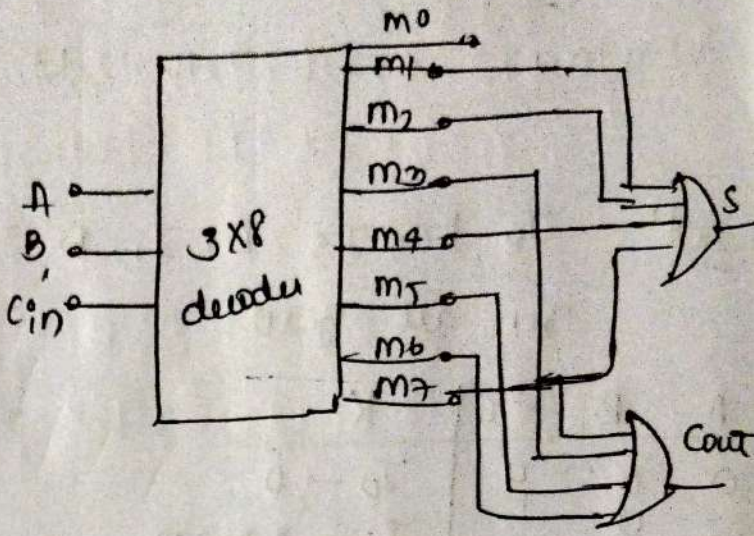
Full adder using decoder

adv: single IC but conventional method gates more cumbersome

space & cost saved.

Truth table

A	B	Cin	S	Cout	
0	0	0	0	0	m_0
0	0	1	1	0	m_1
0	1	0	1	0	m_2
0	1	1	0	1	m_3
1	0	0	1	0	m_4
1	0	1	0	1	m_5
1	1	0	0	1	m_6
1	1	1	1	1	m_7



$F_S = \Sigma m(m_1, m_2, m_4, m_7)$

$F_{Co} = \Sigma m(m_3, m_5, m_6, m_7)$

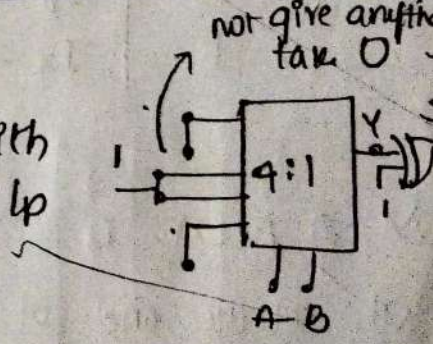
Q) The ckt shown does represents

a) $S(A,B) = \Sigma(1,2)$

b) $S(A,B) = \Pi(0,3)$

c) EXOR gate with A & B as i/p

d) ~~$\bar{A}\bar{B} + AB$~~



Ans

A B	Y	
0 0	0	m_0
0 1	1	m_1
1 0	1	m_2
1 1	0	m_3

$Y = \bar{A}\bar{B}\cdot 0 + \bar{A}B\cdot 0 + A\bar{B}\cdot 1 + AB\cdot 1$

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

$S = \bar{Y}\cdot 0 + Y1$

$S = \bar{A}B + A\bar{B}$ ✓

$Y = S = \Sigma m(1,2) = \Pi(0,3)$

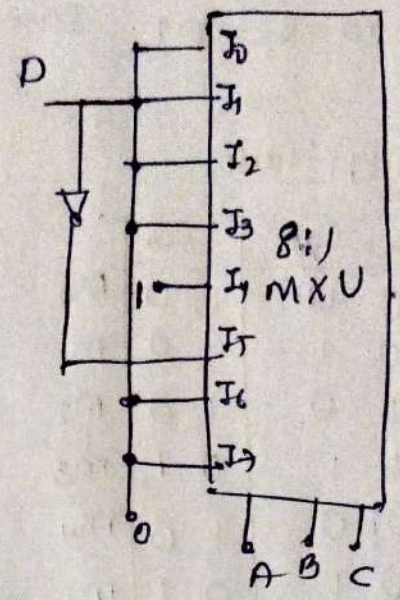
Q) The ckt below represents func $X(A,B,C,D)$ as

a) $\Sigma(3,8,9,10)$

b) $\Sigma(3,8,10,14)$

c) $\Pi(0,1,2,4,5,6,7,11,12,13,15)$

d) $\Pi(0,1,2,4,5,6,7,10,12,13,15)$



Sol: $I_4 = 1$, $I_5 = \bar{D}$, $I_1 = D$

Rest all zero

A	B	C	X
0	0	0	$I_0 \rightarrow 0$
0	0	1	$I_1 \rightarrow \bar{A}\bar{B}\bar{C}D$
0	1	0	$I_2 \rightarrow 0$
0	1	1	$I_3 \rightarrow 0$
1	0	0	$I_4 \rightarrow A\bar{B}\bar{C}$
1	0	1	$I_5 \rightarrow A\bar{B}\bar{C}\bar{D}$
1	1	0	$I_6 \rightarrow 0$
1	1	1	$I_7 \rightarrow$

m3 m10

0 0 1 1 1 0 1 0

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

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

1 0 0 1 + 1 0 0 0

m9 m8

$= \Sigma m(3,8,9,10)$

Q) A MUX is

a) selects one of the several i/p & transmits it to a single o/p

b) routes the data from a single i/p to one of many o/p.

c) Converts 11^{th} data to serial data.

d) Comb ckt

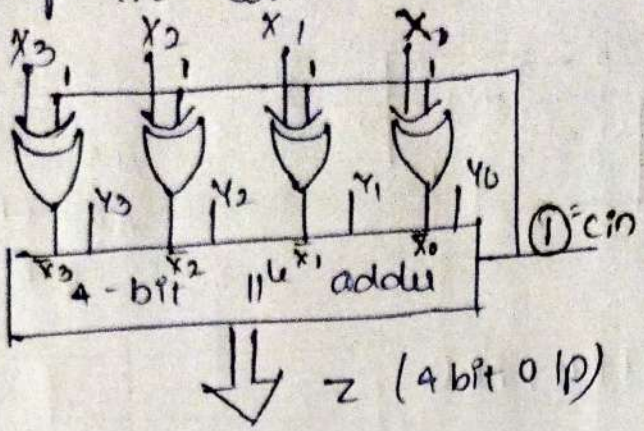
* Coincidence logic at same time

i/p ≥ 2

o/p $= 1$

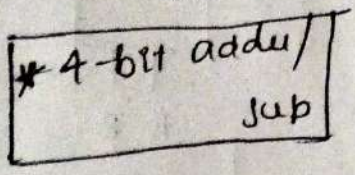
XNOR gate should be there.

Q. Identify the correct statement w.r.t to following ckt



a) x+Y
b) Y-X
c) X+1
d) Y+1

→ it can act as



4-bit 11's addu :-

$Y_0 + \bar{X}_0$ → $Y_0 - X_0$ & $Y_1 - X_1$, $Y_2 - X_2$, $Y_3 - X_3$

↓ 2's comp

adding 2's comp = sub

∴ output Y-X

$A \oplus B = A \oplus 1$

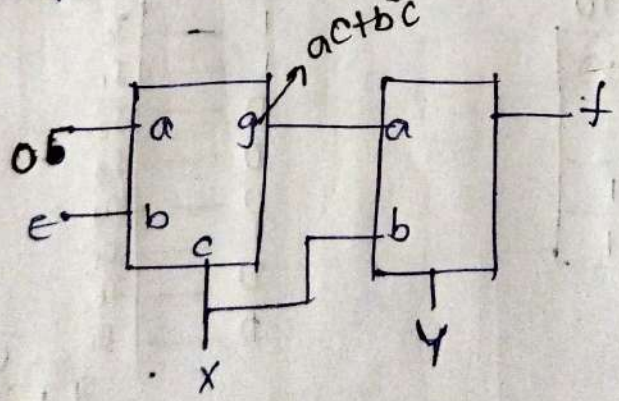
$\bar{A}\cdot 1 + 1 \cdot \bar{A}$

$\bar{A} \cdot 0$

$= \bar{A}$

$C_{in} = 1$ it acted like subtractor
If $C_{in} = 0$ it acts as adder

Q. Consider the ckt shown below. The o/p of 2:1 is given by the function $g = ac + b\bar{c}$



Then f is

a) X XOR Y
b) $X\bar{Y} + E\bar{X}Y$
c) $XE + X\bar{Y}$
d) None.

Sol:

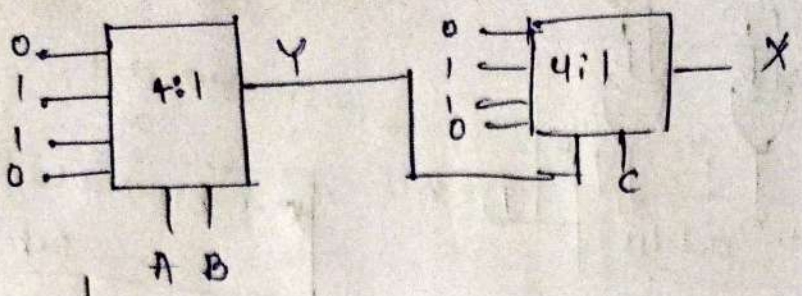
$f = \bar{Y}X + Y\bar{X}E$ if $c = 0$ $g = b$

$f = \bar{Y}X + E\bar{X}Y$ $c = 1$ $g = a$

$g = \bar{c}a + b\cdot c$

$g = \bar{c}b + ac$

Q4 X = ?



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

$Y = \bar{A}\bar{B}0 + \bar{A}B + A\bar{B} + AB0$

$Y = A \oplus B$

$X = \overline{A \oplus B} \cdot C + A \oplus B \cdot \bar{C}$

$X = \bar{A}B\bar{C} + A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC$ ✓

- - - - - -

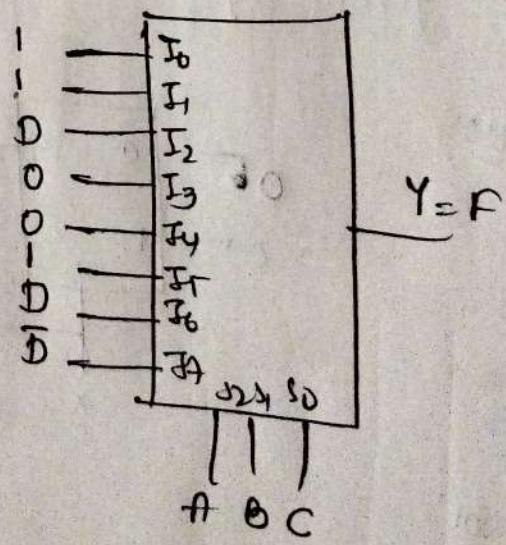
Q: Implement the following boolean algebra expression using an 8:1 MUX

$f(A,B,C,D) = \Sigma m(1,3,5,10,11,13,14) + d(0,12)$

Sol:

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

A	B	C	S_2	S_1	S_0	Y
			0	0	0	$I_0 = 1$
			0	0	1	$I_1 = 1$
			0	1	0	$I_2 = D$
			0	1	1	$I_3 = 0$
			1	0	0	$I_4 = 0$
			1	0	1	$I_5 = 1$
			1	1	0	$I_6 = D$
			1	1	1	$I_7 = \bar{D}$



Q6 Design a comb ckt with 3 i/ps X, Y & Z and 3 o/ps A, B, C when the binary i/p is 0, 1, 2 or 3, the binary o/p is 2 greater than the i/p when the binary i/p is 4, 5, 6 or 7 the binary o/p is 0 less than the i/p.

i/p	X	Y	Z	A	B	C
0	0	0	0	0	1	0
1	0	0	1	0	1	1
2	0	1	0	1	0	0
3	0	1	1	1	0	1
4	1	0	0	0	1	0
5	1	0	1	0	1	1
6	1	1	0	1	0	0
7	1	1	1	1	0	1

Kmap for A:

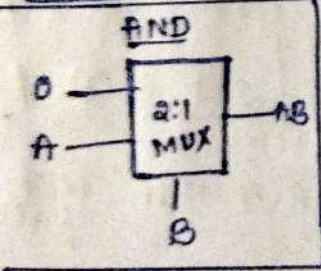
X_0 \ YZ	00	01	11	10
0			1	1
1			1	1

or directly $A = Y$

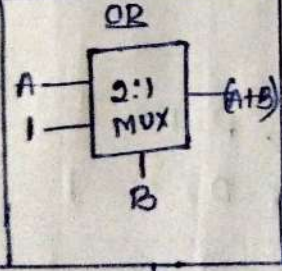
$B = \bar{Y}$, $C = Z$

MUX as universal Logic Circuit :-

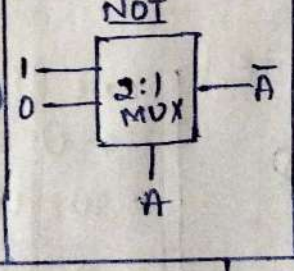
AND



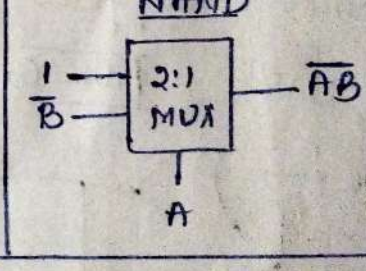
OR



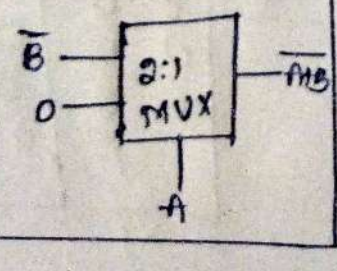
NOT



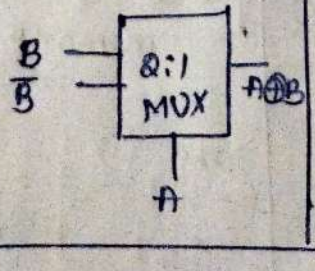
NAND



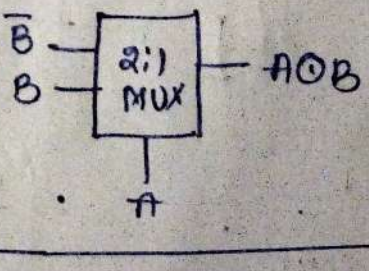
NOR



EXOR

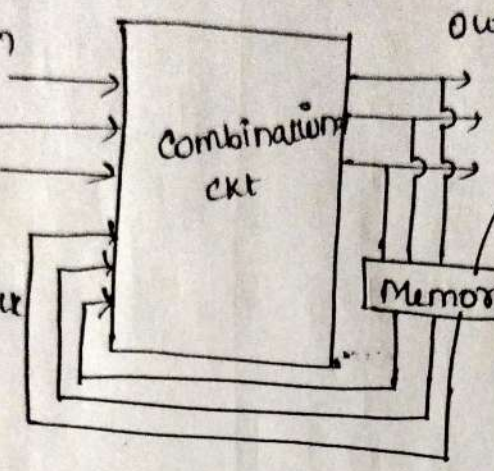


EXNOR



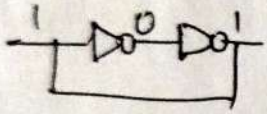
Sequential Circuits:-

→ The sequential circuits, the present o/p depends on the present i/p as well as past o/p or o/p's



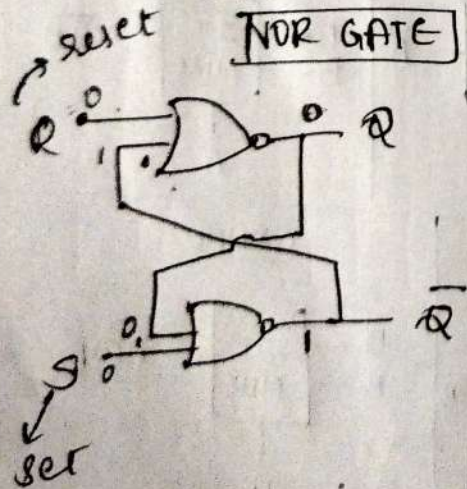
In sequential ckt everything is combckt except memory/ feedback

eg



SR-latch < NAND GATE / NOR GATE

→ The basic storage element is called LATCH, as the name suggests it latches "0" as "1"



reset → Q=0
set → Q=1

T.T for NOR gate:

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

Case (i)

S=0, R=1

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

S=0, R=0

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

memory

Case (ii)

S=1, R=0

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

S=0, R=0

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

memory

Case (iii)

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

S=0, R=0, contradiction

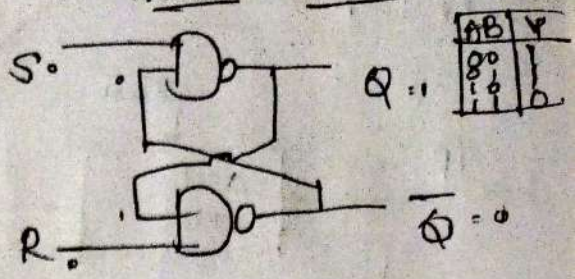
Not used

for S=0, R=0 2 possible o/p's: $\bar{Q}$=1, Q=0 / $\bar{Q}$=0, Q=1 | not good

for NOR gate

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

NAND GATE



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

Case (i):

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

not used. S = 0, R = 0 → not valid.

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

memory

Case (ii)

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

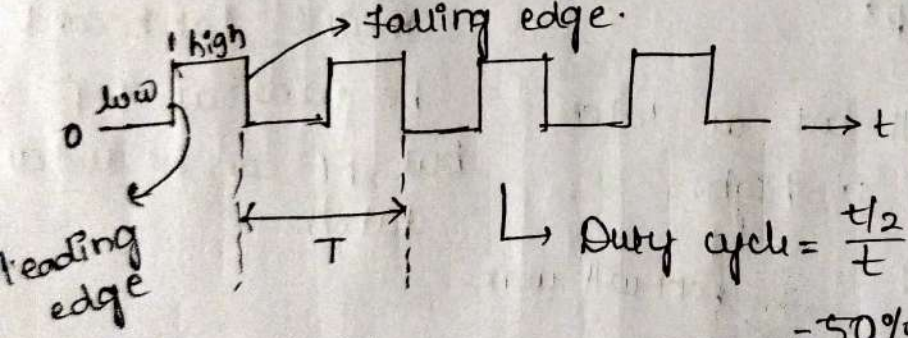
S = 1, R = 1, Q = 0, $\bar{Q}$ = 1

memory

not used S = 0, R = 0 → not valid

→ cannot i/p

What is a Clock:- signal



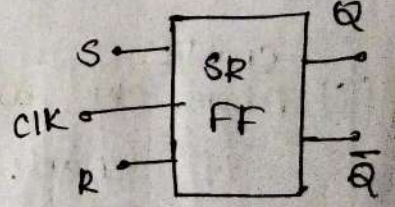
Duty cycle = $\frac{t/2}{t}$

= 50%.

$f = 1/T$

To maintain CKT high speed f ↑ so T ↓

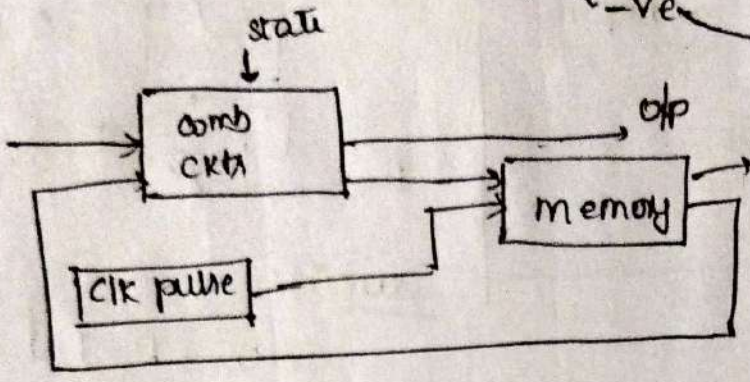
eg: CKT will work only if clk is high.

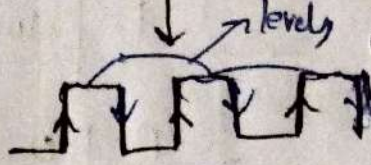


Duty cycle:- Ratio of time signal high to total time

$$D = \frac{\text{time signal high}}{\text{total time}} \times 100\%$$

Triggering Methods:- level (when ever signal high change/transition in latch/FF)
edge +ve
-ve

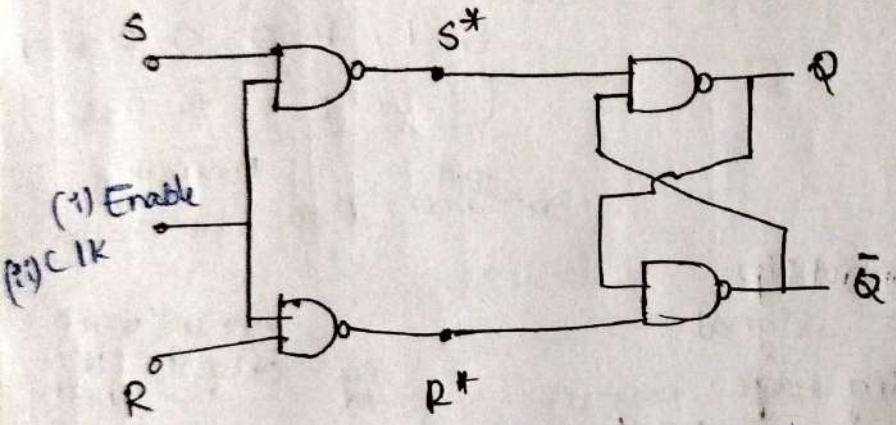




level sensitive → enable
edge sensitive → clk.

+ve → Low to high
-ve → high to low

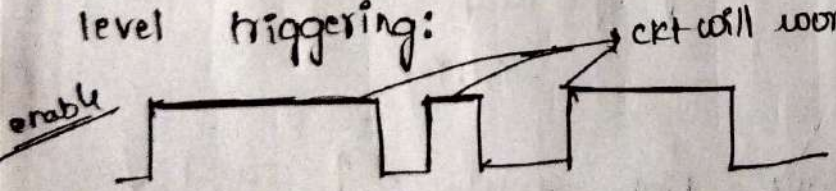
Difference btw Latch & flipflops:-



In latch we don't have control signal (clk) but in flipflop we have clk means latch can't be used as F.F but FF can be used as latch.

Enable
(i) Above ckt will act as latch if enable is high.

level triggering:



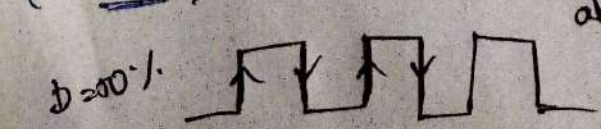
because $S^* = \bar{S} + \bar{En}$

If $En = 1$, $S^* = \bar{S}$

if $En = 0$, $S^* = 1$, $R^* = 1$ → memory

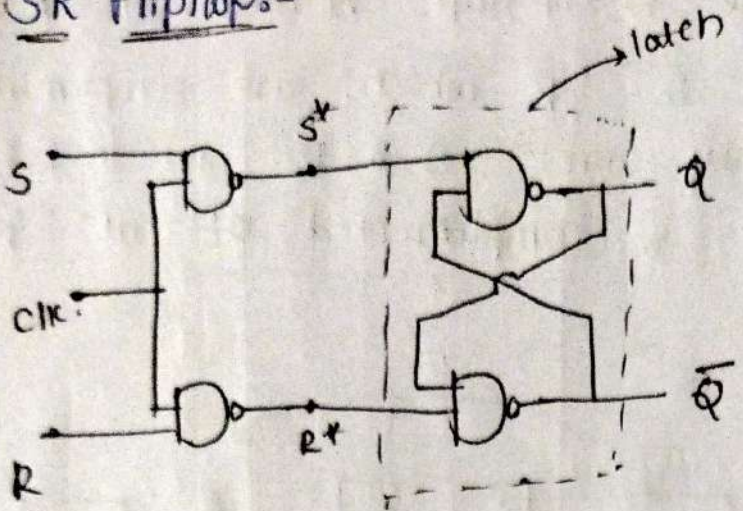
SR latch ✓

(ii) CLK :-



above ckt is FF when there is CLK & edge triggering

SR Flipflops:-

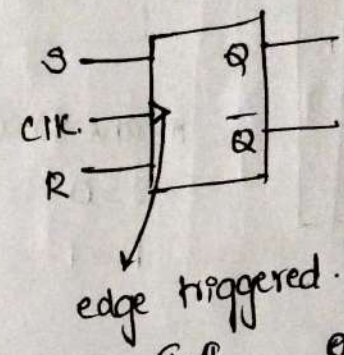


SR latch with NAND

S^*	R^*	Q $\bar{Q}$
0	0	not used
0	1	1 0
1	0	0 1
1	1	Memory

CLK is used to prevent ckt from changing accidently, it will change when ever we need to change

$$S^* = \overline{S \cdot CLK} = \bar{S} + \overline{CLK}$$
$$R^* = \overline{R \cdot CLK} = \bar{R} + \overline{CLK}$$ → (1)



edge triggered.

Offical TT:

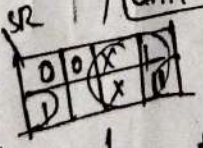
CLK	S	R	Q_{n+1} next state
0	X	X	Q_n → previous state
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	invalid

T.T a/c to (1)

* CLK	S	R	Q $\bar{Q}$
0	X	X	memory
1	0	0	Memory
1	0	1	0 1
1	1	0	1 0
1	1	1	Not used

Memory → $Q_n = Q_{n+1}$

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



* Characteristic table:

CLK = 1

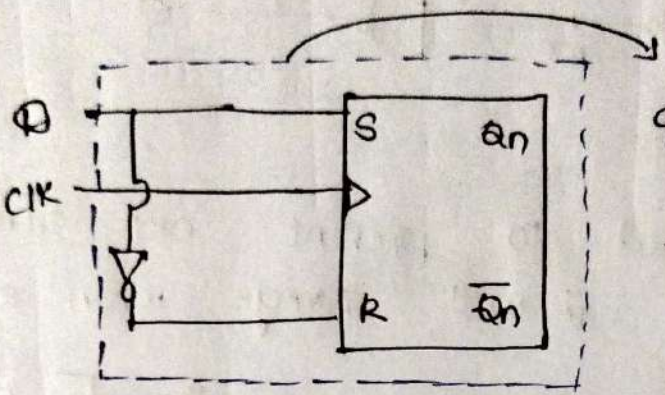
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Q_n invalid X
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	X

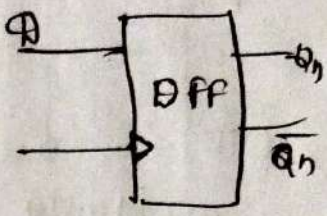
* Excitation table:-

Q_n	Q_{n+1}	SR
0	0	0 X
0	1	1 0
1	0	0 1
1	1	X 0

D flip flop:- we use D flip-flop because in SR FF we need to give 2 i/p which are complemented to each other they are S & R. so in D.FF we will give one i/p & complemented of that i/p

D means Data





T.T :-

CLK	D	Qn+1
0	X	Qn
1	0	0
↓	1	1

means D=0 | D=1
S=0 | S=1
R=1 | R=0

SR FF

CLK	S	R	Qn+1
0	X	X	Qn
1	0	0	Qn
1	0	1	0
1	1	0	1
1	1	1	invalid

char table:

CLK=1

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

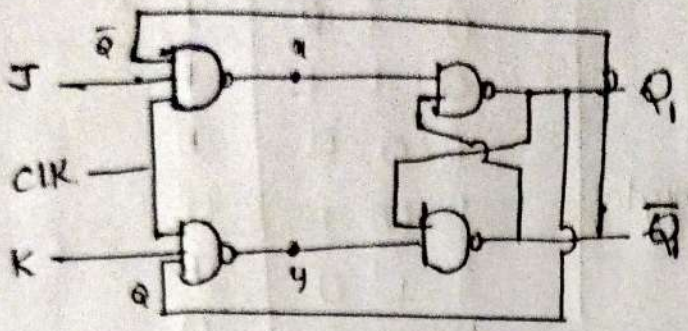
$Q_{n+1} = D$

Excitation table:-

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

JK flipflop:-

In SR & D FF have invalid case, but we will overcome that problem.



case(i):-

CLK = 0, J = 0, K = 0

x = 1, y = 1 (HL)

$x = \overline{CLK} + \bar{J} + \overline{\bar{Q}}$

$= \overline{CLK} + \bar{J} + Q$

$= 1 + \bar{J} + Q$

$= 1$

Ny $y = 1$ (from NAND SR latch)

from, x = 1, y = 1

Q, $\bar{Q}$ auth as memory

case(ii):-

CLK = 1, J = 0, K = 1, $Q_1 = 0$ $\bar{Q}_1 = 1$

$x = \overline{CLK} + \bar{J} + \bar{Q}$ | $y = \overline{CLK} + K + \bar{Q}$

$x = 0 + 1 + Q$ | $y = 0 + 0 + \bar{Q}$

$x = 1$ | $y = \bar{Q}$ if Q = 1

$y = 0$

$\bar{Q}$ & (y) 10, from SR latch

S = 1, R = 0

$Q_1 = 0$, $\bar{Q}_1 = 1$

Ny case(iii)

CLK = 1, J = 1, K = 0, $Q_1 = 1$, $\bar{Q}_1 = 0$

as S = 0, R = 1

Q = 0, 1, 0, 1, 0 - - -

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

case(iii):-

CLK = 1, J = 1, K = 0

assume) Q = 0 & $\bar{Q} = 1$

so, $x = \overline{1 \cdot J \cdot CLK}$

$= \bar{1} + \bar{J} + \overline{CLK}$

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

$= 0$

$x = 0$ · = S

$y = \overline{D \cdot K \cdot CLK}$

$= 0 + 0 + 1$

$y = 1$: R

so, $Q_1 = 1$, $\bar{Q}_1 = 0$

now, 2) Q = 1, $\bar{Q}_1 = 0$

then, x = 1, y = 0

$Q_1 = 0$, $\bar{Q}_1 = 1$

Iterations so on

Truth table:

clk	J	K	Q_{n+1}
0	X	X	Q_n } memory
0	0	0	Q_n } memory
1	0	1	0
1	1	0	1
1	1	1	$\overline{Q_n}$ (toggle)

Char table:-

clk=1

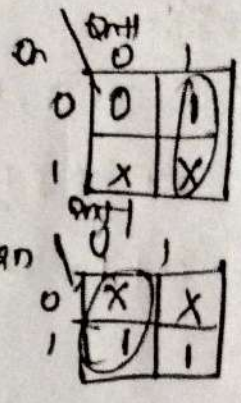
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

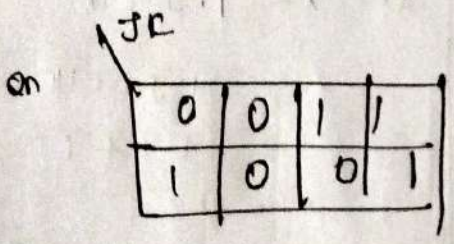
Excitation table:-

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

for $J = Q_{n+1}$

for $K = \overline{Q_{n+1}}$



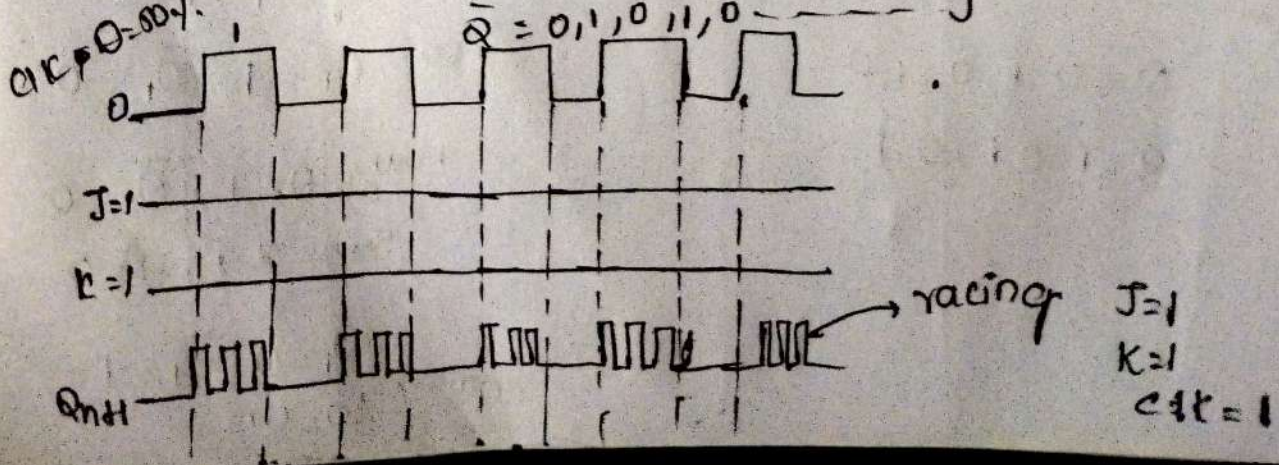


$Q_{n+1} = \overline{Q_n} J + Q_n \cdot \overline{K}$

Race around cond or racing in JK flip flop:-

as we see, $Q = 1,0,1,0,1,---$ } o/p is racing.

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



Conditions to overcome racing:-

(i) $T/2$ < propagation delay of FF → not practical, so not used

(ii) Instead of level trigg use edge trigg

(iii) Master slave.

* imp

racing =

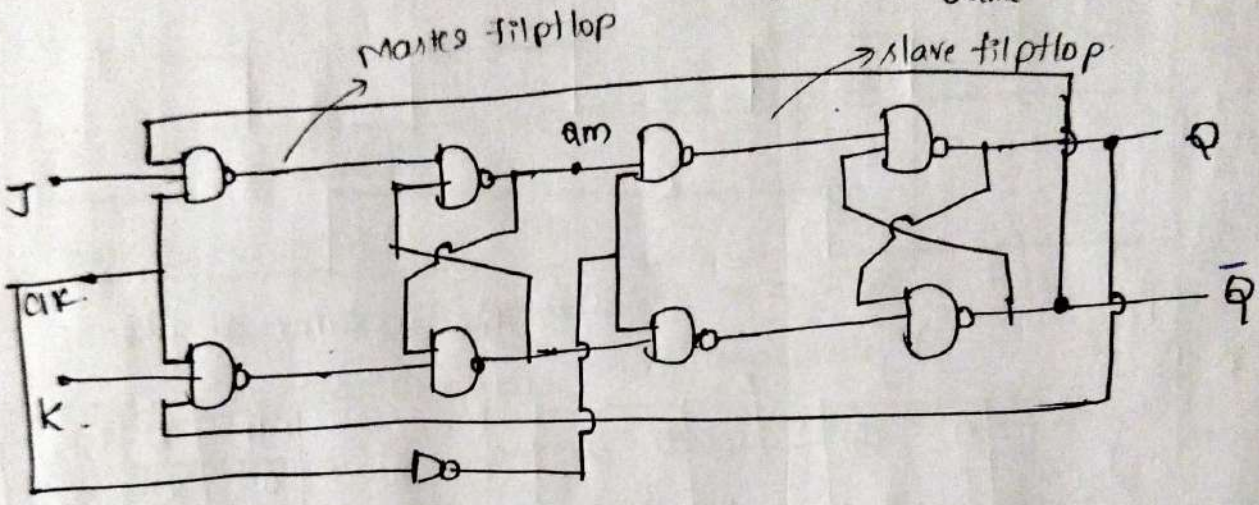
toggling =

Master slave of JK flipflop:-

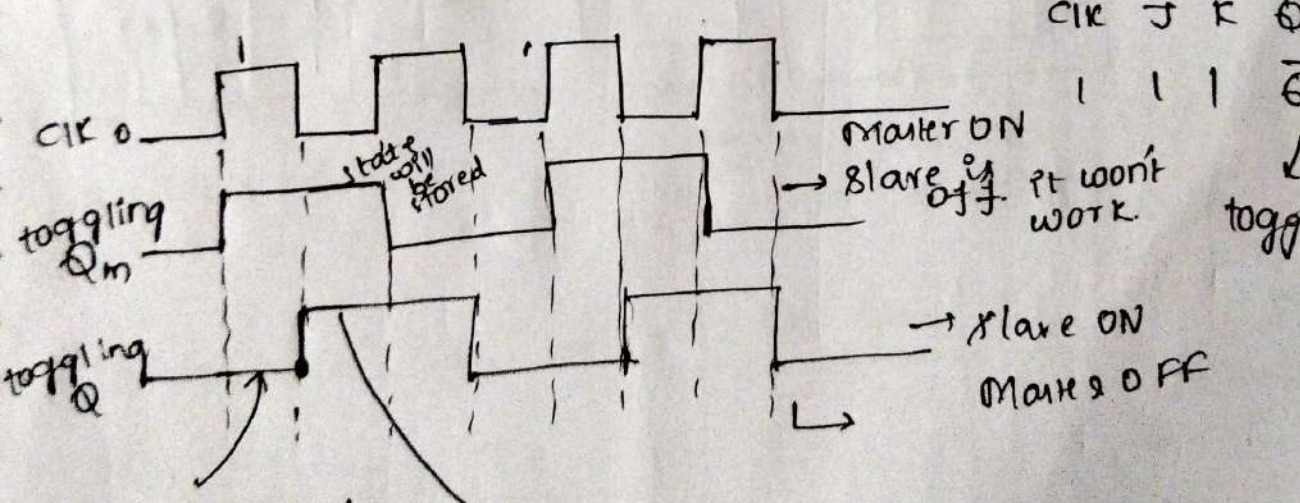
racing is uncontrolable so it should toggle means which is controllable

edge triggering → +ve edge triggering

→ -ve edge triggering ⇒ same Master slave



clk	J	K	Q_{n+1}
↓	1	1	$\bar{Q}$ ↓ toggling

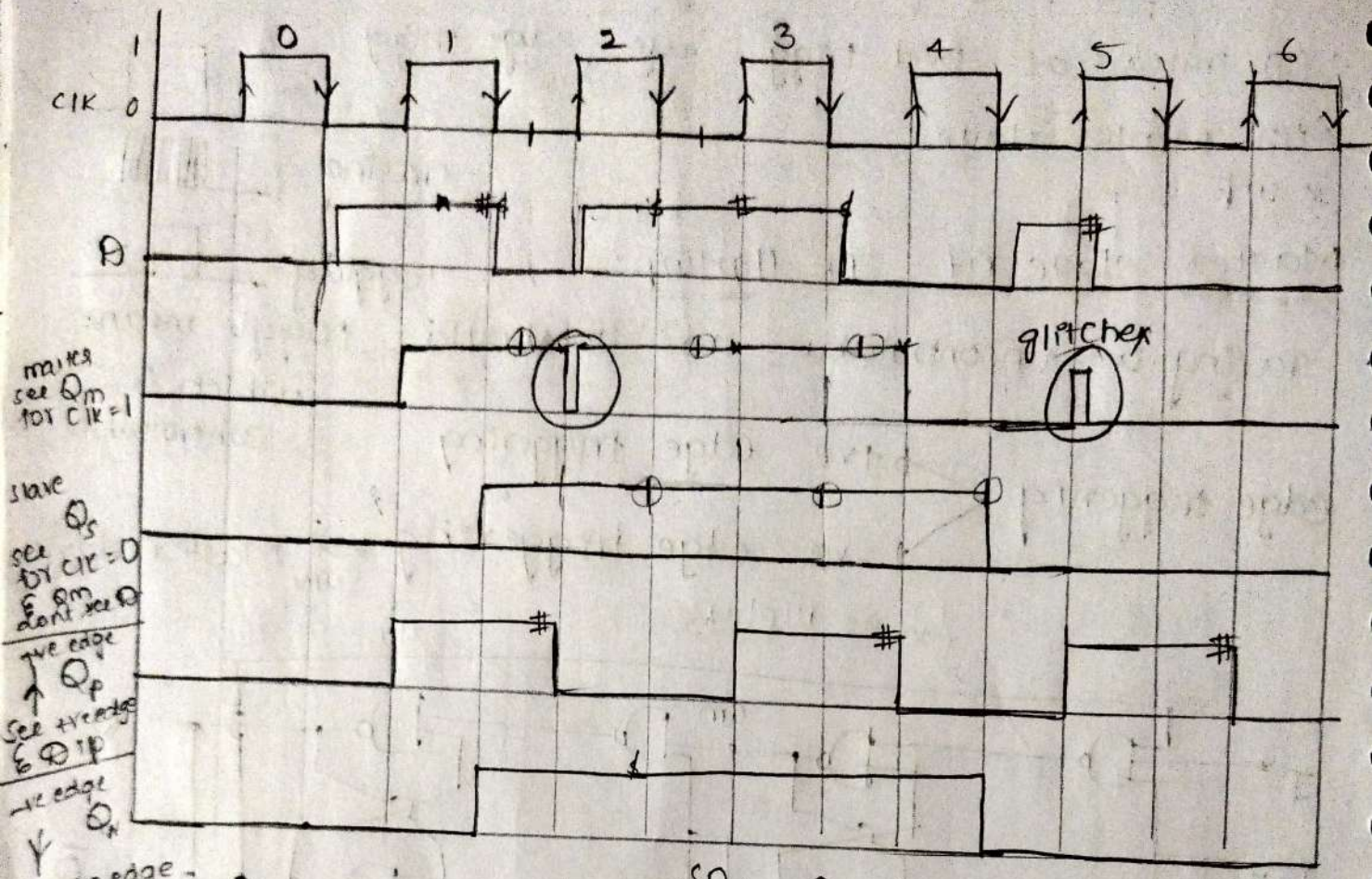


Q=0 when it clk=1

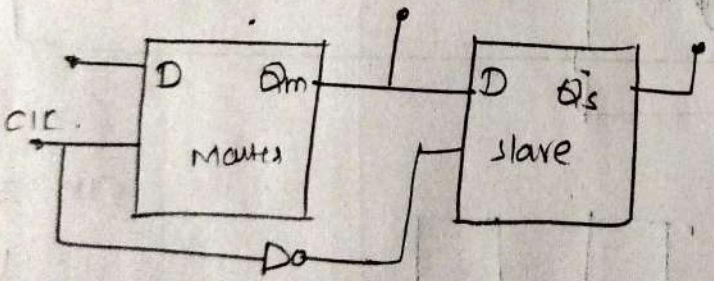
like Slave is off initially

now clk=0 Slave is ON

Master slave D flipflop:-



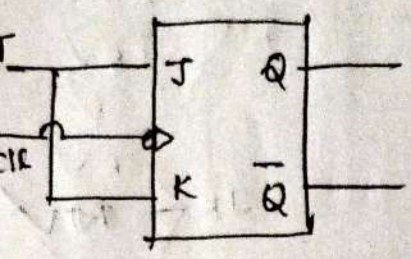
So, Q_s is same as Q_N



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

T - flipflop:-

↳ toggle.



T.T for T FF

clk	T	Q_{n+1}
0	X	Q_n (mem)
1	0 (T=L=0)	Memory
1	1 (J=L=1)	$\bar{Q}_n$

char table clk=1

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

Odd i/p detector

X-OR

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

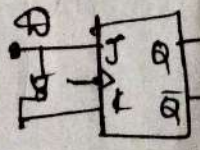
Excitation table

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

Filpflop conversion:-

1) Identify available & req f.f
2) Make characteristic table for req F.F
3) Make excitation table for available FF
4) write boolean expression for available FF
5) Draw the ckt

v)



Eg: JK to D filpflop conversion

i) available FF = JK

req FF = D

(ii) Characteristic table of D

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

(iii) Excitation table for JK FF

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

write JK then

iv) for J

Q_n \ D	0	1
0	0	1
1	X	X

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

for K

Q_n \ D	0	1
0	X	X
1	1	0

$L = \bar{Q}_{n+1} = \bar{D}$

SR to D flipflop

available → req.

(i) char table of D

Q_n	D	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	1	0
1	0	0	0	1
1	1	1	X	0

(ii) excitation table of SR

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

D ✓ → SR ✓, JK ✓, T ✓

(iv) for S:-

Q_n \ D	0	1
0	0	1
1	0	X

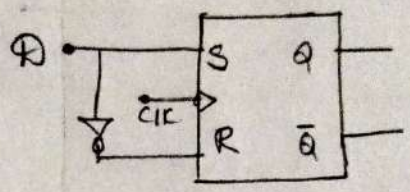
$S = D$

for R:-

Q_n \ D	0	1
0	X	0
1	1	0

$R = \overline{D}$

(v)



SR to JK

available → req

(i) char of JK :- (ii)

Q_n	J	K	Q_{n+1}	S R
0	0	0	0	0 X
0	0	1	0	0 X
0	1	0	1	1 0
0	1	1	1	1 0
1	0	0	1	X 0
1	0	1	0	0 1
1	1	0	1	X 0
1	1	1	0	0 1

(iii) excitation table of SR:-

(iv) Boolean expression for S:-

Q_n \ JK	00	01	11	10
0	0	0	1	1
1	X	0	0	X

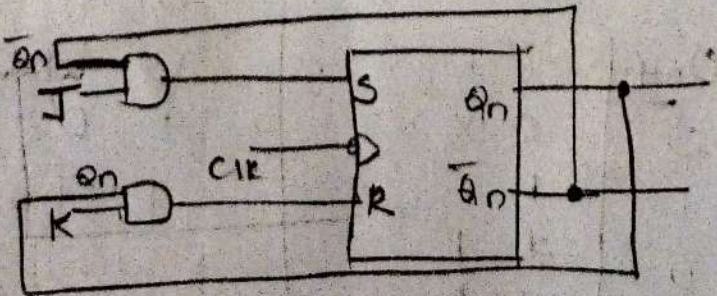
$S = \overline{Q_n} J$

for R:-

Q_n \ JK	00	01	11	10
0	X	X	0	0
1	0	1	1	0

$R = Q_n K$

(v)



SR to T :

i) available ii) req

ii) Char for T

Qn	T	Qn+1	S	R
0	0	0	0	X
0	1	1	1	0
1	0	1	X	0
1	1	0	0	1

(iii) excitation table for SR :-

Qn	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	X	0
1	1	0	1

iv) for S :-

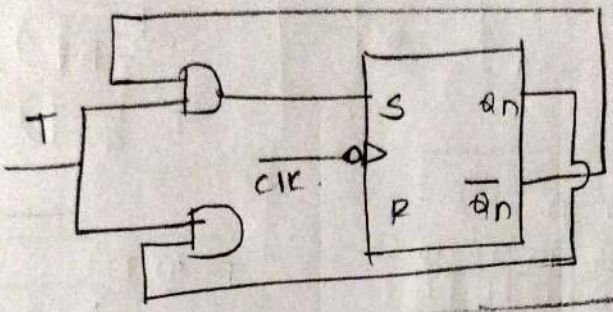
Qn \ T	0	1
0	0	(1)
1	X	0

$S = \overline{Q_n} T$

for R :-

Qn \ T	0	1
0	X	0
1	0	(1)

$R = Q_n T$



JK to SR :-

i) available ii) req

ii) Char SR :-

Qn	S	R	Qn+1	J	K
0	0	0	0	0	X
0	0	1	0	0	X
0	1	0	1	1	X
0	1	1	X	X	X
1	0	0	1	X	0
1	0	1	0	X	1
1	1	0	1	X	0
1	1	1	X	X	X

(iii) ∈ T JK :-

for J :

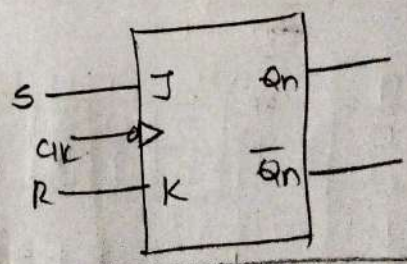
Qn \ SR	00	01	11	10
0	0	0	X	1
1	X	X	X	X

$J = S$

for K :

Qn \ SR	00	01	11	10
0	X	X	X	X
1	0	1	1	0

$K := R$



JK to T FF :

available req

Char T :

Qn	T	Qn+1	JK
0	0	0	0X
0	1	1	1X
1	0	1	X0
1	1	0	X1

∈ T JK :-

for J :-

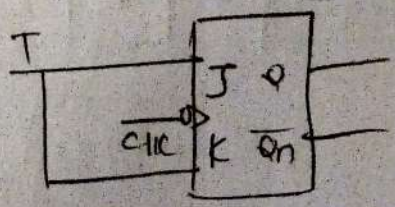
Qn \ T	0	1
0	0	(1)
1	X	(X)

$J = T$

for K :

Qn \ T	0	1
0	X	(X)
1	0	(1)

$K = T$



D to SR:

(i) available (ii) req

char SR - Exc Tab D

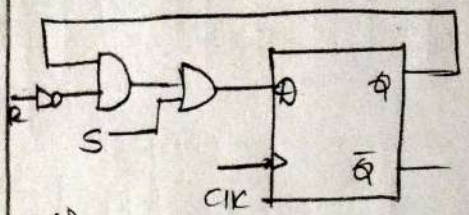
Q_n	S	R	Q_{n+1}	D
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	X	X
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	X	X

(iv) for D:

Q_n \ SR	00	01	11	10
0	0	0	X	1
1	1	0	X	1

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

(v)



D to T:

(i) available (ii) req

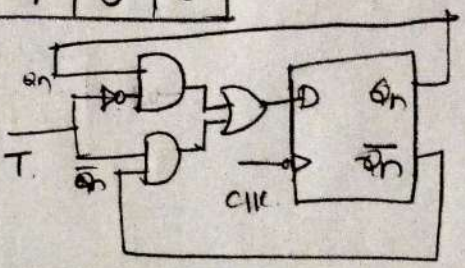
char T - E.T D

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

(iv) for D:

Q_n \ T	0	1
0	0	1
1	1	0

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



D to JK:

available req

char JK - Exc Tab D

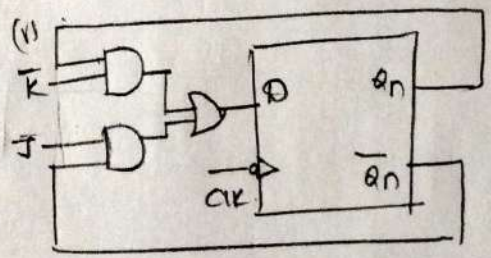
Q_n	J	K	Q_{n+1}	D
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	0	0

(iv) for D;

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

$D = \bar{Q}_nJ + Q_n\bar{K}$

(vi)



T to D:

(i) available req

char D - E.T T

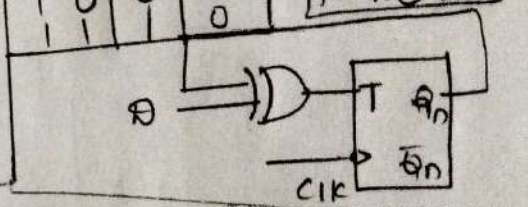
Q_n	D	Q_{n+1}	T
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	0

(iv) for T:

Q_n \ D	0	1
0	0	1
1	1	0

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

$T = D \oplus Q_n$



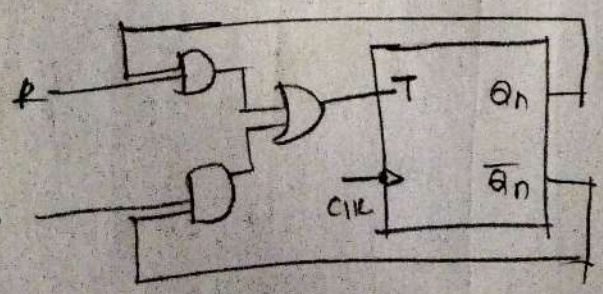
T to SR:

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

for T:-

Q_n \ SR	00	01	11	10
0	0	0	X	1
1	0	1	X	0

$T = Q_nR + S\bar{Q}_n$



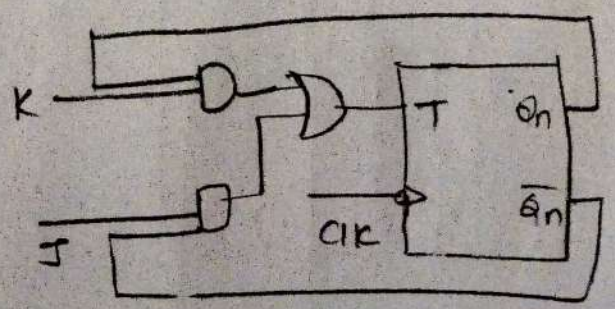
T to JK:-

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

for T;

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

$T = Q_nK + \bar{Q}_nJ$



Present and clear inputs:-

→ They are direct i/p's or overriding i/p's or asynchronous i/p's.

→ The synchronous i/p are S, R, J, K, D & T

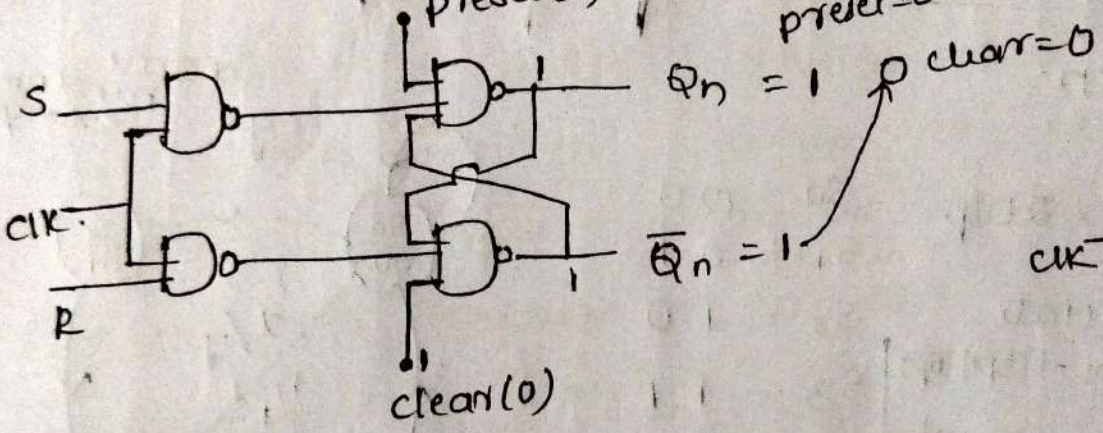
preset = 0 ⇒ $Q_n = 1$

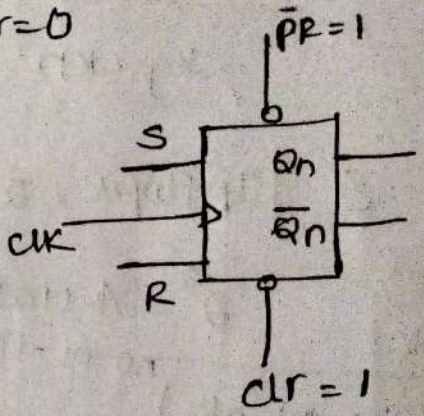
clear = 0 ⇒ $Q_n = 0$ because $\bar{Q}_n = 1$

↓

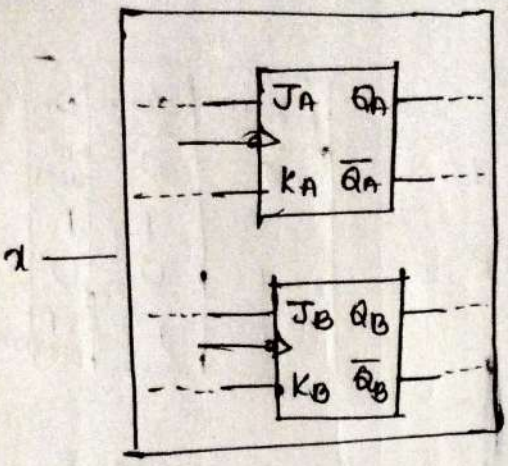
what ever be the value of clock & synchronous i/p

preset	clear	Q_n
0	0	not used
0	1	1
1	0	0
1	1	FF normal





State table: — A table tells us the relation btw present state, next state and o/p.



Seq CKT

State table:-

PS		x	NS		y	
Q_A	Q_B		Q_A^+	Q_B^+		
0	0	1	1	0	1	→ taken randomly

2 filpflops, 2 o/p

2^2 = 4 state

~~no of filpflops~~

(no. of o/p) = no of state

$S_0 = 00$

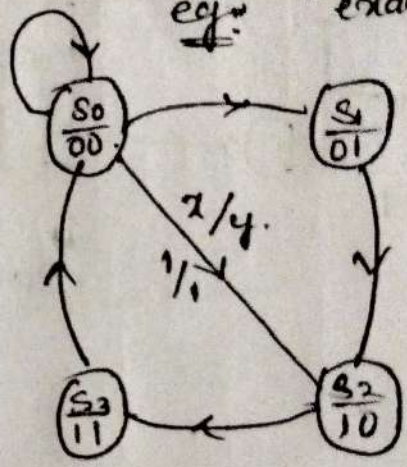
$S_1 = 01$

$S_2 = 10$

$S_3 = 11$

some random example

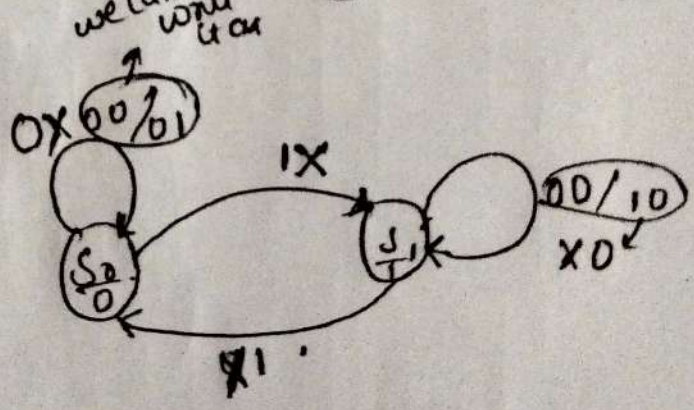
eg:-



for JK FF:-

PS Q_n	J	K	N.S Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

we can write it as



$(2)^1$ = 2 state

$S_0 = 0, S_1 = 1$

State Eq:

LHS = RHS

Q_{n+1} P.S & i/p

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

* Design proudure for clked Sequntial ckt :-

(i) A state diagram or timing diagram is given, which describles the behaviour of the ckt that is be deigned.

(ii) Obtain the state table

(iii) The no. of states can be redued by state reduction method.

(iv) Do state assignment (if req) | Letter symbols & assign

(v) Determine the no. of flipflops req

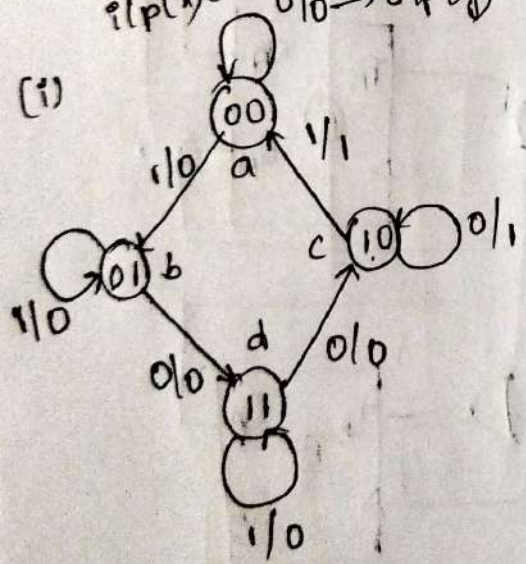
(vi) Decide the type of flipflop to be used.

(vii) Derive the ckt excitation table from state table.

(viii) Obtain the expression for ckt o/p & flipflop i/p

(ix) Implement the ckt.

(i)



(ii)

State table :-

PS		NS $x=0, x=1$		o/p (y)	
Q_A	Q_B	$Q_A^+ Q_B^+$	$Q_A^+ Q_B^+$	$x=0$	$x=1$
0	0	0 0	0 1	0	0
0	1	1 1	0 1	0	0
1	0	1 0	0 0	1	1
1	1	1 0	1 1	0	0

(iii) no reduction because no 2 rows are similar.

(iv) State allignment
a, b, c, d to binary already given

(v) 4 states ◇ 2 → 2 FF → A & B FF FF
2 → o/p
o/p o/p
Q_A Q_B

(vi) 'T' FF

means skip this point.

(vii) ckt excitation table :-

PS Q_A	PS Q_B	I	NS Q_A^{+1}	NS Q_B^{+1}	flip flop T_A	flip flop T_B	y
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	0	1	0	0	0
1	0	0	1	0	0	0	1
1	0	1	0	0	1	0	1
1	1	0	1	0	0	1	0
1	1	1	1	1	0	0	0

sel Q_A & Q_A^+

sel Q_B & Q_B^+

A → T flip flop
B →

Clk	T	Q_{n+1}
0	X	Q_n
1	0	Q_n
1	1	$\overline{Q_n}$

↓ excitation table

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

(viii)

$$T_A = \overline{Q_A} Q_B \bar{x} + Q_A \overline{Q_B} x$$

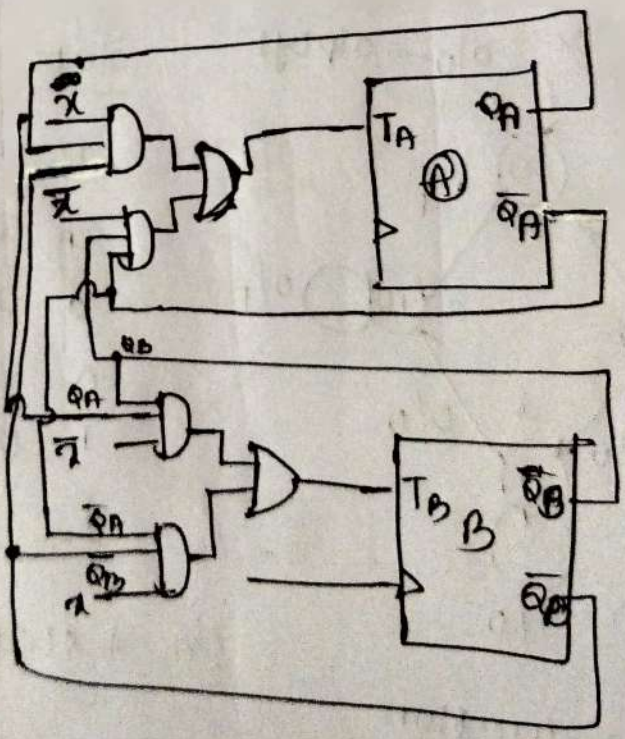
$$T_B = \overline{Q_A}\,\overline{Q_B} x + Q_A Q_B \bar{x}$$

$$y = \overline{Q_A}\,\overline{Q_B} \bar{x} + Q_A \overline{Q_B} x$$

$$= Q_A \overline{Q_B} (\bar{x} + x)$$

$$\boxed{y = Q_A \overline{Q_B}}$$

(ix)



Mealy & Moore State Machine :—

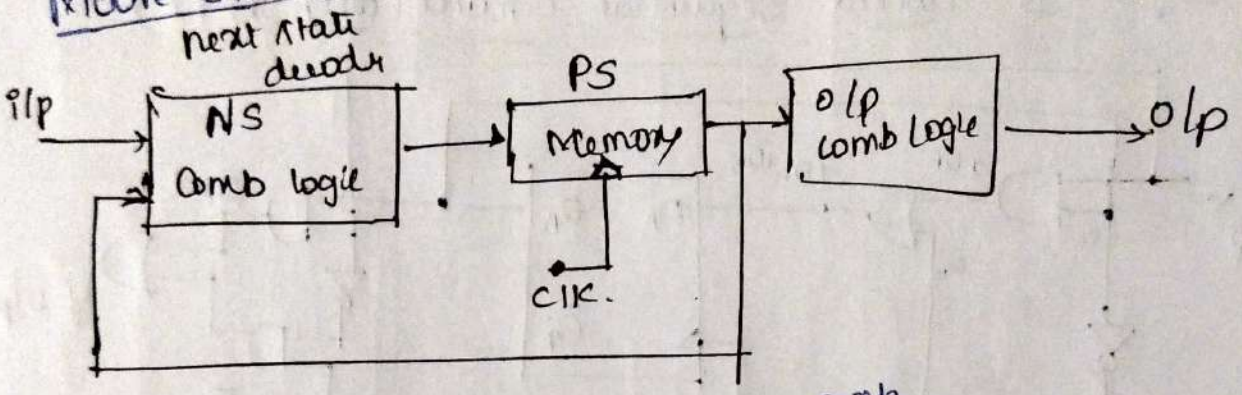
→ There are 2 models developed for representing synchronous seq ckts.

(i) Moore ckt / Moore state Machine :- o/p is the fun of ps only.

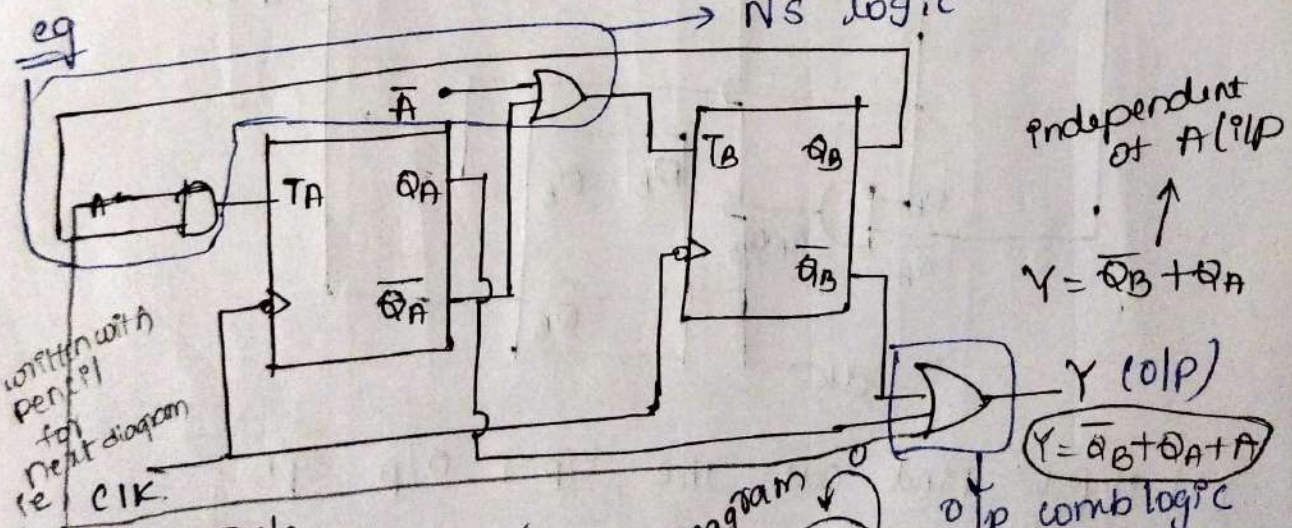
(ii) Mealy ckt / Mealy " " :- o/p is func of o/p & i/p.

* Diff bno these 2 ckts is only in the way the o/p is generated

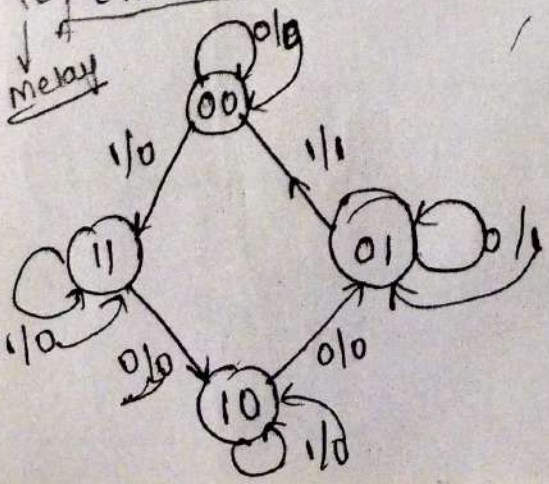
Moore State Machine : (only o/p is func of i/p)



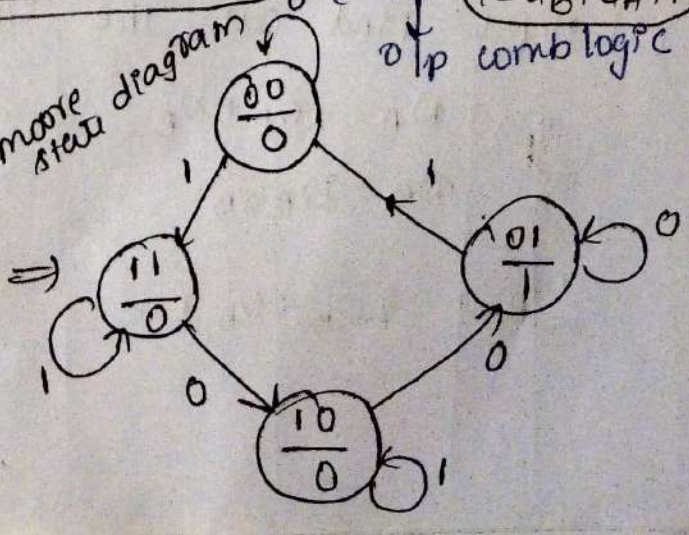
eg



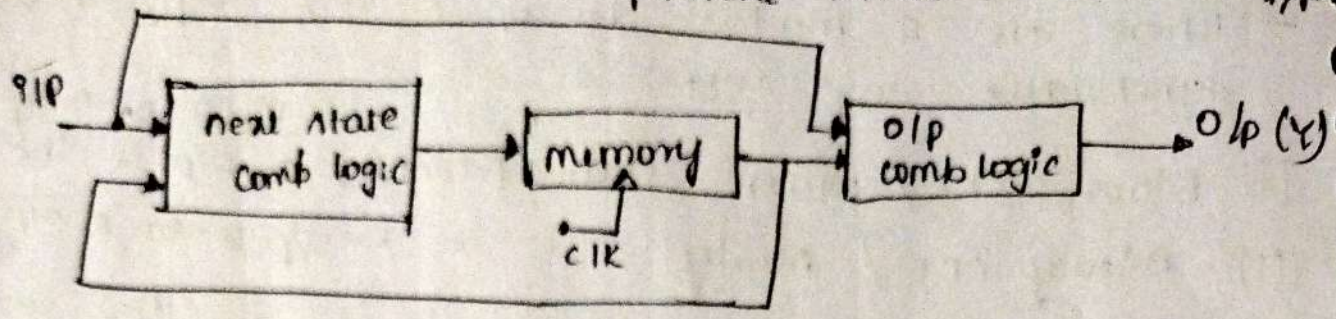
Mealy



moore state diagram



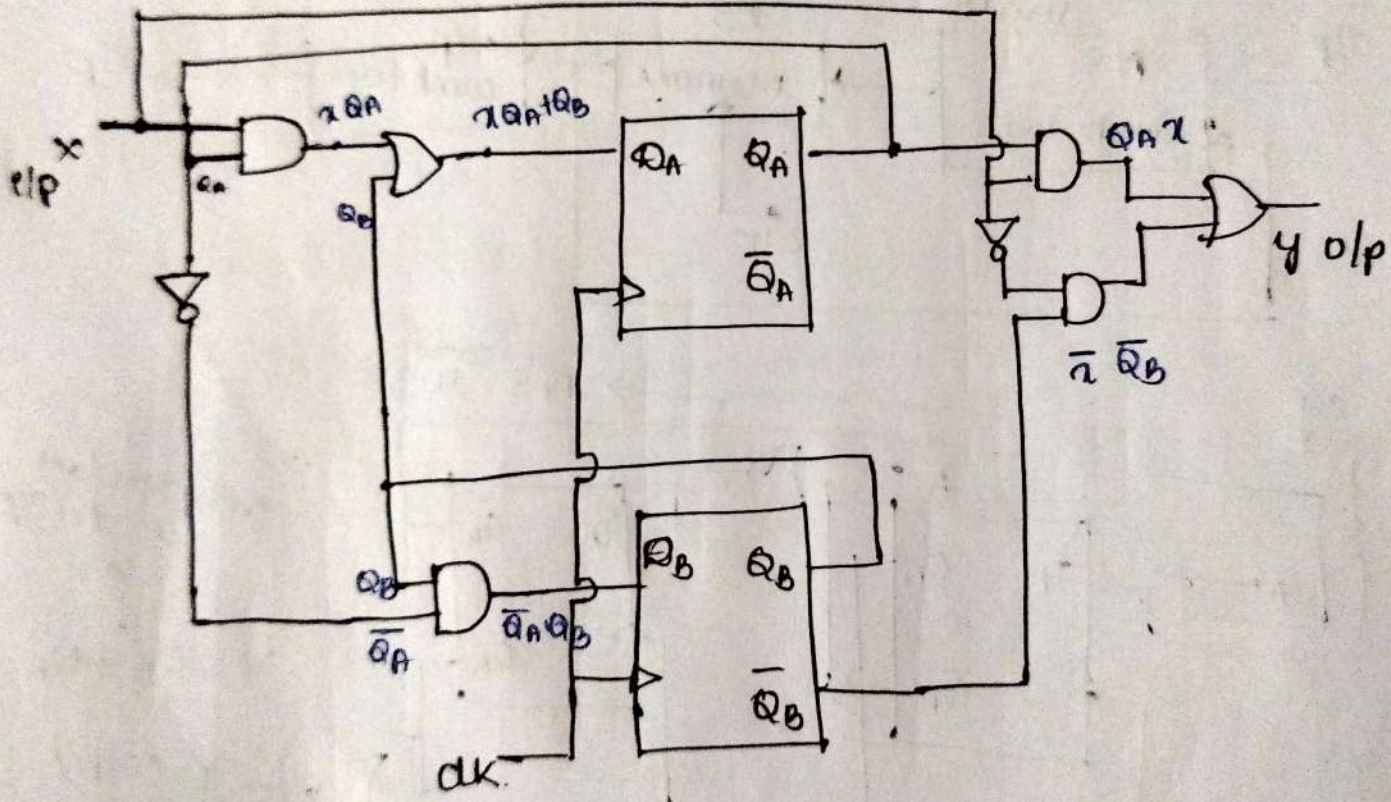
Mealy State machine :- The o/p is the func of present state as well as i/p



eg See prev example work with pencil

$$Y = \bar{Q}_B + Q_A + A$$

Analysis of clocked Sequential Circuit (with D FF)



(i) step 1 : find out the i/p & o/p eqn

i/p :- $D_A = xQ_A + Q_B$

$D_B = \bar{Q}_A Q_B$

o/p :- $\bar{x}\bar{Q}_B + Q_A x$

Step 2: State table

PS			NS		
Q_A	Q_B	x	Q_A^+	Q_B^+	y
0	0	0	0	0	1
0	0	1	0	0	0
0	0	0	1	1	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	0	0

2 dilpflops, 80 4 states

$D = Q_{n+1}$

$Q_A^+ = D_A$

$Q_B^+ = D_B$

$Q_A^+ = D_A = x Q_A + Q_B$

$= 0 \cdot 0 + 0$

$= 0$

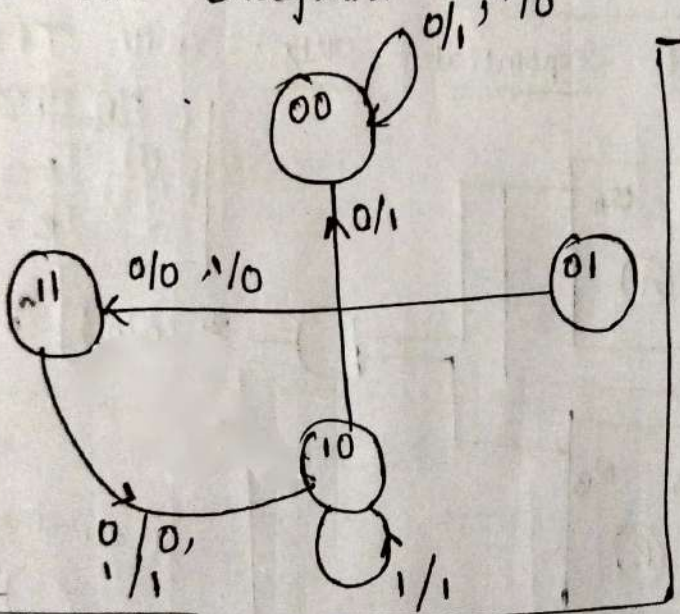
$Q_B^+ = D_B = \overline{Q_A} Q_B$

$= \overline{0} \cdot 0 = 0$

$y = \overline{0} \cdot \overline{0} + 0 \cdot 0 = 1$

$y = 1$

Step 3: State diagram



Step 1:- F/p eqn's

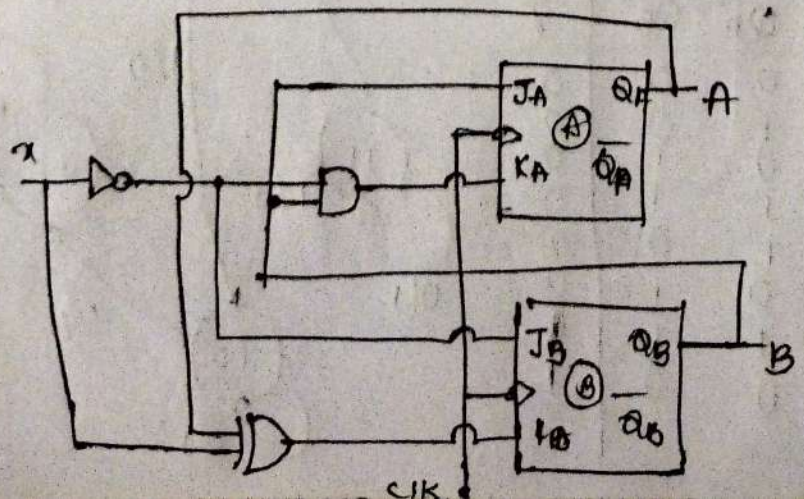
$J_A = Q_B$

$K_A = Q_B \cdot \overline{x}$

$J_B = \overline{x}$

$K_B = Q_A \oplus x$

Analysis of clocked sequential CKT (with JK F.F)



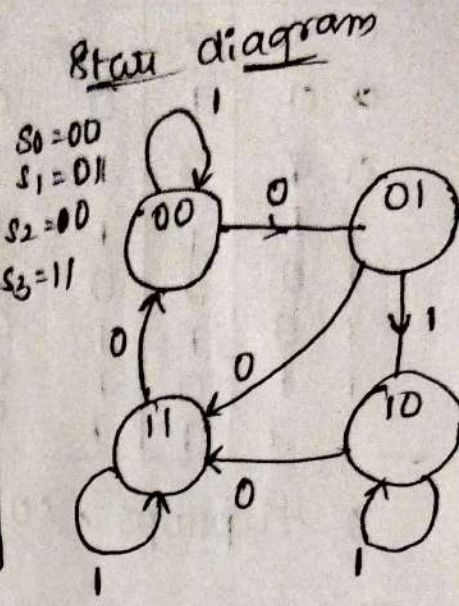
State table:-

PS Q_A	PS Q_B	x	J_A	K_A	J_B	K_B	NS Q_A^+	NS Q_B^+
0	0	0	0	0	1	0	0	1
0	0	1	0	0	0	1	0	0
0	1	0	1	1	1	0	1	0
0	1	1	1	1	0	1	1	0
1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	1	1	0
1	1	0	1	1	0	0	0	1
1	1	1	1	1	0	1	1	1

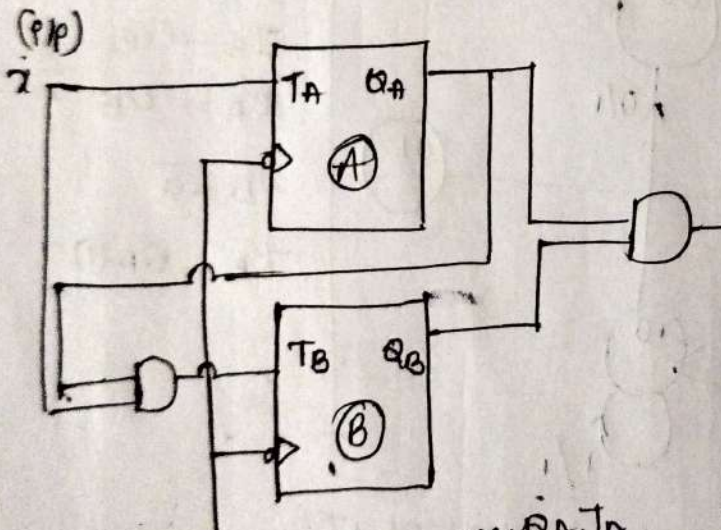
NS → Q_A^+ / J_A / K_A

→ Q_B^+ / J_B / K_B

State diagram



Analysis of clocked sequential ckts :- (with TFF)



i/p eqns

(i) $T_A = x$

$T_B = Q_A x$

$y = Q_A Q_B$

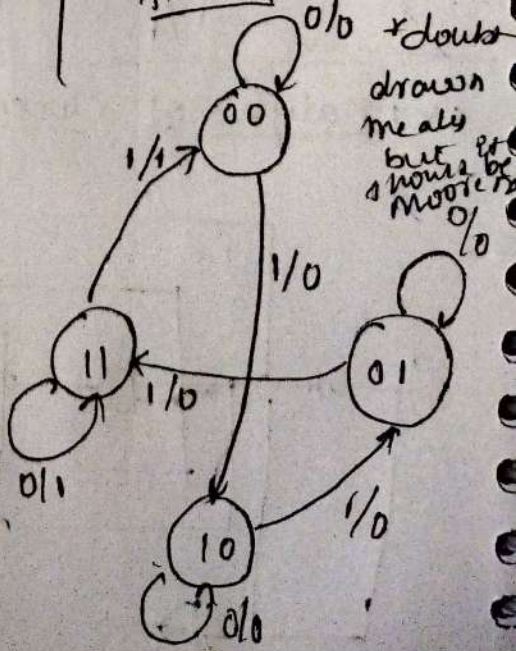
State table :-

PS Q_A	PS Q_B	x	T_A	T_B	NS Q_A^+	NS Q_B^+	y
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	1	0	0	0
0	1	1	0	1	0	0	0
1	0	0	0	0	1	0	0
1	0	1	0	0	1	0	0
1	1	0	1	1	0	1	1
1	1	1	1	1	0	0	1

→ Q_A^+, T_A

NS

State diagram



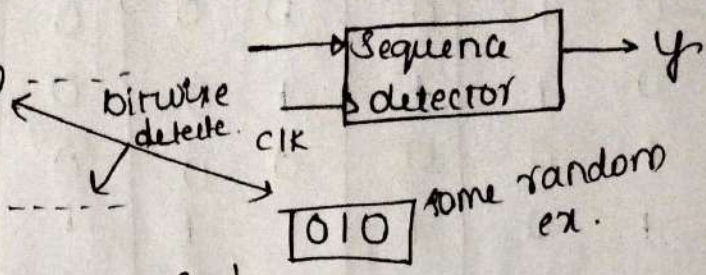
Pattern or Sequence Detector:-

→ The stream of bit has been as i/p, when the clk is high and a particular pattern/sequence is detected.

→ As soon as sequence is detected the o/p becomes high & then again becomes low.

eg:- x = 0 1 1 0 0 1 0 1 0 0 ---- (overlap)

y = 0 0 0 0 0 0 1 0 1 0 ----



step 1: state diagram: (Mealy Machine)

S_0 = reset (power up)

$S_1 = 0$

$S_2 = 01$

step 2: state assignment:

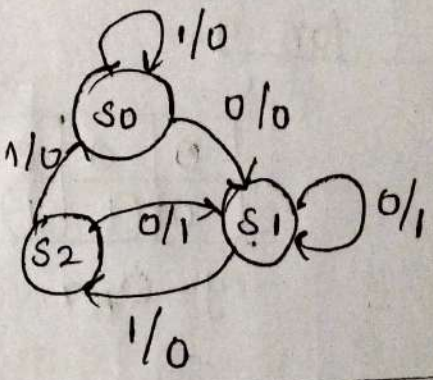
$S_0 = 00$

$S_1 = 01$

$S_2 = 00$

(i) Overlapping

(ii) NO Overlapping



Q) Design a sequence detector to detect 3 or more consecutive 1's in a string of bits coming through an i/p line.

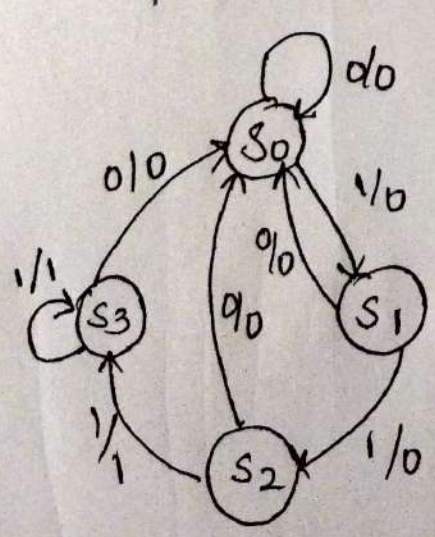
x = 0 0 1 1 1 0 1 1 1 0 ---

y = 0 0 0 0 1 0 0 0 1 1 0 ---

y = 0 0 0 0 1 0 0 0 1 0 0 ---

overlap

non overlap



$S_0 = 00$

$S_1 = 01$

$S_2 = 10$

$S_3 = 11$

	Has	Wrath
S_0 = reset	--	1 1 --
S_1 = 1	1	1 1 --
S_2 = 11	11	11 --
S_3 = 111	111	--

State table:-

PS Q_A	PS Q_B	x	Q_A^+	Q_B^+	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

NS = D

$D_A = Q_A^+$

$D_B = Q_B^+$

for Q_A^+ :- D_A

Q_A \ $Q_B x$	00	01	11	10
0	0	0	1	0
1	0	1	1	0

$Q_A^+ = D_A = Q_B x + Q_A x$

for Q_B^+ :-

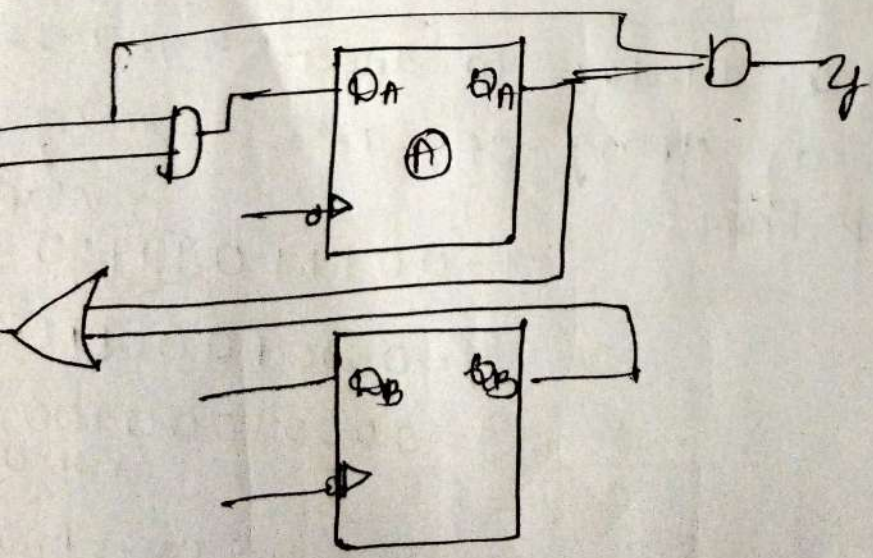
Q_A \ $Q_B x$				
0	0	1	0	0
0	1	1	1	0

$Q_B^+ = D_B = (\bar{Q}_B x + Q_A x)$

for y:-

0	0	0	0	
0	1	1	0	

$y = Q_A x$.



State reduction & assignment

State table

PS	NS x=0	NS x=1	O/P x=0	O/P x=1
a	a	b	0	0
b	c	d=f	0	0
c	a	d=f	0	0
d	e	f	0	1
e	a	f	0	1
g	g=e	f	0	1
f	a	f	0	1

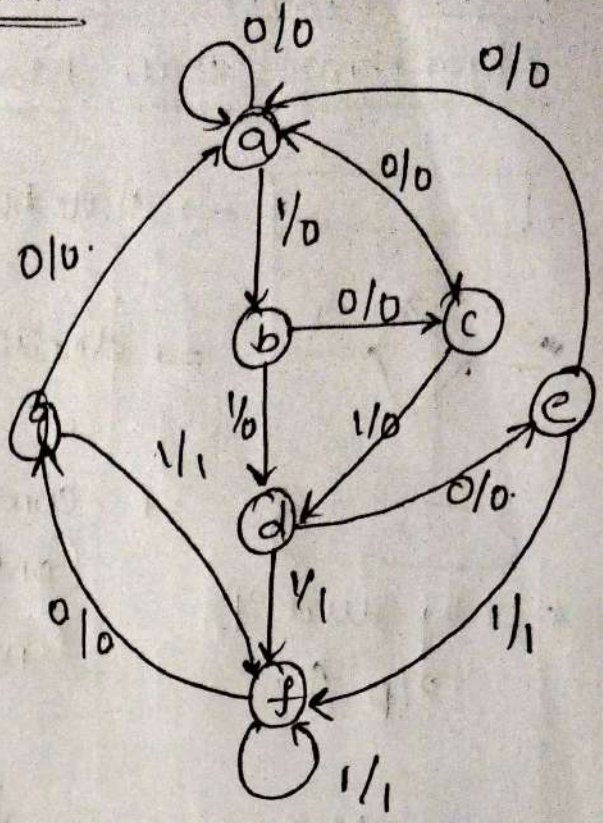
'g' eliminate c=g

* If NS & O/p of 2 present state are same then we can eliminate one state.

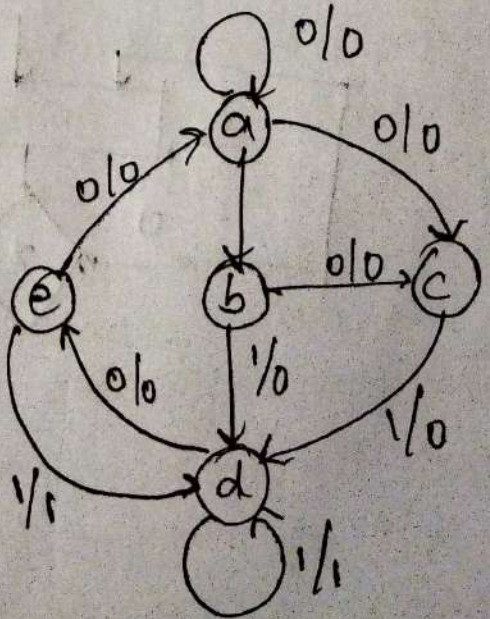
so after eliminating 'g' make 'g' as 'e'

d = f

now, no 2 O/p of PS & NS are same.

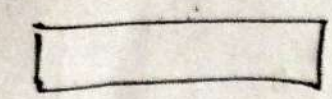


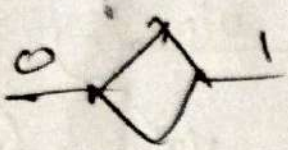
PS	NS x=0	NS x=1	O/P x=0	O/P x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1



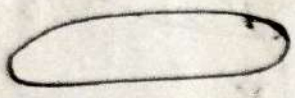
ASM chart:-

Algorithmic state Machine:

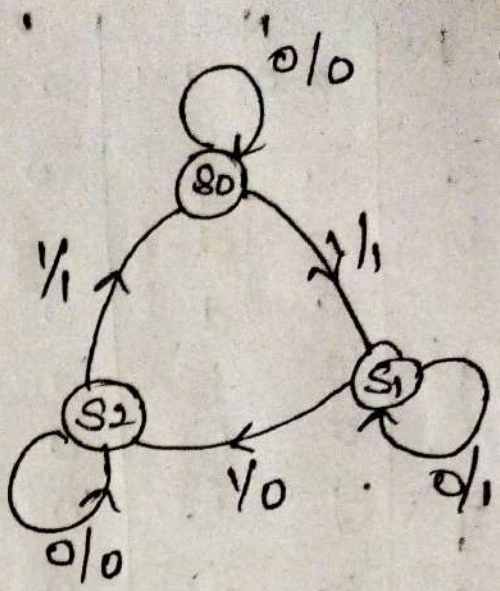

⇒ state box



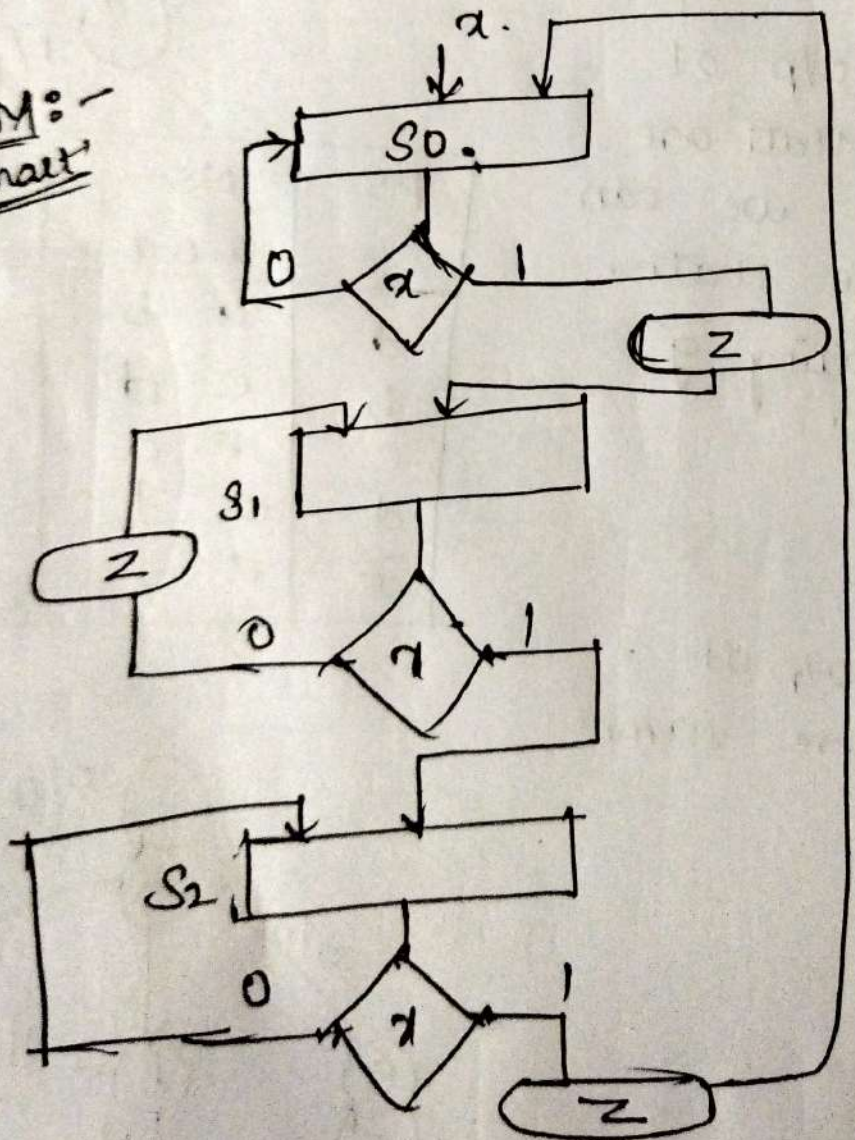
⇒ decision box


⇒ cond box (only used in ready box)

* only used if o/p ↑.

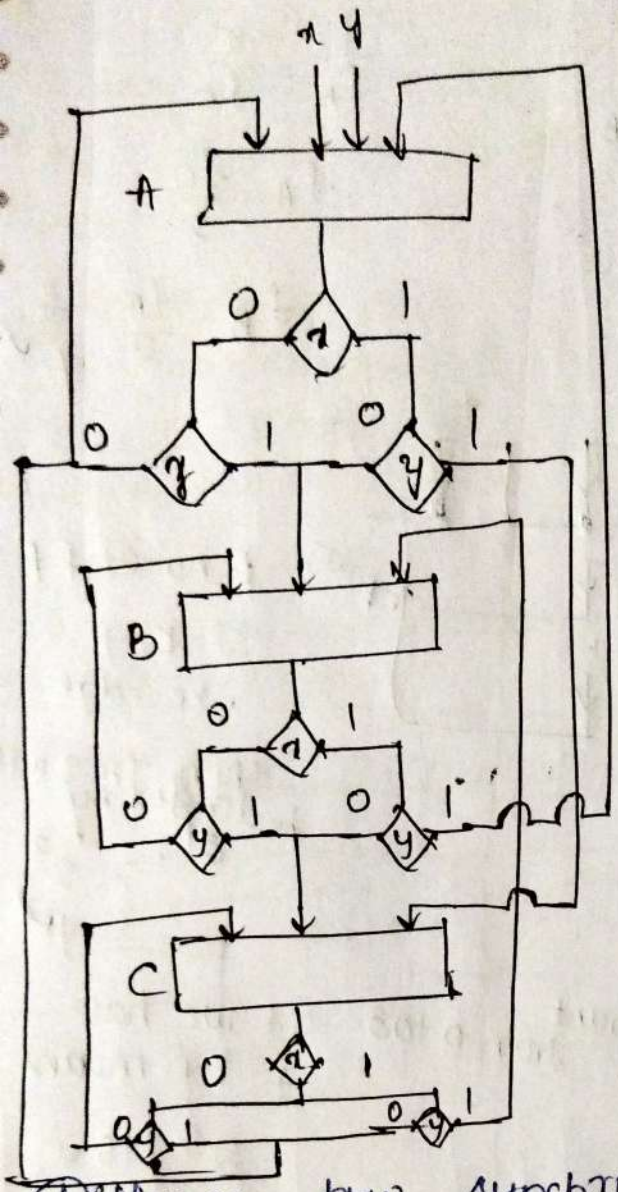


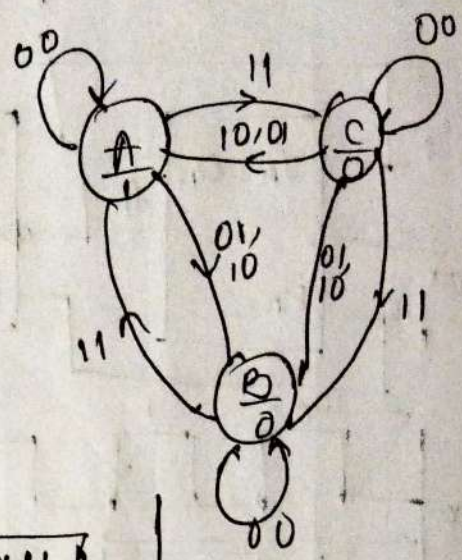
ASM:- chart



ASM Chart for Moore State Machine

o/p depends on p.s





x y
0 0
0 1
1 0
1 1

A → O/P = 1

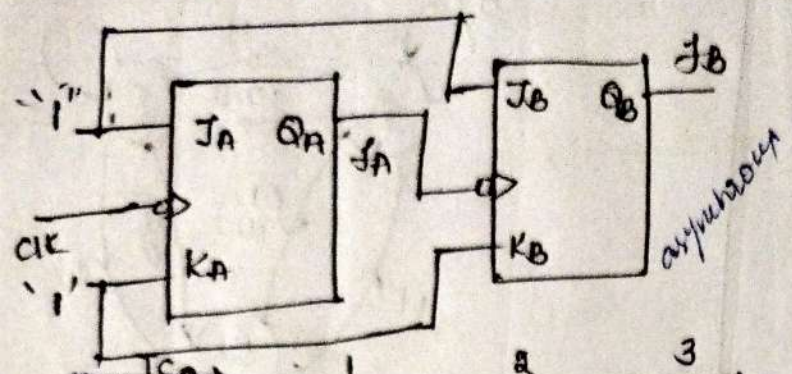
B = C = O/P = 0

Difference btw Synchronous & Asynchronous :-

Synchronous	Asynchronous
→ These ckts are easy to design & but they are slower	→ These ckts are diff to design but they are faster as clk is not present
→ A clocked FF acts as memory element [FF]	→ An unclocked FF or time delay element is used as memory element [latch]
→ The status of memory element is affected only at the active edge of clk, if i/p is change	→ The status of memory will change any time as soon as i/p is changed

Introduction to Counters:-

FF as divide by 2 ckt:-

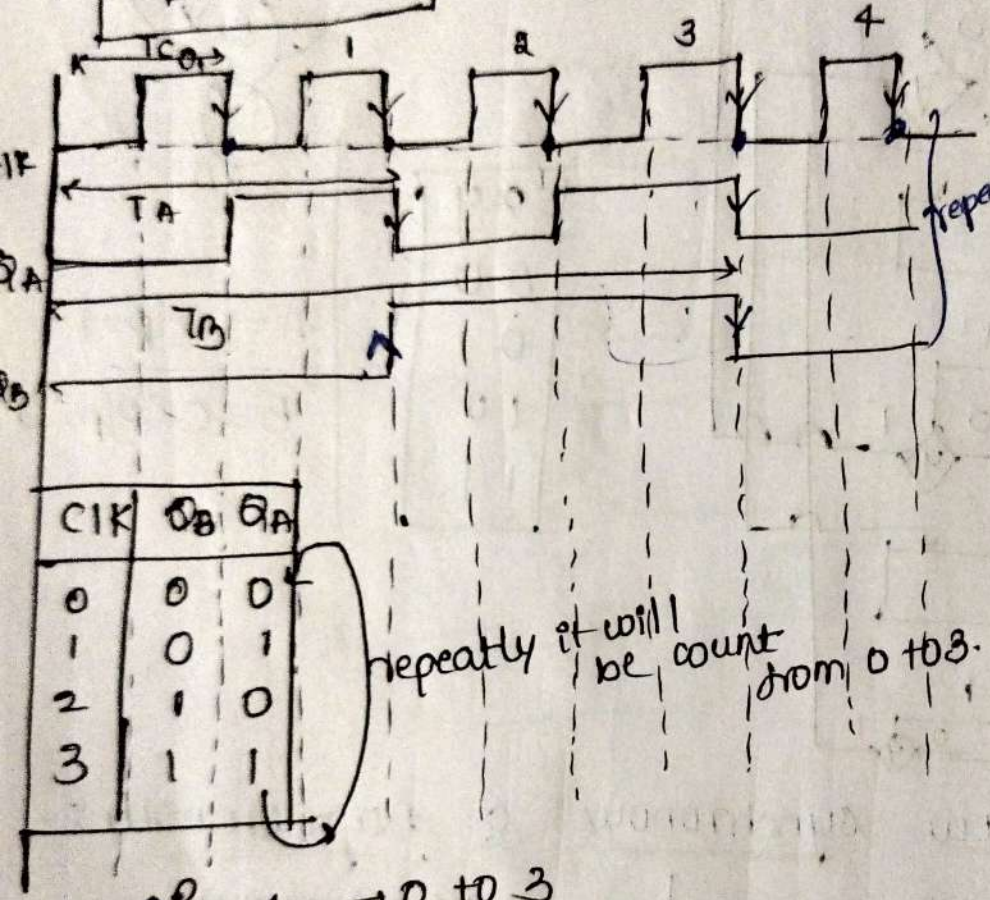


$T_A = 2T_C$

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

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

$f_B = \frac{f_A}{2} = \frac{f_C}{4}$ ✓



CLK	Q_B	Q_A
0	0	0
1	0	1
2	1	0
3	1	1

repeatly it will be count from 0 to 3.

$2^P = 4 \rightarrow 0$ to 3

$2^P = 16 \rightarrow 0$ to 15

P no. of FF

J = K = 1

-Ve edge

then freq will be divided by

$2^P = 2^2$

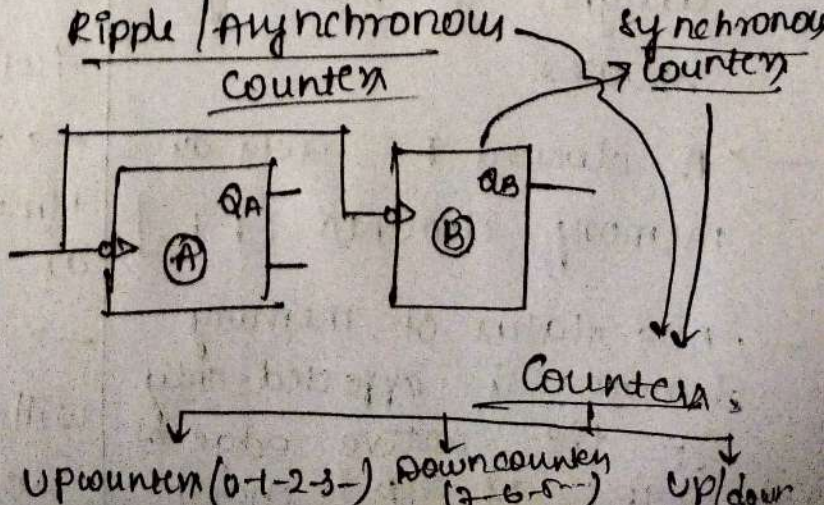
$= 4$ ✓

if we have 4 FF means

$2^4 = 16$ ✓

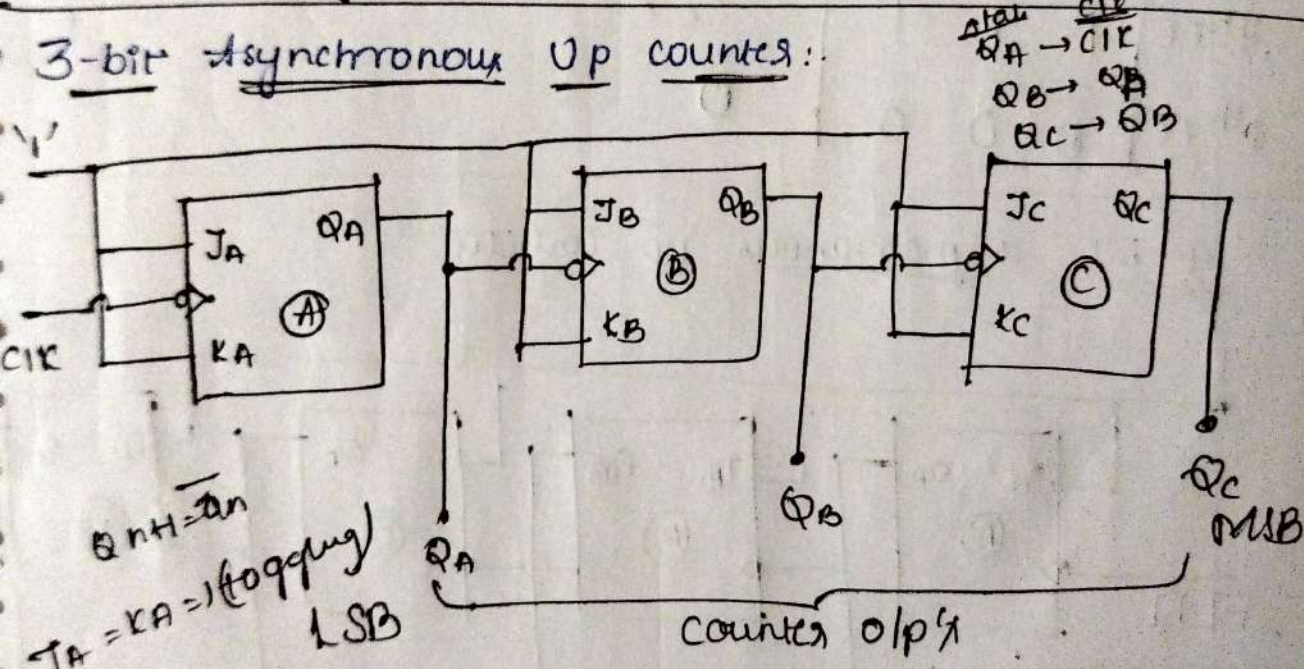
$f/16$ ✓

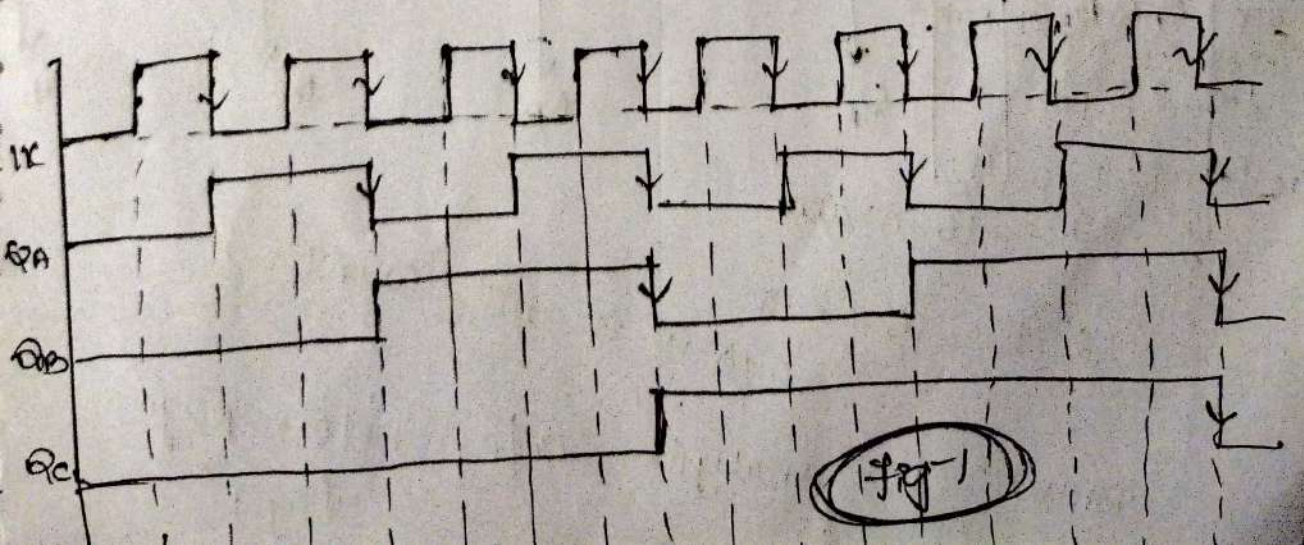
Types of counters:-



Asynchronous / ripple counter	Synchronous counter
1) FF are counted in such a way that the o/p of 1st FF drives the clk of next FF	1) There is no connection b/w o/p of 1st FF & clk of next FF
2) FF are not clked simultaneously & ckt is simple for more no. of states	2) FF are clocked simultaneously & ckt becomes complicated as no. of states ↑.
3) speed is slow as clk is propagated through no. of stages	3) speed is high as clk is given at same time

3-bit Asynchronous Up counter:





Clk	Q_C	Q_B	Q_A	Decimal eq
initially	0	0	0	0
1st (1) Falling edge	0	0	1	1
2nd FF (2)	0	1	0	2
3rd (3) FF	0	1	1	3
4th (4) FF	1	0	0	4
5th FF	1	0	1	5
6th FF	1	1	0	6
7th FF	1	1	1	7
8th FF	0	0	0	0

0 to 7

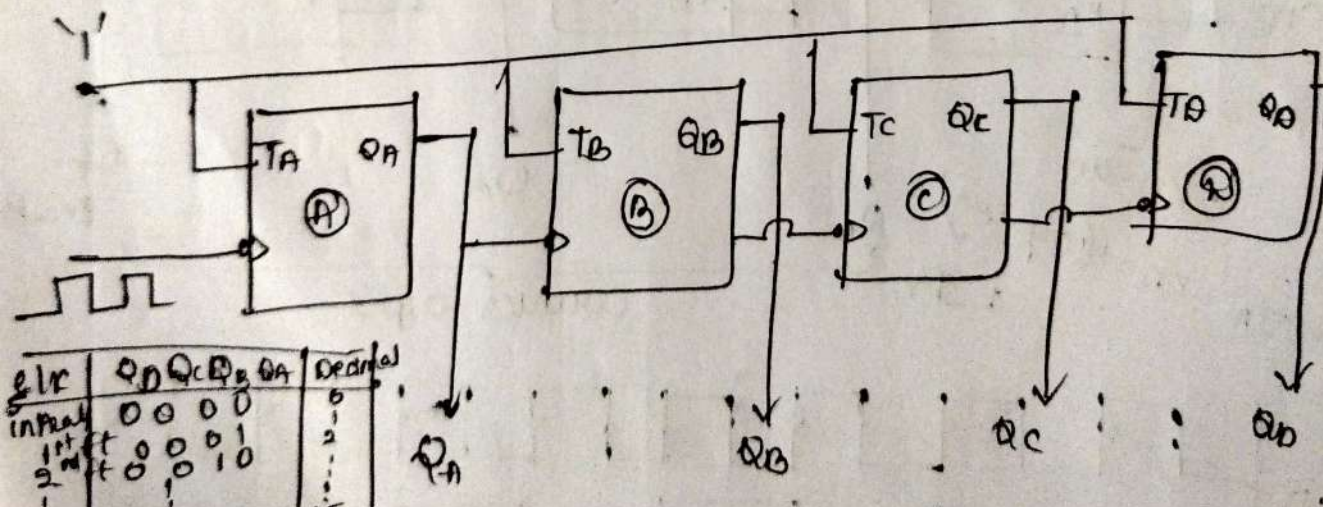
$2^n = 2^3 = 8$ ← no. of FF

8 states

maximum count

$2^n - 1 = 8 - 1 = 7$

4-bit asynchronous up counter



Clk	Q_D	Q_C	Q_B	Q_A	Decimal
initially	0	0	0	0	0
1st FF	0	0	0	1	1
2nd FF	0	0	1	0	2
...	...	...	...	...	...
15th FF	1	1	1	1	15

$2^n = 2^4 = 16$ states

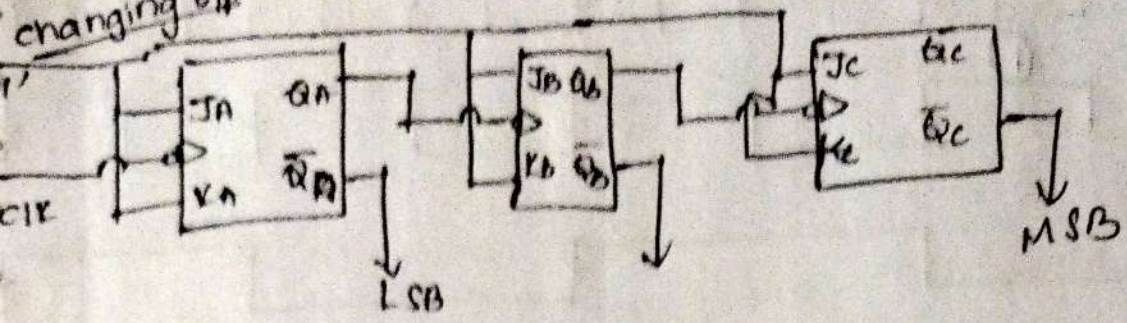
$2^n - 1 = 16 - 1 = 15$

0 to 15 FF will count.

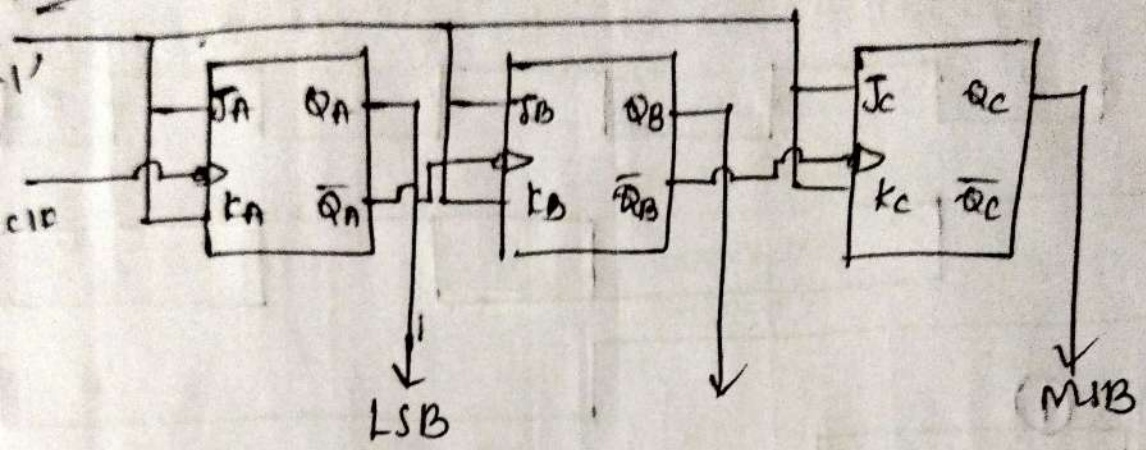
Draw clk analysis a/c before page 'fig-1'

3-bit & 4-bit Asynchronous Ripple counter:-

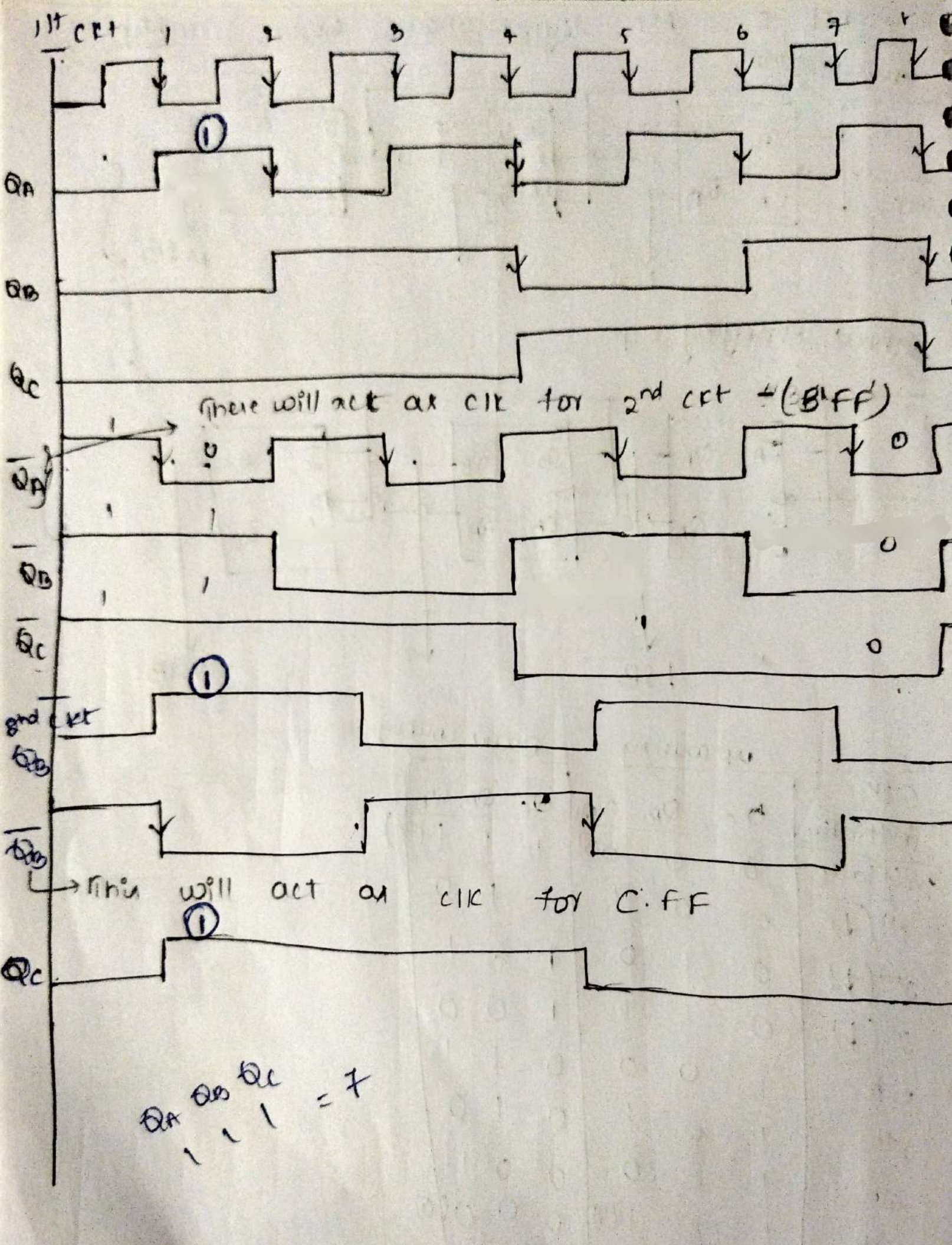
changing o/p



without changing o/p



	up counting			down counting		
CLK	Q_C	Q_B	Q_A	$\overline{Q_C}$	$\overline{Q_B}$	$\overline{Q_A}$
~~initially~~						
initially	0	0	0 (0)	1	1	1 (7)
1st (↓)	0	0	1	1	1	0
2nd (↓)	0	1	0	1	0	1
3rd (↓)	0	1	1	1	0	0
4th	1	0	0	0	1	1
5th	1	0	1	0	1	0
6th	1	1	0	0	0	1
7th	1	1	1 (7)	0	0	0 (0)
8th	0	0	0	1	1	1 (7)

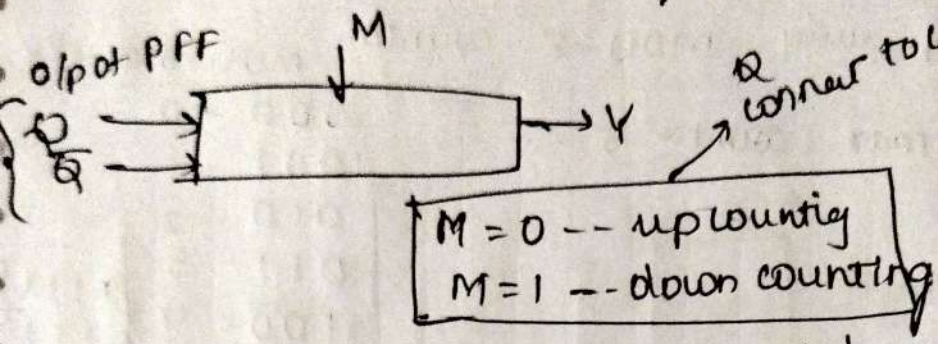

1st CLK
1
2
3
4
5
6
7
Q_A
Q_B
Q_C
There will act as clk for 2nd ckt (B FF)
$\overline{Q_A}$
$\overline{Q_B}$
$\overline{Q_C}$
0
2nd ckt
$\overline{Q_B}$
$\overline{Q_B}$
This will act as clk for C.FF
Q_C
Q_A Q_B Q_C = 7
1 1 1

3-bit & 4-bit Up/Down ripple counter :-

→ We have designed the up counters & down counters separately but in practice both these modes are combined

→ A mode control i/p (M) is used to select either up or down mode.

→ A comb ckt is req btw each pair of FF.



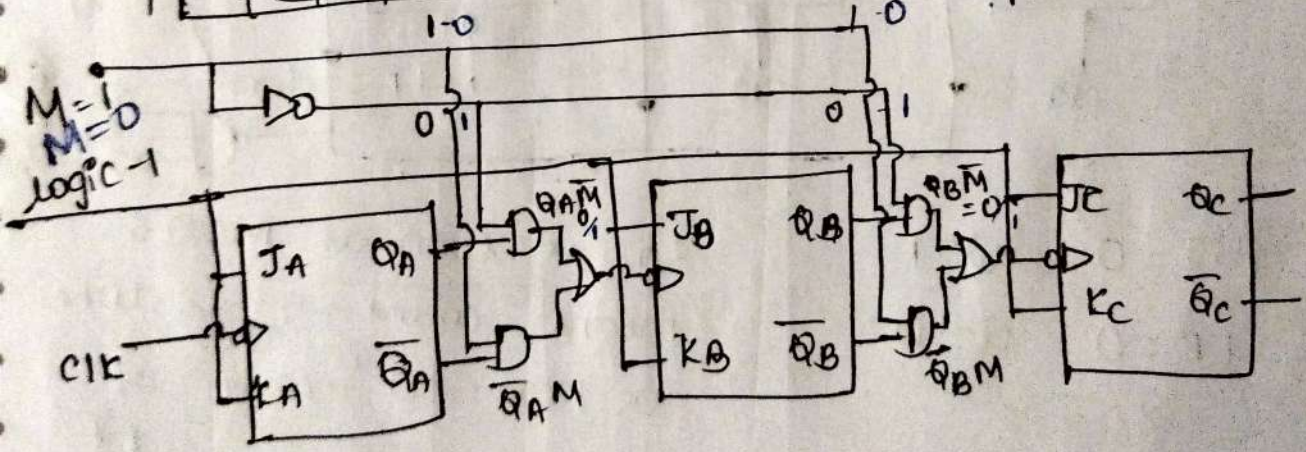
M	b	$\bar{Q}$	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Y = $\bar{Q}$

M \ $Q\bar{Q}$	00	01	11	10
0	0	0	1	1
1	0	1	1	0

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

M = 1 → down counter



Modulus of the Counter & Counting up to particular Value:-

→ 2-bit up/down Counter is called MOD-4 or modulus 4 counter

→ 3-bit counter is called MOD-8 counter.

n → no. of bits

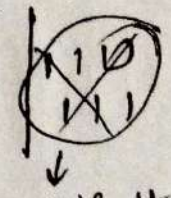
MOD number = 2^n

Eg: MOD-6 counter using MOD-8 counter.

State = 6 max Count - 5

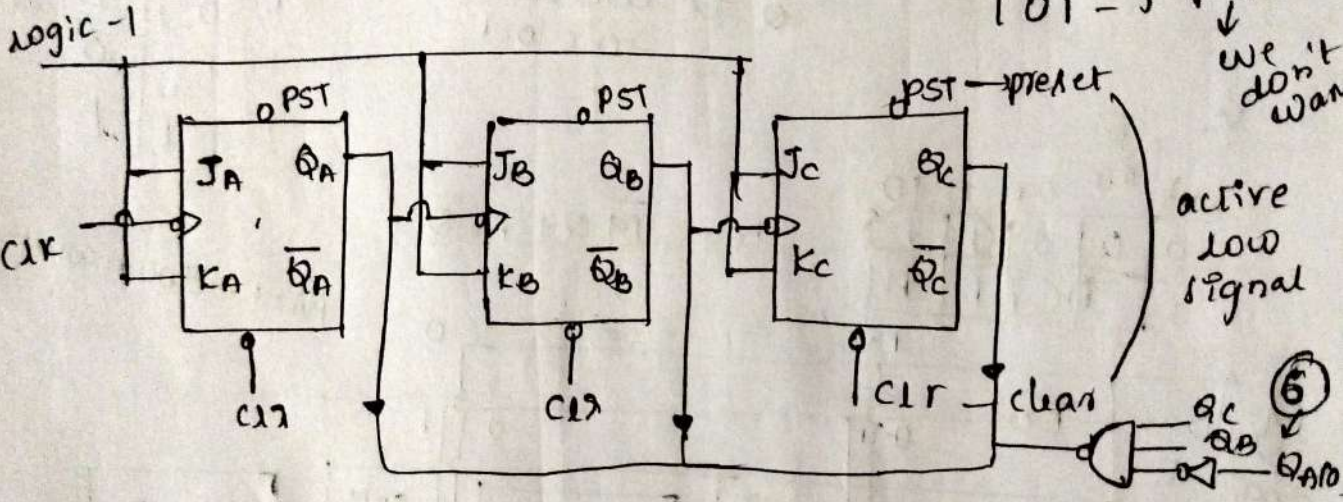
MOD-6 counter

000 - 0
001 - 1
010 - 2
011 - 3
100 - 4
101 - 5



we don't want

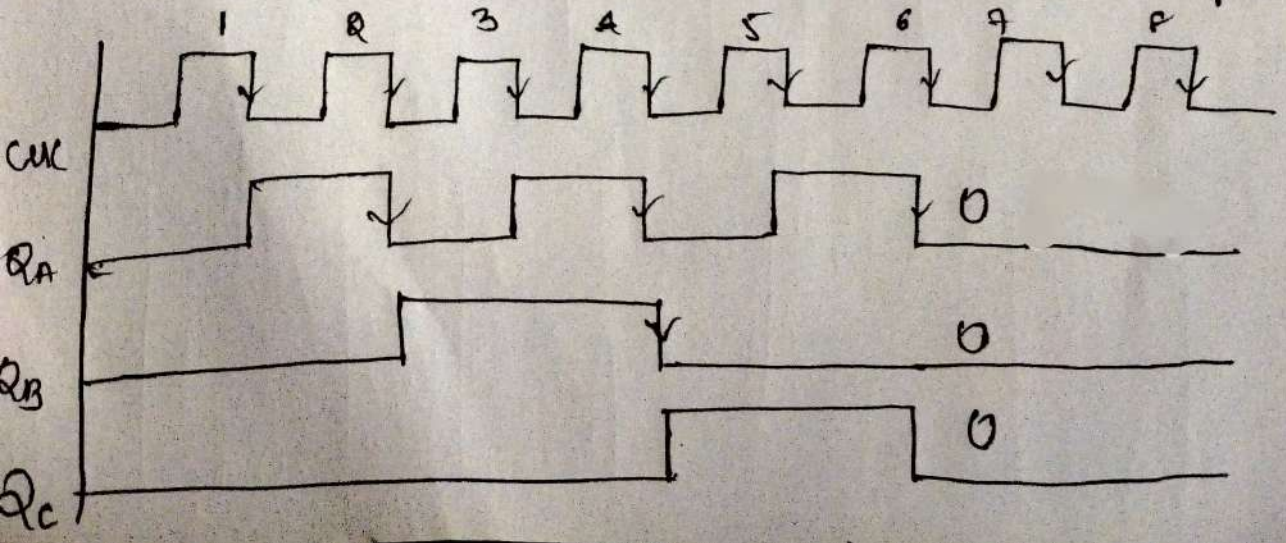
MOD counter



PST = 0, Q = 1

CLR = 0, Q = 0

So as soon as MOD 6 means, count-5 is done we have to reset of FF



State diagram of a Counter:-

n = no. of bits

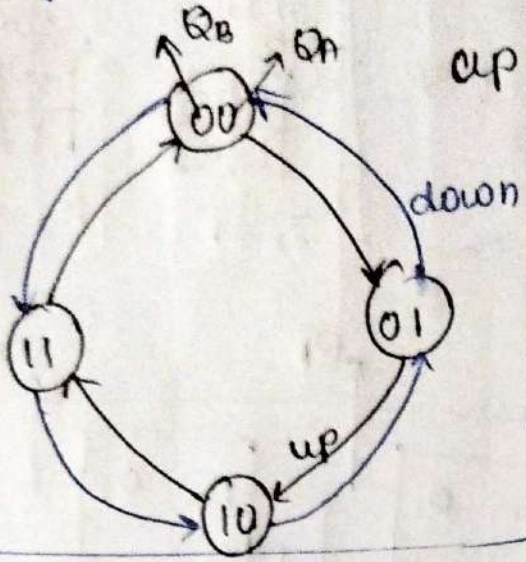
Max count = $2^n - 1$

2-bit up counter

$2^n = 2^2 = 4$ states

Q_B	Q_A
0	0
0	1
1	0
1	1

up ↓ ↑ down



Decade BCD Ripple counter:-

Imp points:-

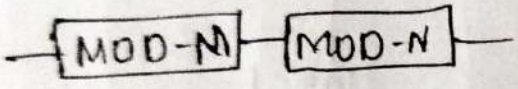
(i) -ve edge trigger : Q is clock → UC

+ve edge trigger : $\bar{Q}$ is clock → UC

-ve edge trigger : $\bar{Q}$ is clock → DC

+ve edge trigger : Q is clk → DC

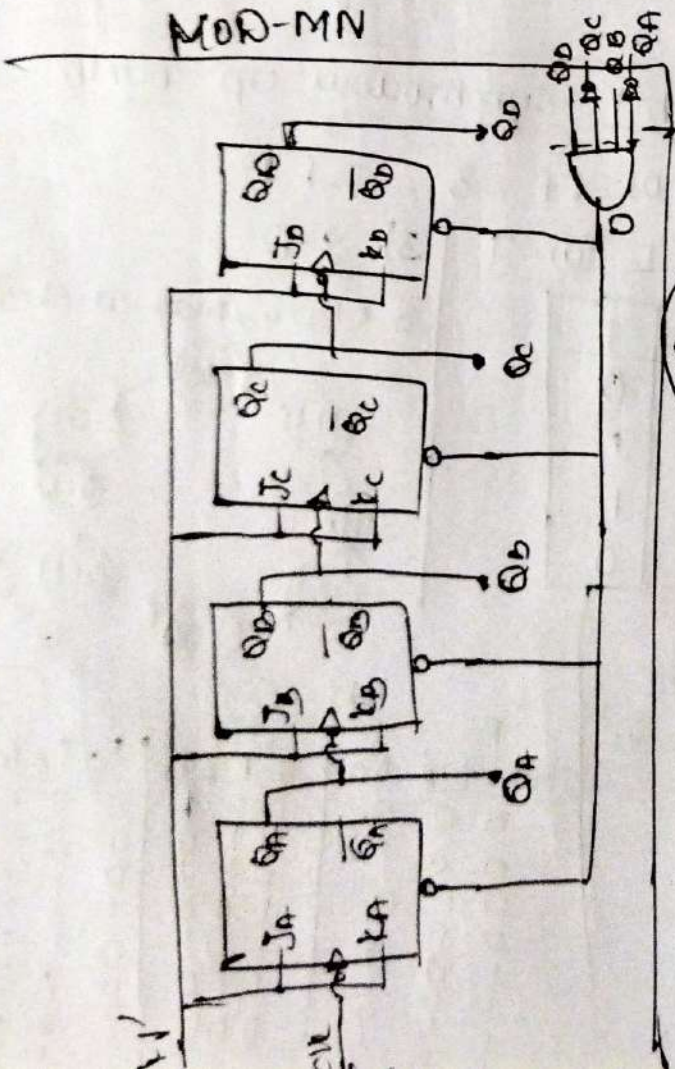
(ii) Cascade of counters

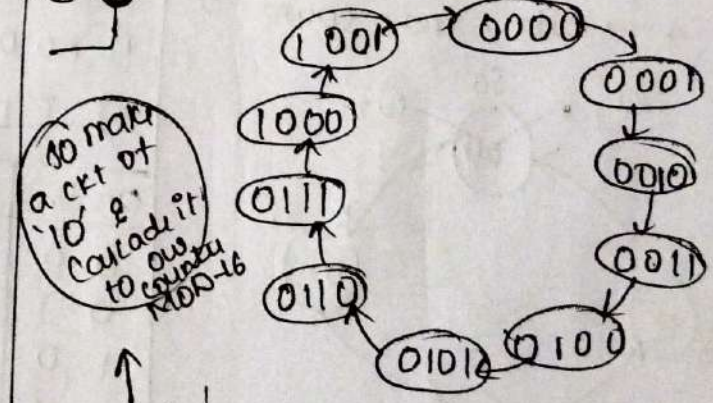


MOD-MN

no of state = 10

Max count = 9





at '10' stop it we can use MOD-16

clk	Q_D	Q_C	Q_B	Q_A
'Nth'	0	0	0	0
1	0	0	0	1
2	0	0	1	0
⋮	⋮	⋮	⋮	⋮
9	1	0	0	1
10	1	1	1	1

Synchronous Counter :-

1) Decide no. of FF
2) Excitation table of FF
3) State diagram for CKT
4) Obtain simplified eqns using K map
5) Draw the logic diagram

for J_1

Q_1 \ Q_2	0	1
0	0	1
1	X	X

$J_1 = Q_2$

for K_1

Q_1 \ Q_2	0	1
0	X	X
1	0	1

$K_1 = Q_2$

for J_2

Q_1 \ Q_2	0	1
0	1	X
1	1	X

$J_2 = 1$

for K_2

Q_1 \ Q_2	0	1
0	X	1
1	X	1

$K_2 = 1$

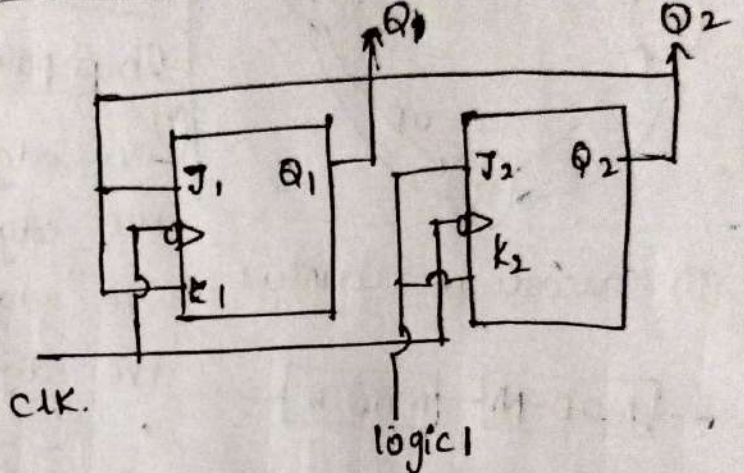
• 8-bit Synchronous up Counter :-

JK / D / T

(i) No. of FF = 2

(ii) ET of JK FF

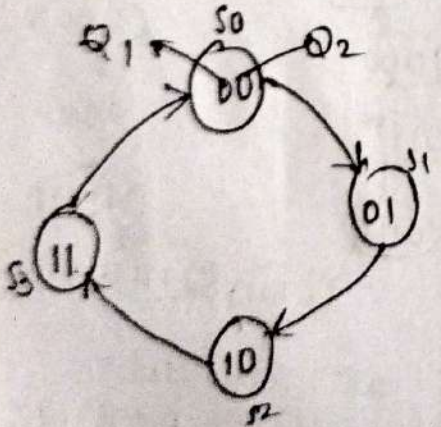
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0



(iii) State diagram

$2^n = 2^2 = 4$

4 states, max count - 3



• 3-bit Synchronous up counter

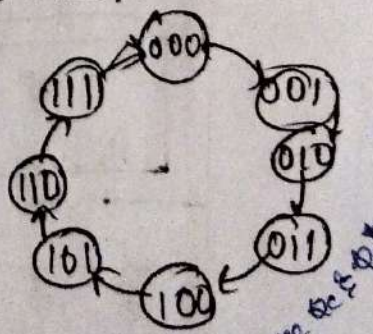
1) no. of ff = 3, T-FF

2) ET for T

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

3) 3-bit

8 states, Max count - 7

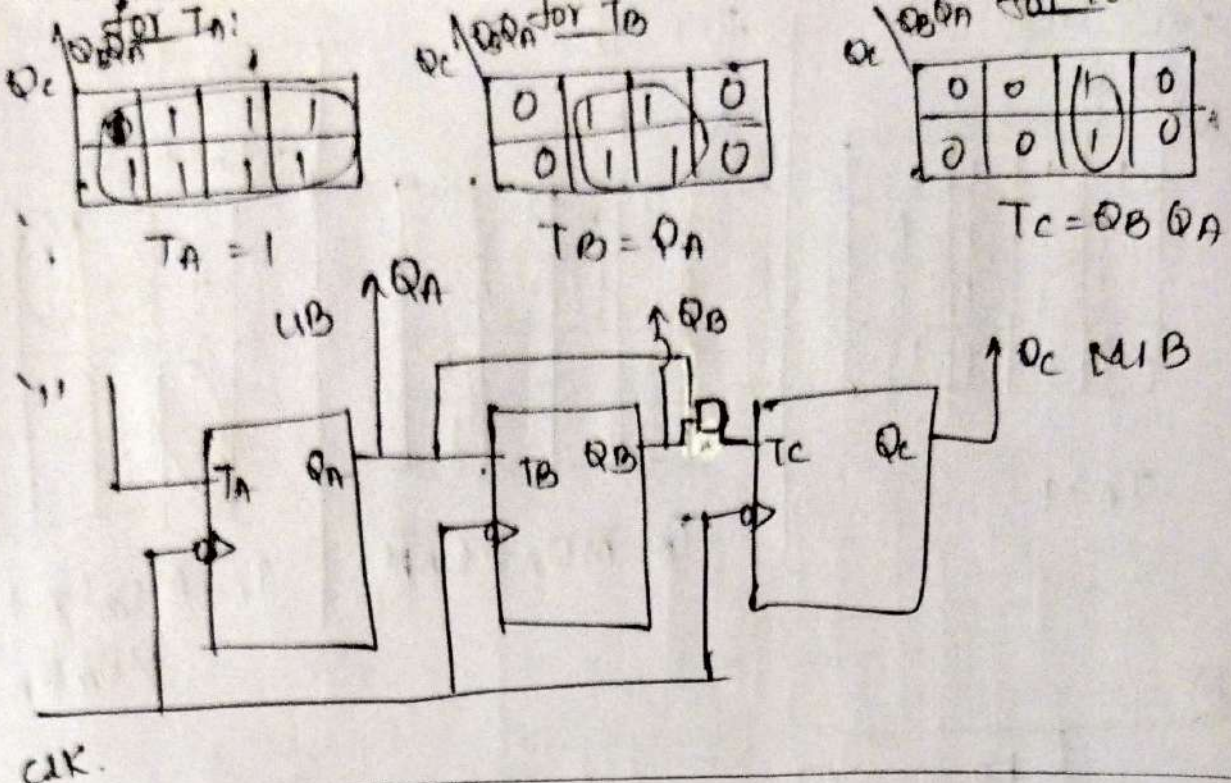


CKT excitation table → $Q_1 Q_1^+$

$Q_1 Q_2$	$Q_1^+ Q_2^+$	$J_1 K_1$	$J_2 K_2$
0 0	0 1	0 X	1 X
0 1	1 0	1 X	X 1
1 0	1 1	X 0	1 X
1 1	0 0	X 1	X 1

CKT E.T

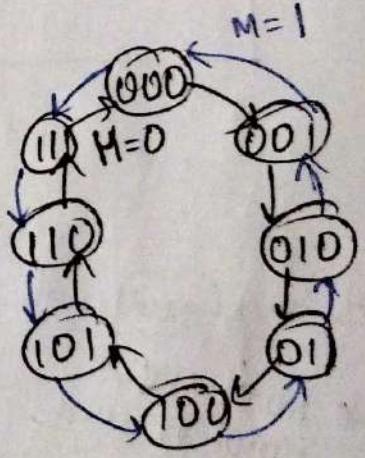
PS Q_C	Q_B	Q_A	NS Q_C^+	Q_B^+	Q_A^+	T_C	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



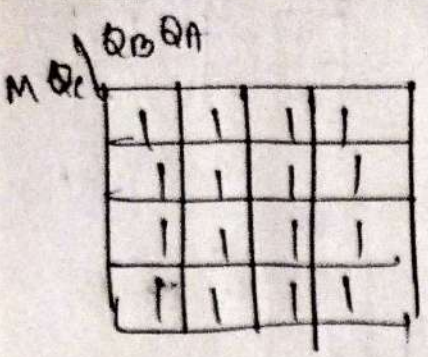
3-bit Up down Counter Synchronous

M = 0 → Up counter
M = 1 → down counter

M	PS Q_C	Q_B	Q_A	NS Q_C^+	Q_B^+	Q_A^+	i/p of FF T_C	T_B	T_A	
0	0	0	0	0	0	1	0	0	1	M=0
0	0	0	1	0	1	0	0	1	1	
0	0	1	0	0	1	1	0	0	1	
0	0	1	1	1	0	0	1	1	1	
0	1	0	0	1	0	1	0	0	1	
0	1	0	1	1	1	0	0	1	1	
0	1	1	0	1	1	1	0	0	1	
0	1	1	1	0	0	0	1	1	1	
1	0	0	0	1	1	1	0	0	1	M=1
1	0	0	1	0	0	0	0	1	1	
1	0	1	0	0	0	1	0	0	1	
1	0	1	1	0	1	0	0	1	1	
1	1	0	0	0	1	1	0	0	1	
1	1	0	1	1	0	0	1	1	1	
1	1	1	0	1	0	1	0	0	1	
1	1	1	1	1	1	0	0	1	1	

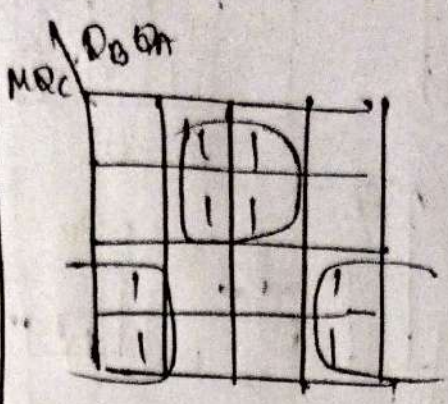


for T_A:



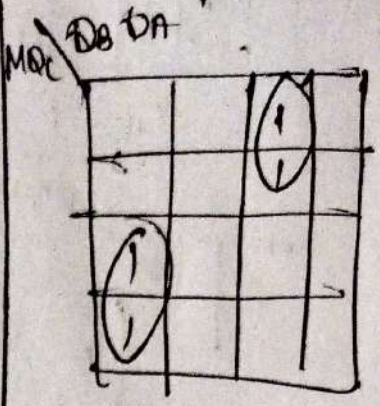
$T_A = 1$

for T_B:-

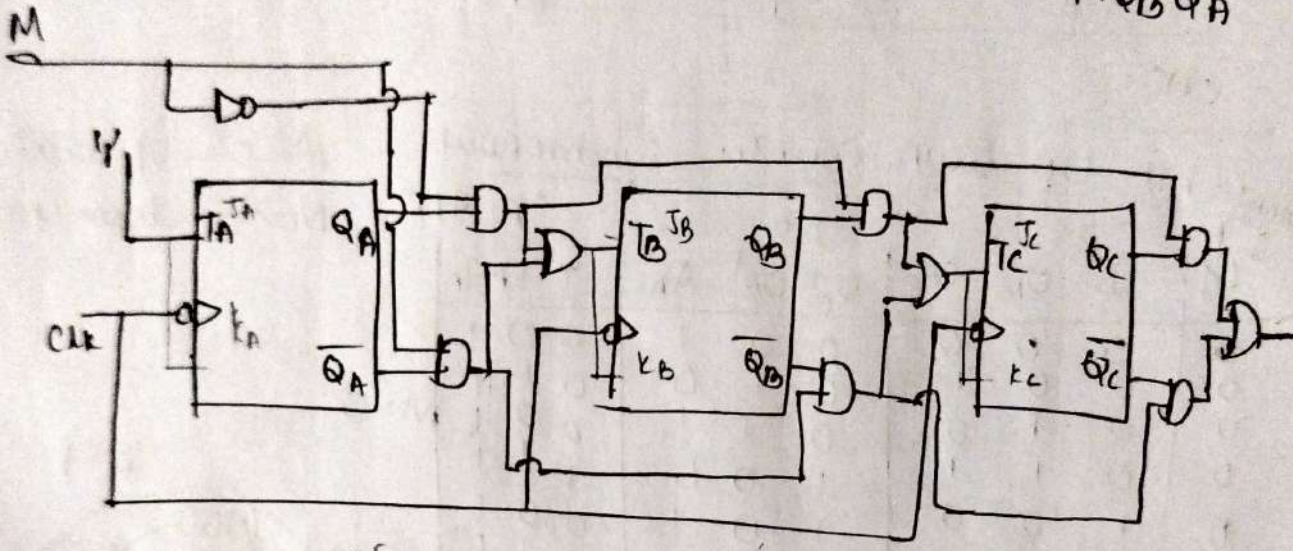


$T_B = \bar{M} Q_A + Q_A \bar{M}$

For T_C:-



$T_C = \bar{M} Q_B Q_A + M \bar{Q}_B \bar{Q}_A$



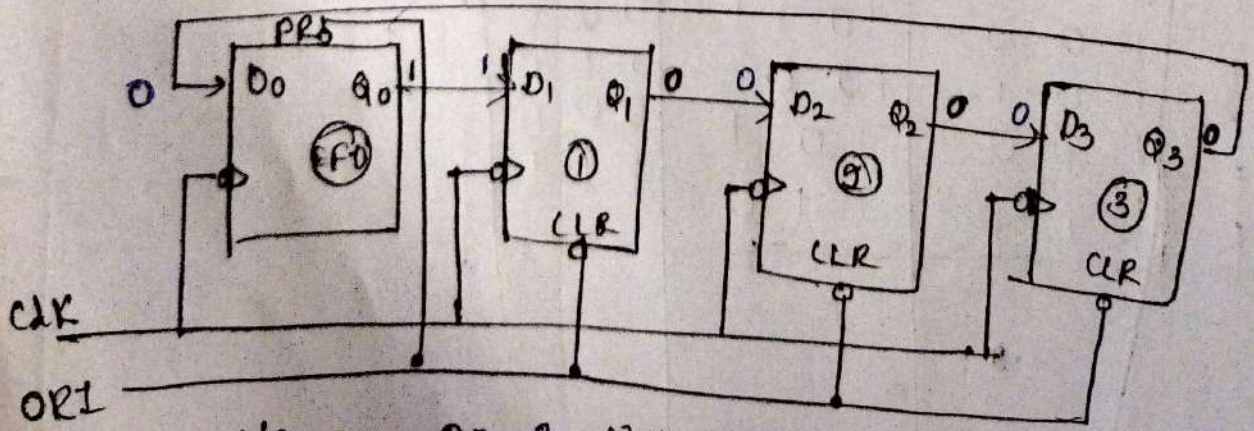
If we need four JK FF then see pencil line

Ring Counter:—

→ ring counter is a typical application of shift register

→ The only change is the o/p of last FF is connected to i/p of 1st FF

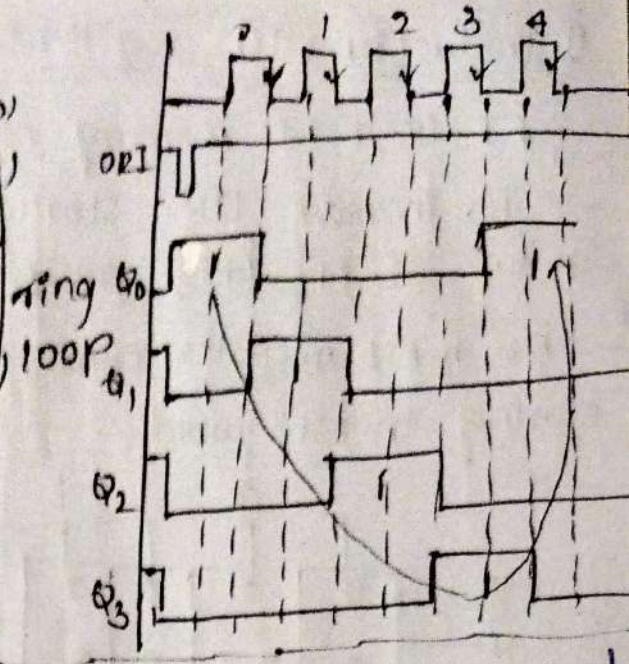
* NO. of states = NO. of FF used



Overriding i/p $PR = 0 \Rightarrow Q = 1$, $CLR = 0 \Rightarrow Q = 0$

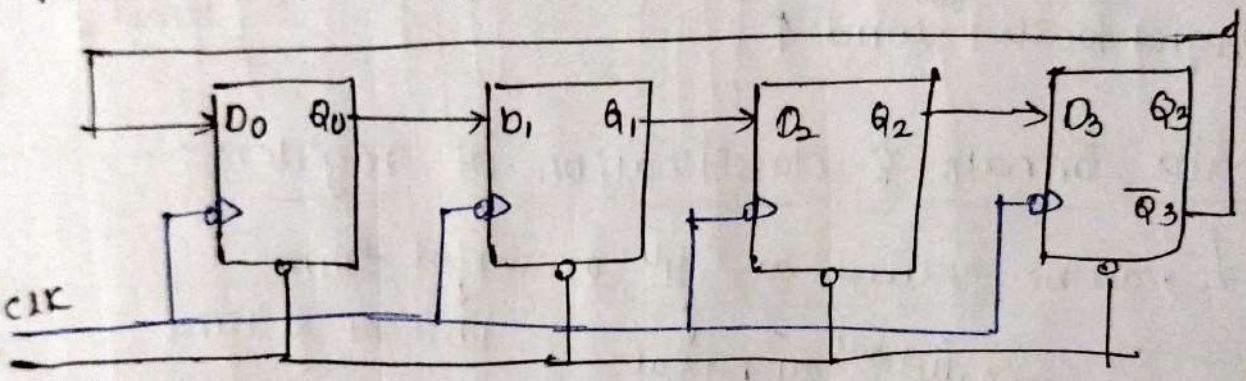
ORI	CLK	Q_0	Q_1	Q_2	Q_3	
⊔	x	1	0	0	0	(0)
1	↓	0	1	0	0	(1)
1	↓	0	0	1	0	(2)
1	↓	0	0	0	1	(3)
1	↓	1	0	0	0	(4)

ring loop



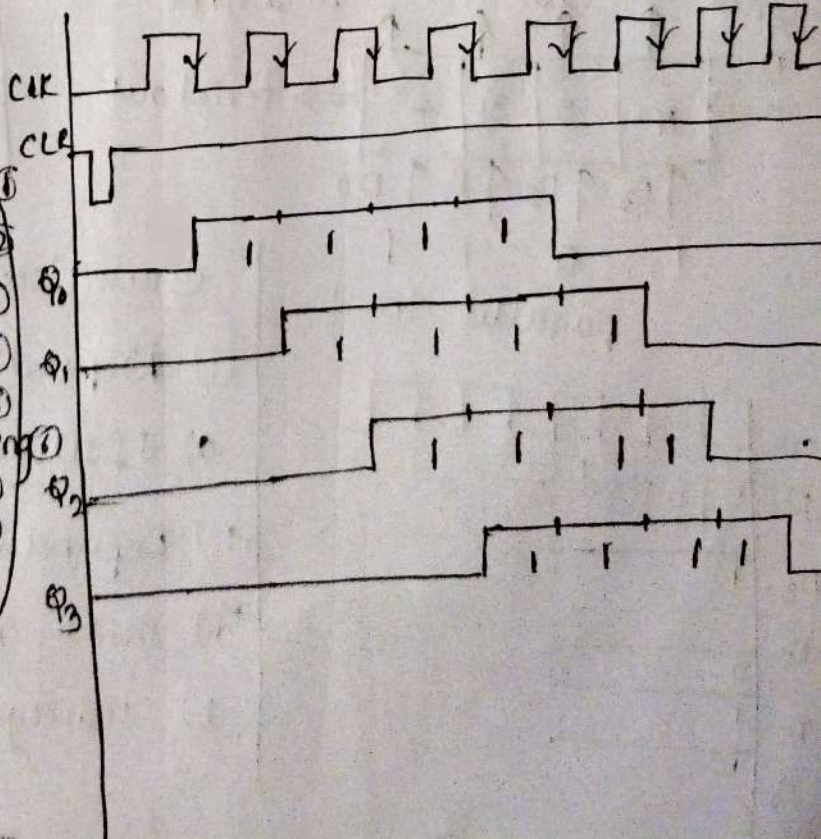
Johnson's Counter (Twisted / Switch Tail ring Counter)

No. of state = 2 × no. of FF



CLR	CLK	Q_0	Q_1	Q_2	Q_3	
⊔	x	0	0	0	0	①
1	↓	1	0	0	0	②
1	↓	1	1	0	0	③
1	↓	1	1	1	0	④
1	↓	1	1	1	1	⑤
1	↓	0	1	1	1	⑥
1	↓	0	0	1	1	⑦
1	↓	0	0	0	1	⑧
1	↓	0	0	0	0	—

ring

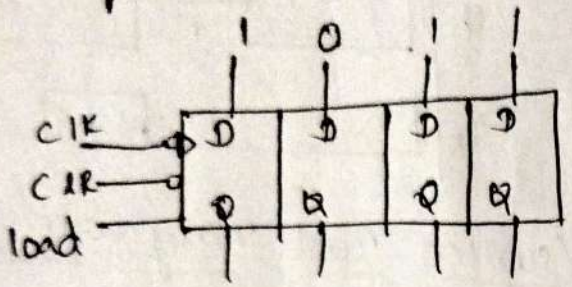


Introduction to registers:-

→ FF is '1' bit memory cell

→ To increase the storage capacity, we have to use group of FF. This group of ff is known as "REGISTER"

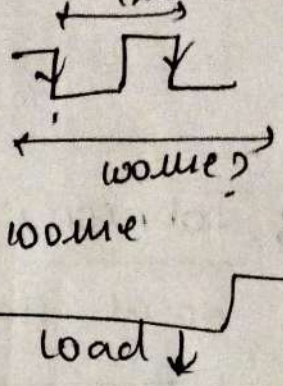
→ The n-bit register consist of "n" no. of FF & capable of storing "n-bit" word.



Synchronous: clock & load ↑

Asynchronous: only load ↑

we are bound to follow the clock

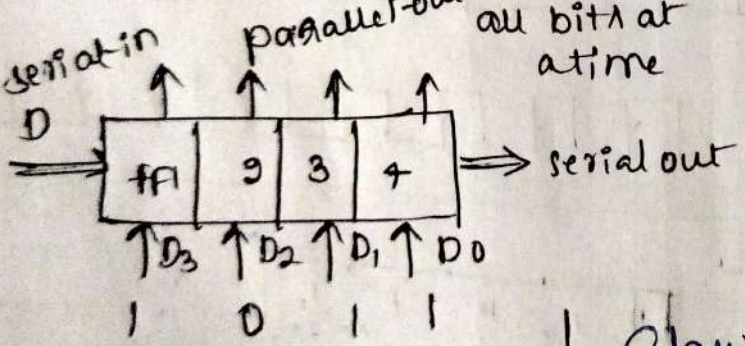


Data formats & classification of registers:-

↳ can be entered as μ or serial form

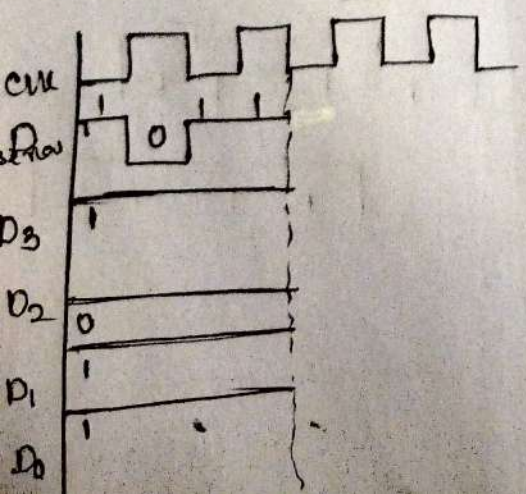
μ → all bits at a time

serial → 1-bit at a time.



Serial form - Temporal code

μ form - Special code



Classification:

(i) Depending on i/p & o/p

a) SISO b) SIPO c) PISO d) PIPO

(ii) Depending on applications

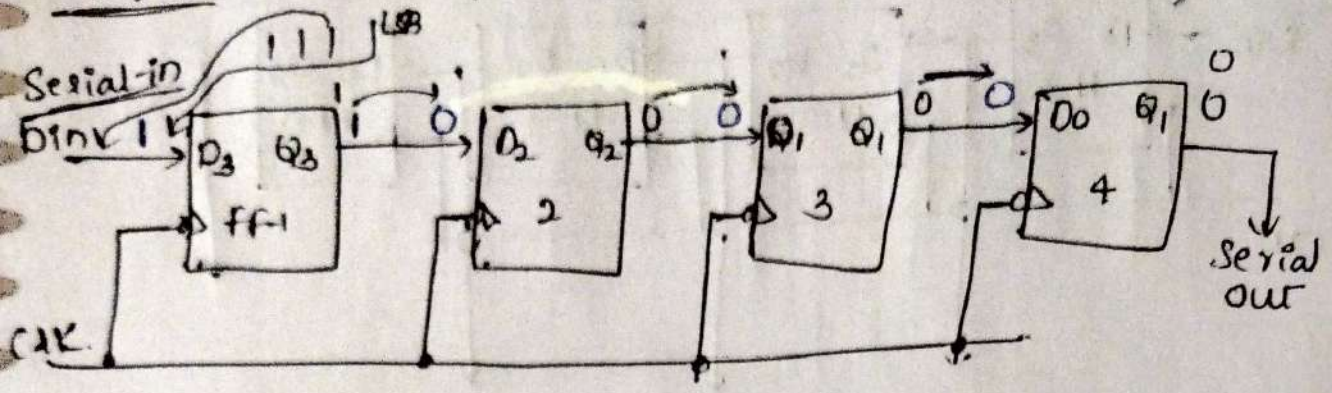
*a) Shift reg ← Bidirectional universal

b) storage reg

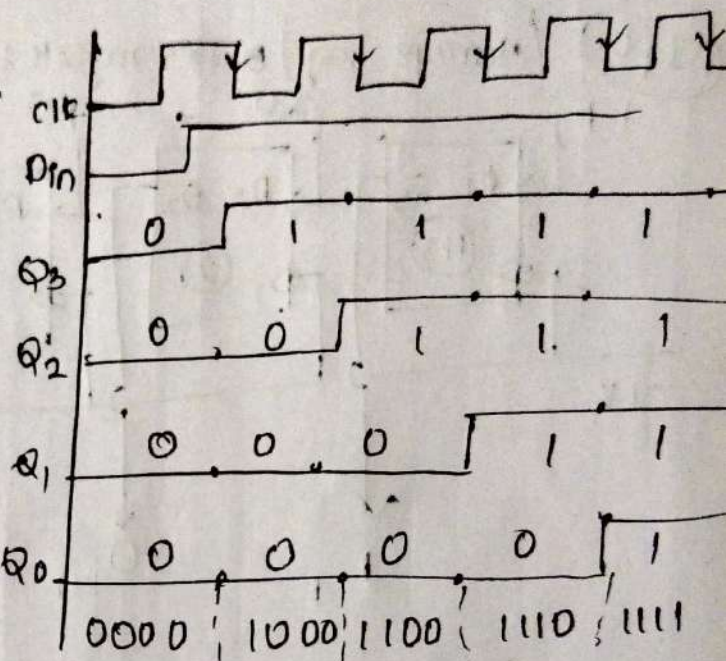
Shift Registers :- Data will shift see table

MSB LSB
serial-in = 1 1 1 1

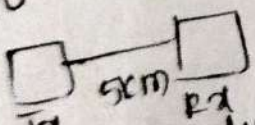
SISO - serial i/p - serial o/p



CLK	Q_3	Q_2	Q_1	Q_0
initially	0	0	0	0
↓	1	0	0	0
↓	1	1	0	0
↓	1	1	1	0
↓	1	1	1	1



for long distance SISO is better



one conductors

for PIPO it need 4 conductors

cost ↑

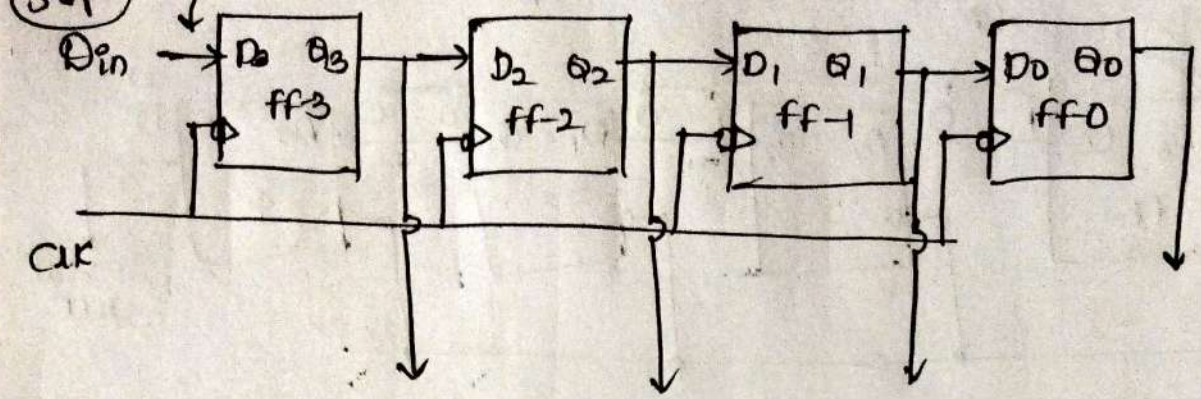
req more CLK pulse

SIPO & PIPO :-

'1111'

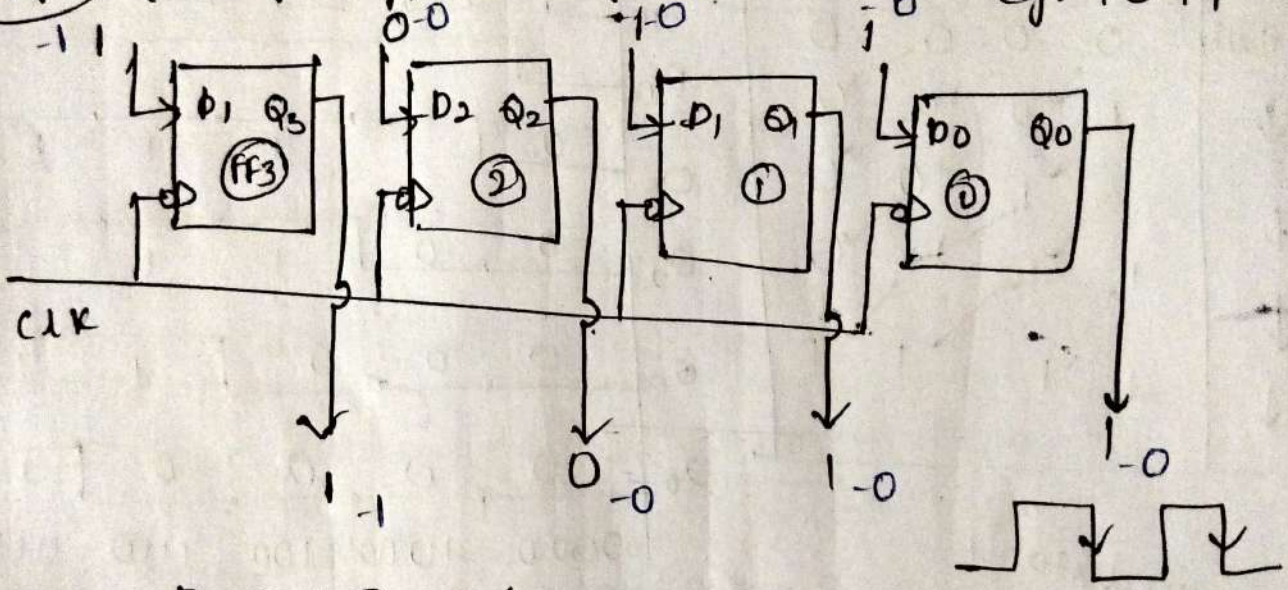
after 4th clk pulse out will parallel out

SIPO



PIPO - storage reg / buffer register.

eg: 1011



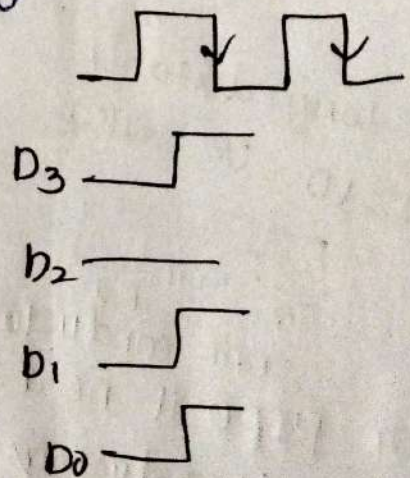
D = 0 ⇒ Qn+1 = 0

D = 1 ⇒ Qn+1 = 1

clk = 0 ⇒ Qn+1 = Qn

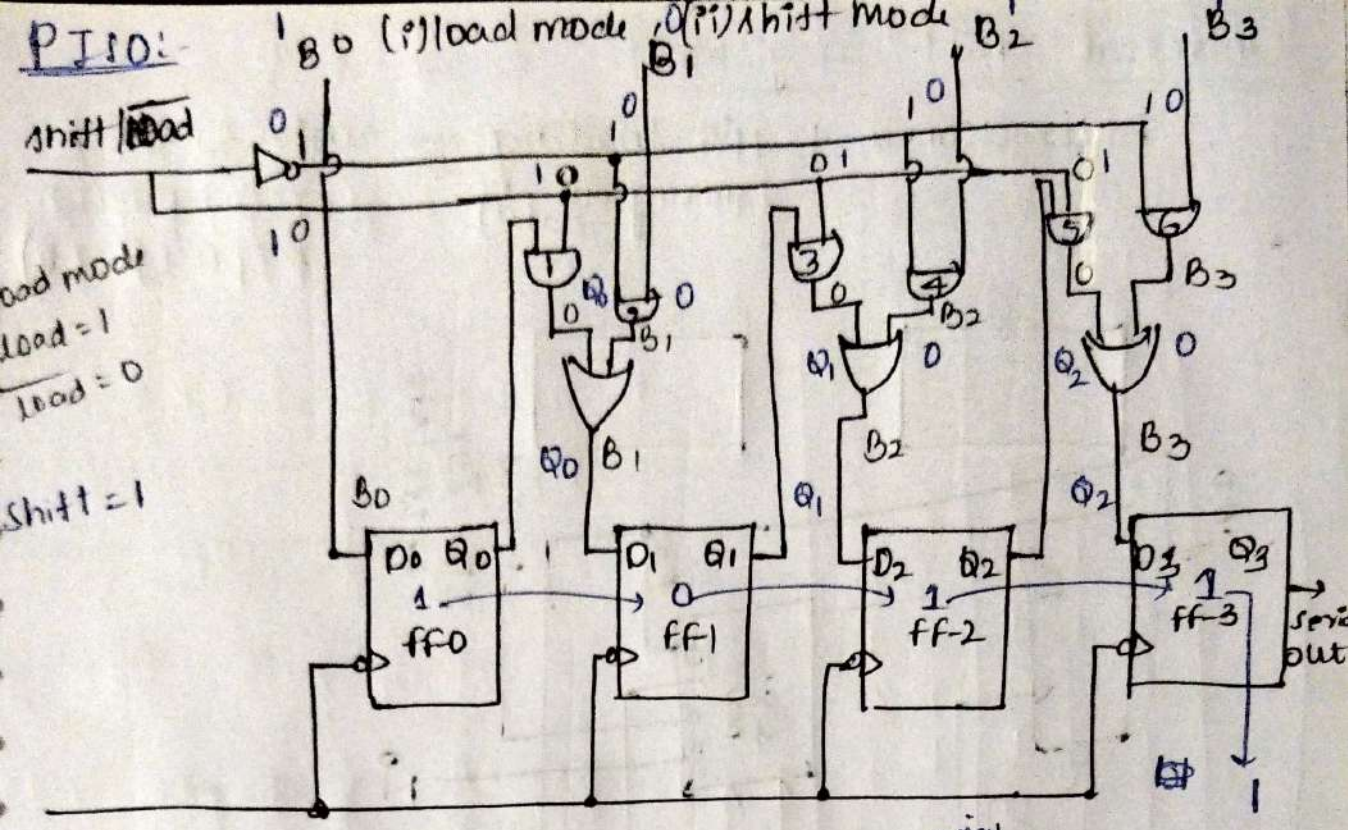
D = x

'1' clock pulse to store the data



SIPO
SISO
] 4 clock pulses

PIO: (i) load mode (ii) shift mode



1 0 1 1 serial out

1 0 1 1

1 0 1 1

1 0 1 1

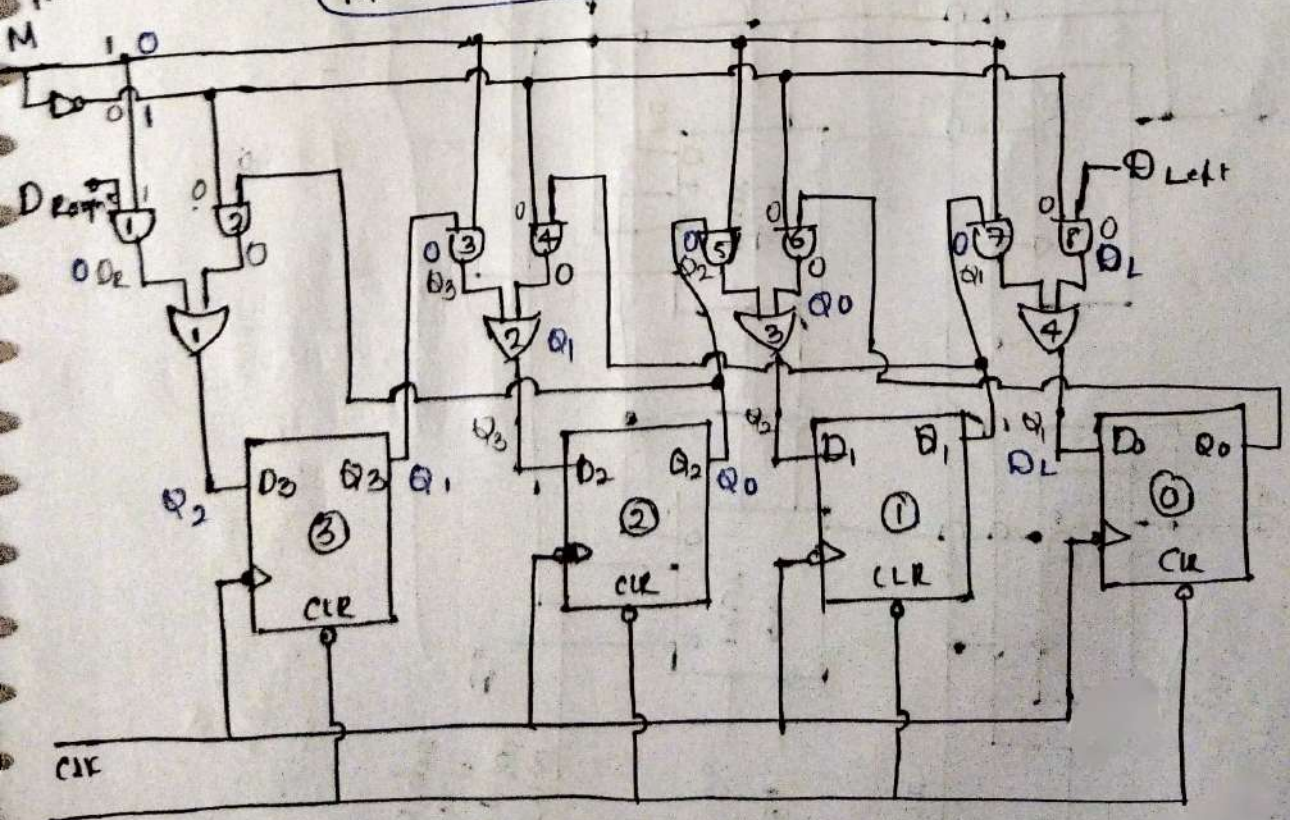
(1 0 1)

Bidirectional shift register

mode control

M = 1 → shift right

M = 0 → shift left



Universal shift register:-

Bidirectional + I/P loading → universal SR
(storing our data) (SISO + PISO + PIPO + SIPO)

Mode control		Reg Op.
S_1	S_0	
0	0	No change.
0	1	shift-right
1	0	shift-left
1	1	11^{b} load

elected
ariably
of
1:1 MUX

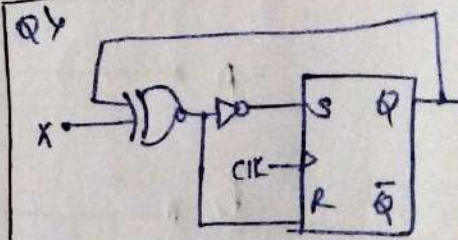
clk	D	Q_{n+1}
0	X	Q_n
1	0	0
1	1	1

Q) A New FF is having behaviour as described below. It has a i/p X & Y & when both i/p's are same & they are 1,1 the ff is going to set else ff reset. If both i/p's are different & they are 0,1 the FF complements itself otherwise it is going to retain the last state. which of the following expressions in the characteristic expression for the new FF?

Sol:

X	Y	
0	0	(reset, o/p=0)
1	1	(set, o/p=1)
0	1	compliment
1	0	memory

Q)



expression for NS Q^+ is

Sol)

o/p = Q^+ (NS)

i/p = X & Q(PS)

$S = X \oplus Q$, $R = X \odot Q$

X	Q	Q^+	S	R
0	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	0	1

$Q^+ = X \oplus Q$

Odd 1's detector

Char Table

(R,S)Q	X	Y	(NS) Q^+
0	0	0	0
0	0	1	1
0	1	0	D
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

for Q^+ :-

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

$Q^+ = \bar{Q}Y + QX$

PLA:- Programmable Logic Array (PLD) device

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

Eg

Step:1

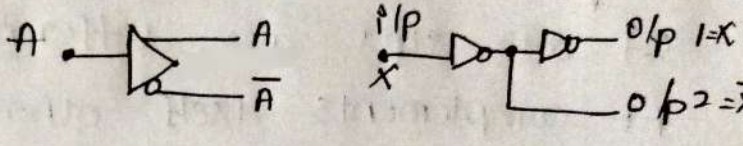
A	B	C	Y_1	Y_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

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

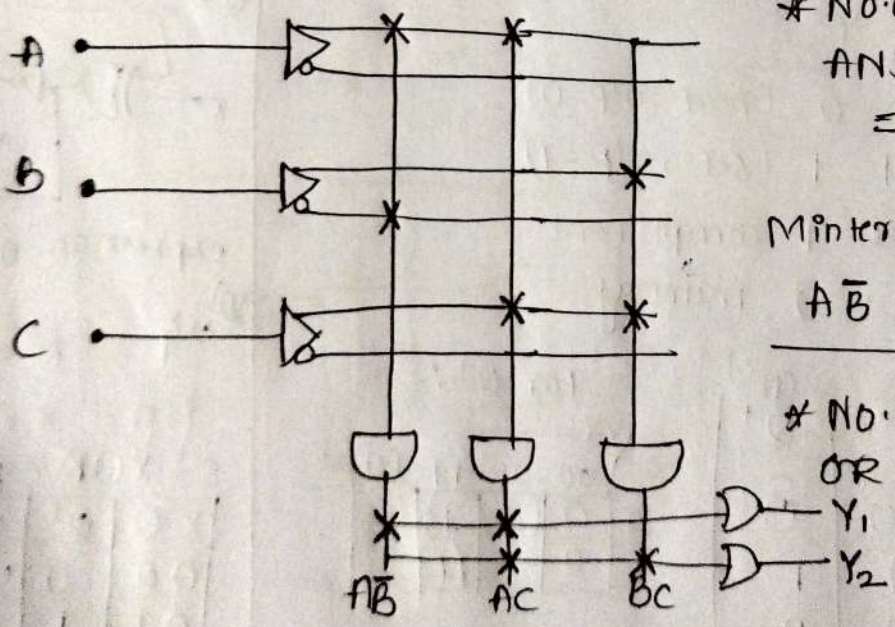
$Y_1 = A\bar{B} + AC$ } Minimal sop form

$Y_2 = BC + AC$ } Step:2

Step 3: No. of i/p Buffers = No. of variables



∴ No. of i/p Buffers = 3



* No. of programmable AND Gates = No. of minterms (not repeated)

Minterms are (3)

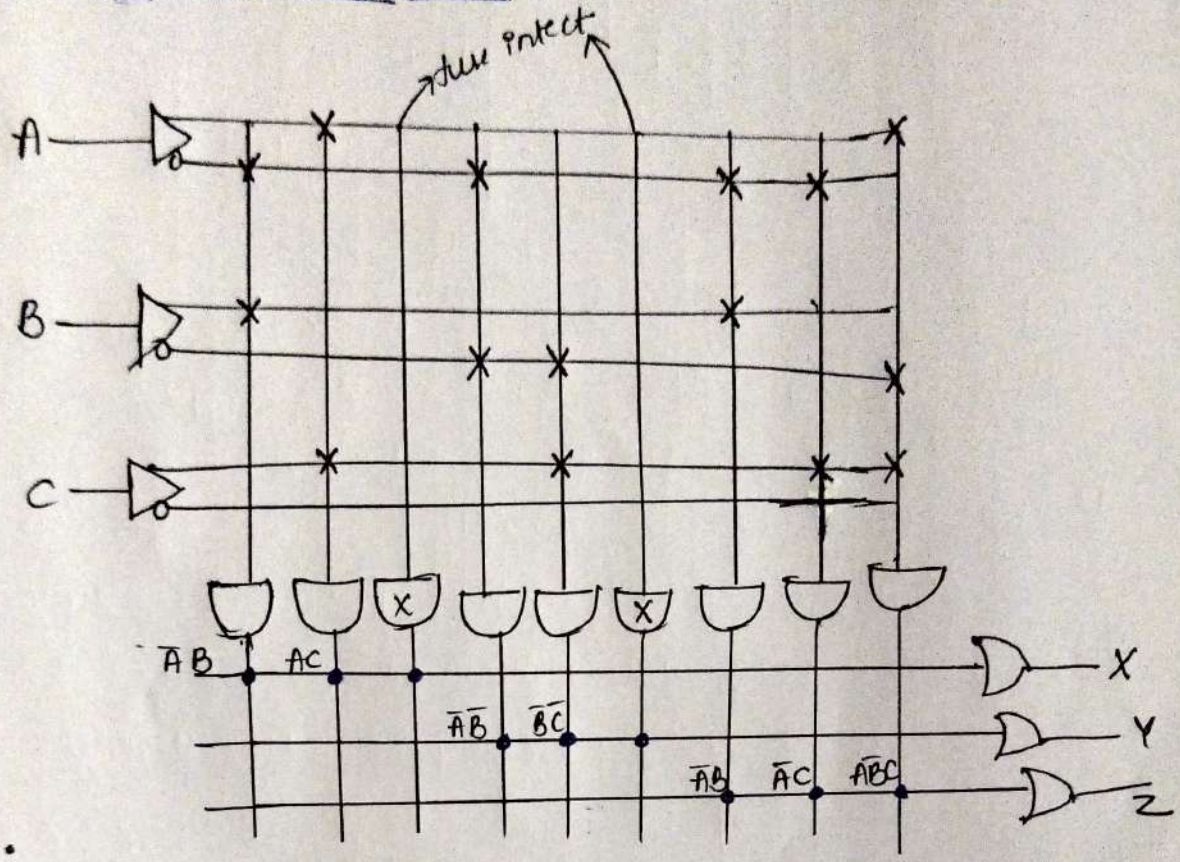
$A\bar{B}$, AC, BC

* No. of programmable OR gate

Programmable Array Logic (PAL):-

→ It is mostly commonly used type of PLD.

→ Has programmable AND array & fixed OR array.



Step 1:

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

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

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

Step 2:- By solving K-maps

No. of variables = 3 = no. of ip buffers (A, B, C)

No. of AND gates unique minterms = {$\bar{A}B$, AC}, {$\bar{A}\bar{B}$, $\bar{B}C$}, {$\bar{A}B$, $\bar{A}C$, $A\bar{B}C$}

= 3

No. of OR gates = O/p = X, Y, Z.

So we have to take 3 AND for each case

total = 9 AND gate