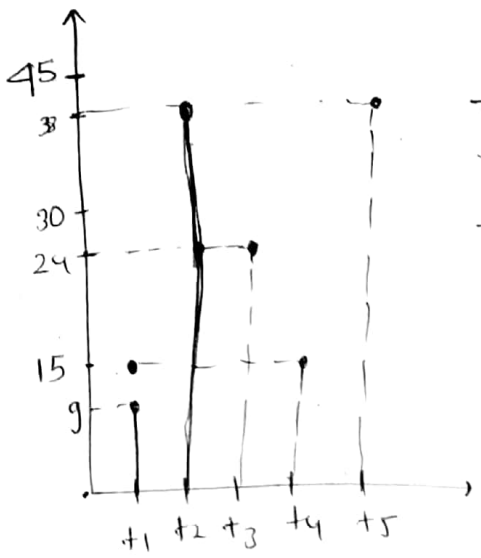
What is digital signal

⇒ In digital signals we discretise both time and magnitude

⇒ We devide the magnitude axis in to fixed numbers of levels and signals can take value equal to these level only.



Discrete time system

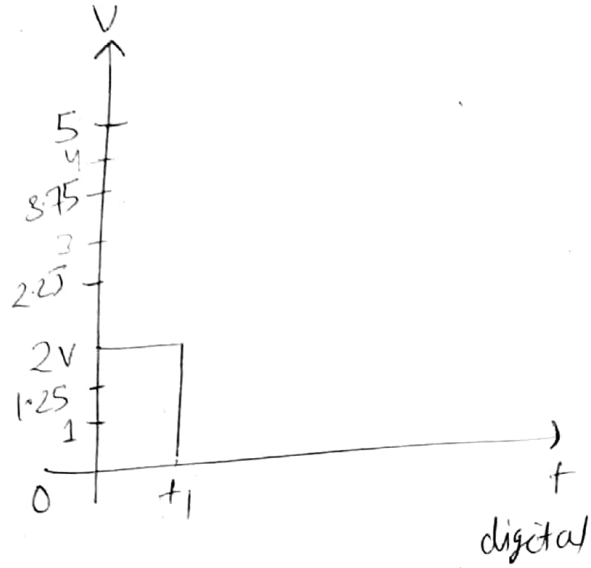
$T(t_1) = 9^\circ C$
$T(t_2) = 38^\circ C$
$T(t_3) = 24^\circ C$
$T(t_4) = 15^\circ C$
$T(t_5) = 45^\circ C$

digital signal

$T(t_1) = 0^\circ C$
$T(t_2) = 30^\circ C$
$T(t_3) = 15^\circ C$
$T(t_4) = 15^\circ C$
$T(t_5) = 45^\circ C$

$\Delta t = t_1 - t_0 = t_2 - t_1 = \dots = t_n - t_{n-1}$

always take



$v(t_1) = 0V$ error of 2 volt

"↑ no. of level ↓ error ↓"

$v(t_1) = 1.95$ V error of 0.75 volt

$v(t_1) = 2V$ error of 0 volt

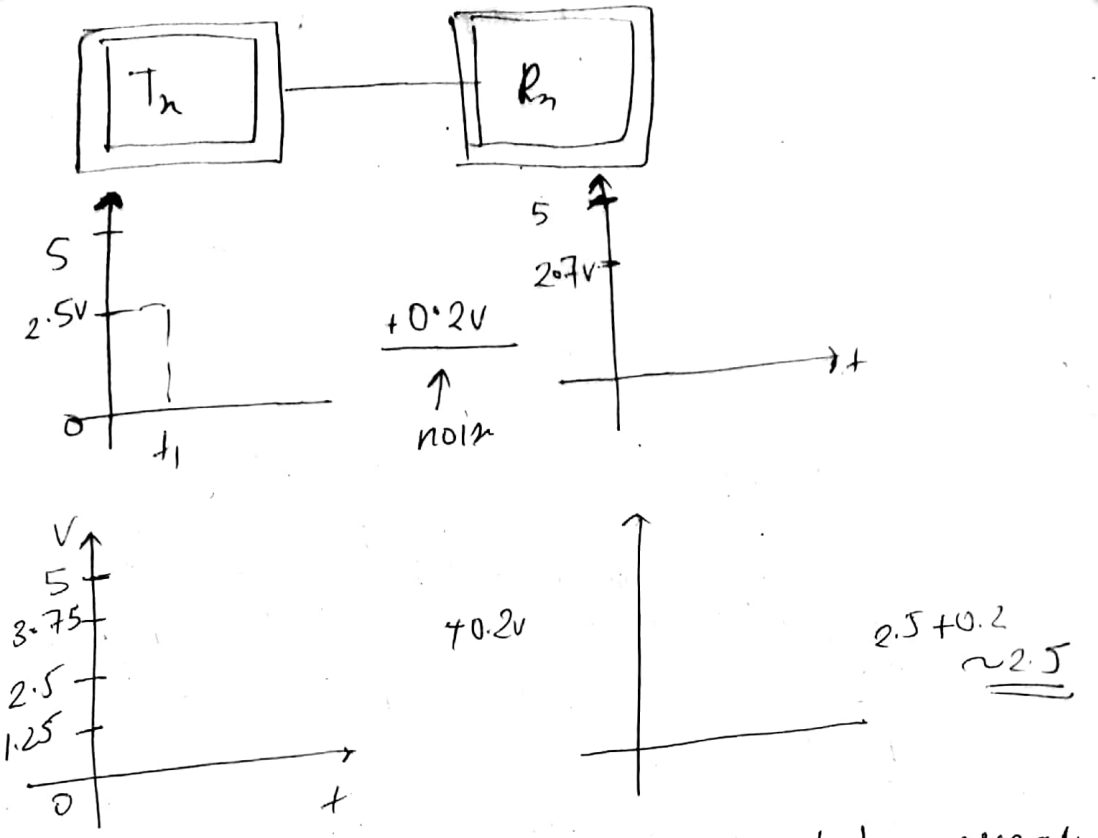
Q What is need of digital signal

Need of digital signal

⇒ All real life signals are analog

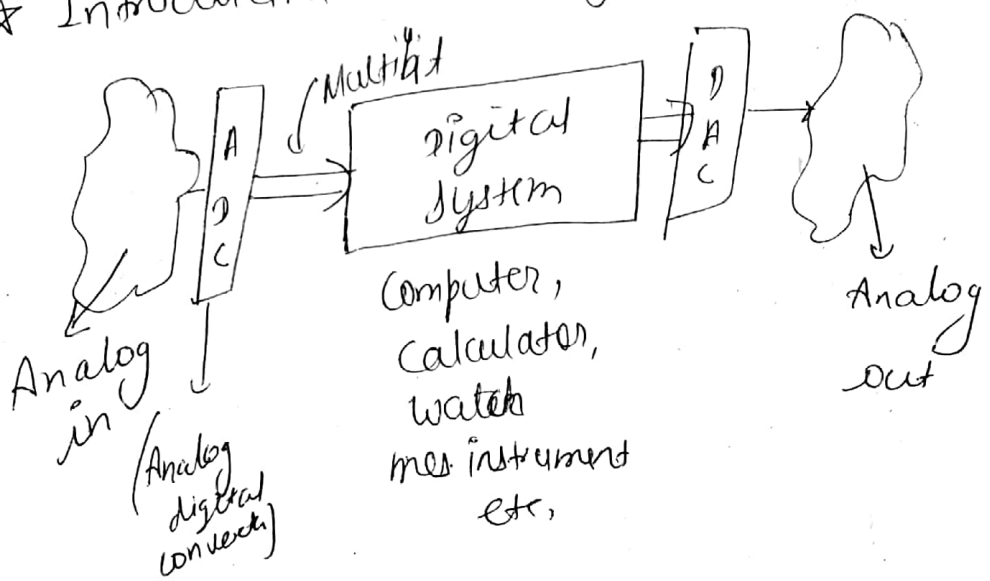
⇒ Digital signal is used in communication process to minimize the effect of noise

↳ Unwanted signal



∴ if noise is small, digital signal has great application

★ Introduction to Digital Electronics



Digital System ~~is made~~

↓

Sub system

↓

Module

↓

Basic unit (logic gate) → circuits (transistor, resistor, Capacitor)

Advantage of digital system

(i) Noise immunity

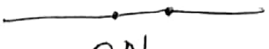
(ii) Uses less bandwidth

(iii) Encryption

(iv) Efficency is highes for long distance transmission.

0 Volt → "0" "OFF"

5 Volt → "1" "ON"



n: no. of levels

m: no. of switches

$n = 2^m$

239 → 3 digit no.

101 → 3 bit no.

no. of switch = no. of bit

accuracy ∝ no. of switch.

0-5V (1024 level)

$2^{10} = 1024$

10 switches.

10 bit

e.g. 720p ~~800~~ 800mb

1080p ~~1680~~ 1.6Gb

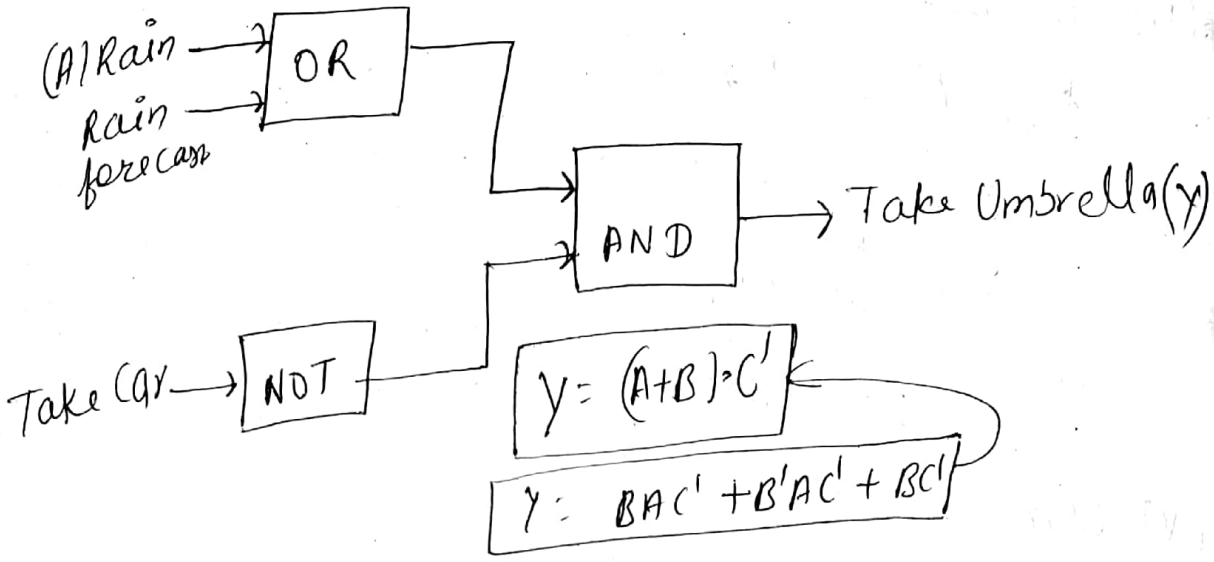
if more pixel, more defination → more accuracy → more bit

Introduction to Boolean Algebra

⇒ It is set of rules, used to simplify the given logic expression without changing ~~or~~ it's functionality.

⇒ It is used when when no. of variable are less

e.g



Rules (i) compliment :-

A compliment = $\bar{A}$ or A' or (not A)

$0' = 1 \quad 1' = 0$

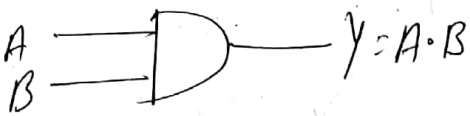
$0'' = 0$ | $1' = 0 \Rightarrow 0' = 1$

$(A')' = A$

(ii) AND

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

$2^2 = 4$



$A \cdot A = A$ | $A \cdot A' = 0$

$A \cdot 0 = 0$

$A \cdot 1 = A$

(III) OR

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

$A+A=A$

$A+0=A$

$A+1=1$

$A+A'=1$

(IV) Distributive law :-

* $A\cdot(B+C) = A\cdot B + A\cdot C$ (operator interchange)
* $A\oplus(B\cdot C) = (A+B)\cdot(A+C)$

e.g. $A+\bar{A}B$

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

$= A+B$

(V) Commutative law :-

$A+B = B+A \quad / \quad A\cdot B = B\cdot A$

(VI) Associative law

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

priority :- NOT → AND → OR

(VII) De-Morgan's law

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

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

$Y = BAC' + B'AC' + BC'$

$= AC'(B+B') + BC'$

$= AC' + BC'$

$(A+B)\cdot C'$

Example

(1) $AB + AB'$

Soln $\Rightarrow$ $F = A(B+B') = A \cdot 1 = A$

(2) $AB + AB'C + AB'C'$

$= AB + AB'(C+C')$

$= AB + AB' = A(B+B') = A$

Time is reduced, power dissipation is reduced

(3) $F = (A+B+C)(A+B'+C)(A+B+C')$

$= [(A+B)+(C \cdot C')](AB'+C)$

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

$A + AB + B'A + AC + BC$

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

$A + AC + BC$

$A(1+C) + BC$

$A + BC$

(4) $G = (A+B)(A+B')(A'+B)(A'+B')$

$[A + (B \cdot B')][A' + (B \cdot B')]$

$A \cdot A' = 0$

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

$A \cdot B + A' \cdot C + BC(A+A') =$ $AB + A'C + ABC + A'BC$

$AB(1+C) + A'C(1+B)$

$AB + A'C$

☆ Redundancy Theorem :- (Consensus Theorem)

(i) Three variable
(ii) Each variable is repeated twice
(iii) One variable is complimented
(iv) Take the complimented variable

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

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

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

$= A\bar{B} + BC$

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

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

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

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

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

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

Sum Of Product (SOP)

	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	

total no. of combination $= 2^n$

↓

SOP form is written only when function is 1

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

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

$F(A,B,C) = \sum m(2,4,5,6,7)$

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

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

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

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

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

$= A + B\bar{C}$

↑ minimal SOP form

Canonical / Standard SOP

using minimal SOP form

★ Sum of Product

Canonical / Standard SOP form

→ Each minterm is having all the variable in normal or complimented form

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

Minimal SOP form: Each minterm does not have all the variable in normal or complimented form

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

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

$= \bar{A}B + AB$

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

$= B$

Simply Given exp.

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

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

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

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

$= A+\bar{B}$

★ Product of Sum (POS form)

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

→ POS form is used when O/P is "0"

$0 \rightarrow A$

$1 \rightarrow \bar{A}$

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

maxterm

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

Compliment of both side

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

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

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

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

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

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

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

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

$= A + B\bar{C}$

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

↑ minimal pos form

product of sum → Canonical form
→ Minimal form

Que :- For the given truth table minimise the pos expression

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

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

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

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

$\bar{B}$

$Y = \Pi(M_1, M_3)$

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

SOP & POS form example

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

OR

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

$Y(A,B,C) = \prod M(1,4,5)$

☆ Minimal to canonical form Conversion.

$Y = A + B'C$

Step 1 → 3 variables A, B & C (that we can say)

Step-2 → m_1 A ✓
B ✗
C ✗

m_2 A ✗
B ✓
C ✓

Step-3 Jo absent ho unki jagah 1 replace kr.
e.g. if B is absent, then replace 1 by $(B+B')$

e.g.

$Y = A + B'C$

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

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

$= (AB + AB')(C+C') + (B'CA + B'CA')$

$= ABC + ABC' + AB'C + AB'C' + B'CA + B'CA'$

$= ABC + ABC' + AB'C + AB'C' + A'B'C$

★ Minimal to Canonical Form Conversion

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

Step-1 → No. of variable & their name

Step-2 → absent variable

term1 - $(A+B+C')$ ↑ all variable are present

term-2

A ✓
B ✗
C ✓

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

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

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

Examples & tricks (SOP & POS forms)

(1) $A+B'C$

No. of minterms in this expression in canonical form

$F = A+B'C$

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

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

$= ABC + ABC' + AB'C + AB'C' + A'B'C$

(5) ✓

Q With 2 variables maximum posible minterms and maxterm.

minterms ↑ r=0

maxterm r=1

A	B	F
0	0	
0	1	
1	0	
1	1	

∴ if all the output is low
⇒ then we get max. no. of maxterm

if all the output is high
⇒ we get max. no. of minterms

n variables

2^n = max/minterms

2^2 = 4 max. value of max & minterms

Q* For n=4, what is total no. of logical expression.

for n(A,B)

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

16 logical expression

2^{2^n} → no. of variables

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

minterm
↑
r=0

☆ Positive and Negative logic

Positive logic

↓

Higher voltage correspond to logic "1"

"5V" ⇒ logic "1"

"0V" ⇒ logic "0"

Negative Logic

Higher voltage corresponds to logic "0"

5V ⇒ logic "0"

0V ⇒ logic "1"

e.g. logic 0 → -5V

logic 1 → 0V

⇒ Higher voltage corresponds to logic 1

∴ positive logic

☆ Dual Form

⇒ +ve & -ve logic AND gate

⇒ + & -ve logic OR gate

⇒ what is dual form

⇒ Rules to write dual form

☆ +ve -ve logic AND gate

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

+ve

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

-ve

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

☆ +ve & -ve logic OR gate

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

A	B	Y
Low	Low	Low
Low	High	High
High	Low	High
High	High	High

0	0	0
0	1	0
1	0	0
1	1	1

↙

(+ve) logic OR gate

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

-ve logic OR gate

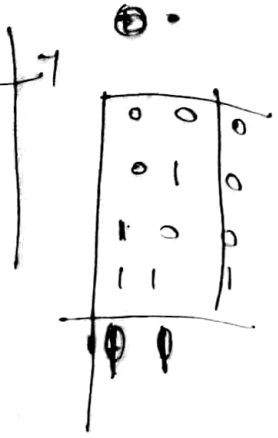
A	B	Y
1	1	1
1	0	0
0	1	0
0	0	0

∴ (+ve) logic AND gate
= (-ve) logic OR gate

(-ve) logic AND gate
≃ (+ve) logic OR gate

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

(+ve) → (-ve) logic is called dual



OR gate

Y
1
0
0
0

✱ Self dual

⇒ For any logic expression, two times dual gives the same expression

⇒ In self dual expression one time dual gives the same expression.

AND ⇔ OR

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

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

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

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

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

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

$AB + AC + BC + BC$

$AB + BC + AC$

trick

(✱) n variables $= 2^{2^{n-1}}$

n: 2

no of self dual expression: $2^{2^{2-1}}$

$2^{2^1} = 4$

A & B

$A \to A \quad B \to B$

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

n-v

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

(2) $n = 4 \qquad 2^{4-1} = 2^{2^3} = 2^8$

✡ Compliment Meaning and Examples.

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

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

$\bar{A}$ or A'

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

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

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

$\bar{F} = ?$

AND ↔ OR

$\cdot \leftrightarrow +$

$0 \leftrightarrow 1$

Compliment of each variable

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

$\bar{A}B = L$ and $\bar{B}A = M$

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

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

$A \oplus B$ $\qquad \bar{A}\bar{B} + AB$

0 $G = \bar{A}BC + \bar{A}B\bar{C} + A\bar{B}C$

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

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

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

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

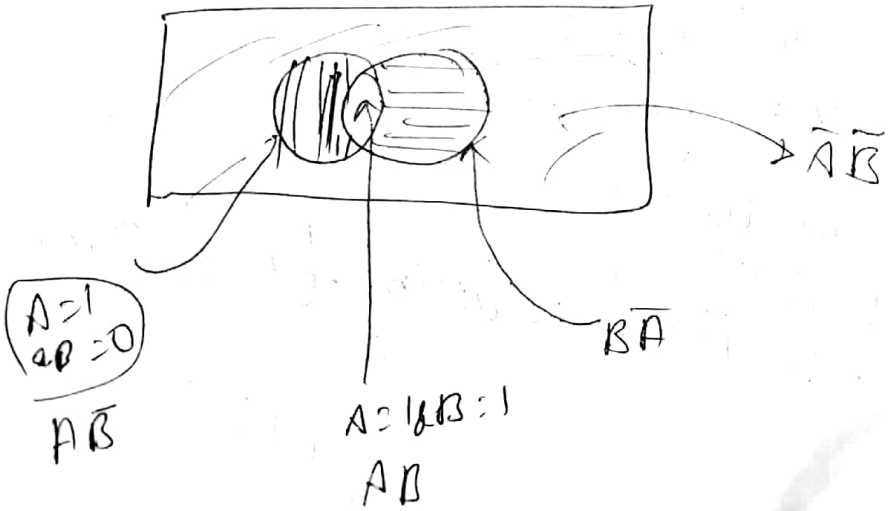
$\bar{A}\bar{B} + \bar{A}\bar{C} + \bar{C}A + \bar{B}\bar{C} + \bar{C}$

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

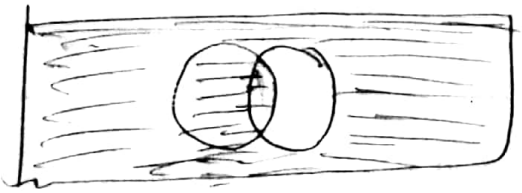
$\bar{A}\bar{B} + \bar{C}$

☆ Venn Diagram

Two variable



☆



Minimize the SOP expression for shaded region.

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

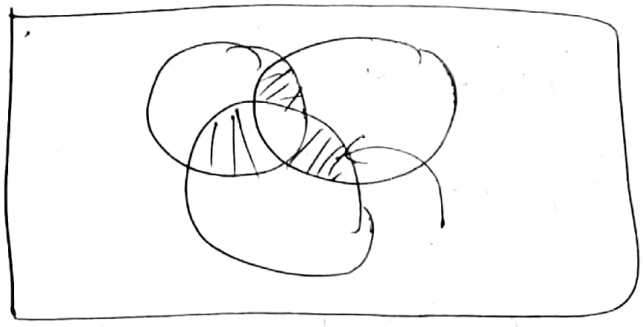
$Y = \bar{B} + AB$

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

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

(1)

(2)



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

★ Switching Circuit

1 - switch → 2 comb^n

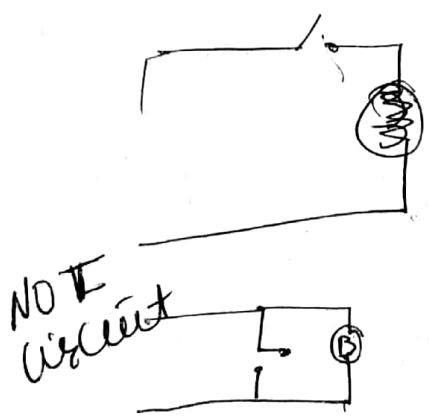
m – switch

$n = comb^m = 2^m$

$\boxed{m = \log_2^n}$

require m switch for n comb^n

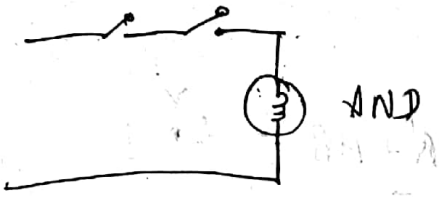
(1)

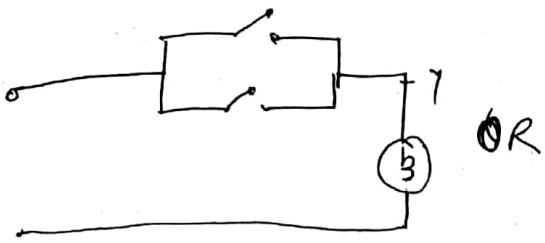


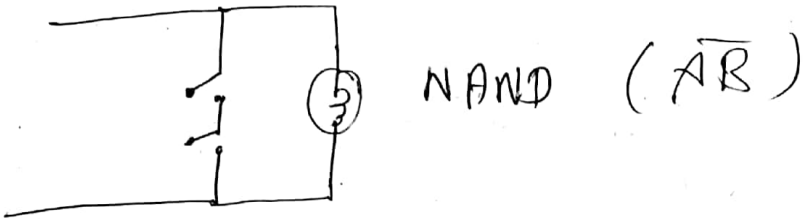
A	Y
0	0
1	0

NOT circuit

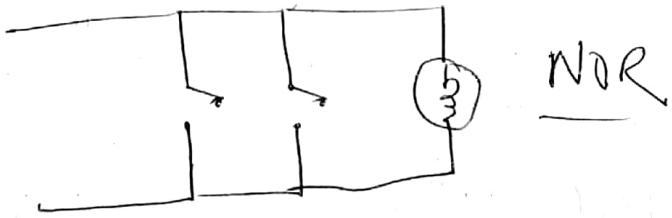
A	Y
0	1
1	0







NOR $(\overline{A+B})$

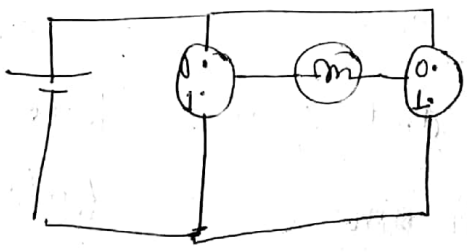


A B (1

(1) Ex-OR Odd 1's detector

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

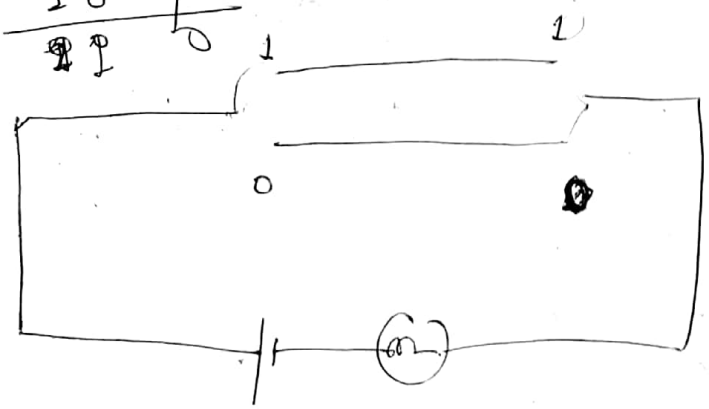
$A \oplus B$

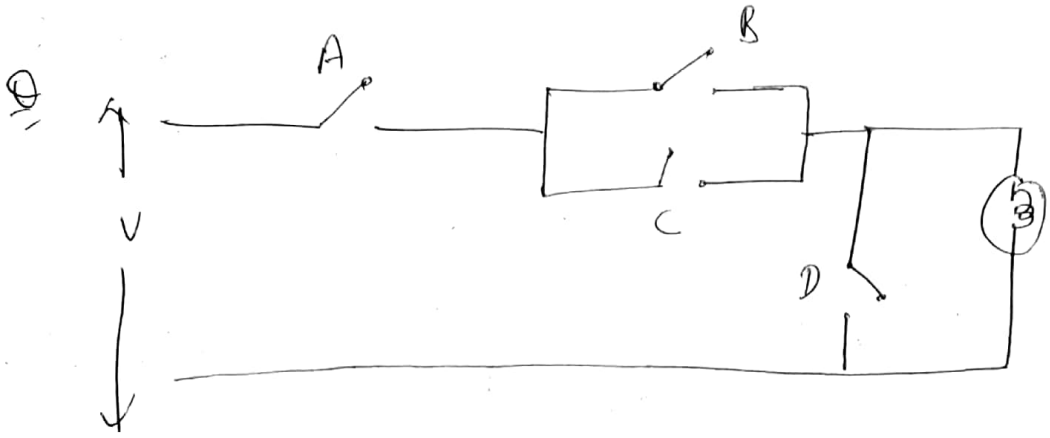


Ex-NOR

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

$A'B' + AB = A \odot B$





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

Statement Problem

(1) Consider a logic circuit with three inputs A, B and C. Output Y is 1 for the following condn.

(I) B and C are true $B \cdot C$

(II) A and C are false $A' \cdot C'$

(III) A, B and C are true $A \cdot B \cdot C$

(IV) A, B and C are false $A' \cdot B' \cdot C'$

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

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

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

Q Y from the condⁿ

(I) A & B are True

(II) A → T
B → F
C → T

$A \cdot B + A B'C + AB'C'$

$A \cdot B + AB'$

(III) A → T
B → F
C → F

A

Q A logic circuit has three inputs A, B & C. The output F is high when the majority of inputs are logic 1.

(a) Minimize the function

(b) Implement the circuit

A	B	C	Y
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

SOP

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

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

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

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

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

$(A + C)B + AC$

$AB + BC + AC$

☆ Introduction to Number System

Name	Base (γ) Radix
Binary	2
Octal	8
Decimal	10
Duodecimal	12
Hexadecimal	16

Number system

- Weighted
  - (1) Decimal
  - (2) Binary
  - (3) Octal
  - (4) BCD
- Unweighted
  - ex. Gray Code
  - Excess
  - etc.

base

γ_1 γ_2

N_1 N_2

quantity

total no. of digits

$\gamma_1 < \gamma_2$

$N_1 > N_2$

Binary No. System

0 to (r-1)

r = 2 0 to (2-1)

* Binary digits (0 & 1) are also called bits

1 0 1 0 1 (every position consider its weights

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

$1X2^4 + 0X2^3 + 1X2^2 + 0X2^1 + 1X2^0$

$16 + 0 + 4 + 0 + 1$

21

$^{4}1\ ^{3}0\ ^{2}1\ ^{1}0\ ^{0}1 . 11$

$1X2^4 + 0X2^3 + 1X2^2 + 0X2^1 + 1X2^0 + X2^{-1} + 1X2^{-2}$

$16 + 4 + 1 + 0.5 + 0.25$

21.75

MSB and LSB (Most significant bit & least significant bit)

MSB ↷ 1 0 1 0 1 → LSB

10100 → 20 (1)

00101 = 5 (16)

left hand is more significant

Octal addⁿ

$$\begin{array}{r} {\scriptstyle 1\ 1\ \ } \\ 5\ 6\ 7 \\ 2\ 4\ 3 \\ \hline 10\ 3\ 2 \end{array}$$

$C_0 \leq 7 \qquad C_0 > 7$

$rem = C_0 \qquad C_0 = 1 \times 8 + 5$

Carry = 0 ↑ Carr ↑ rem

$10 = 1 \times 8 + 2$

$11 = 1 \times 8 + 3$

$8 = 1 \times 8$

Octal subtraction

$$\begin{array}{r} 7\ {\scriptstyle 4}\ {\scriptstyle 3} \\ 5\ 6\ 4 \\ \hline 1\ 5\ 7 \end{array}$$

$$\begin{array}{r} 6\ 2\ 4 \\ -\ 2\ 6\ 5 \\ \hline 3\ 3\ 7 \end{array}$$

$$\begin{array}{r} 6000 \\ -\ 777 \\ \hline \end{array}$$

$7 \times 8^2 + 4 \times 8^1 + 3 \times 8^0$

8×8^0

$$\begin{array}{r} 5\ {\scriptstyle 1}\ {\scriptstyle 1} \\ 3\ 3 \\ \hline 4\ 5\ 6 \end{array}$$

$1\ 8^2\ 8^3 \times 8^0$

$$\begin{array}{r} 6000 \\ -\ 777 \\ \hline 5001 \end{array}$$

$0 \times 8^3\ \ 0 \times 8^2\ \ 0 \times 8^1\ \ 0 \times 8^0$

$7 \times 8^2\ \ 7 \times 8^1\ \ 7 \times 8^0$

$60_8 \; 8_7 \; 8_8$

$- \; 7 \; 7 \; 7$

$5 \; 0 \; 0 \; 1$

Hexadecimal

$0 \; 56.89$

$+ \; 45 \; 74$

$9 \; B \; F \; D$

$1 \; 1$

$A \; D \; D$

$D \; A \; D$

$1 \; 8 \; 8 \; A$

$26 = 1X16+10$

$24 = 1X16+8$

Subtraction

9654

5321

4333

$9 \; 7 \; 4 \; B$

$5 \; 8 \; 7 \; C$

$3 \; E \; C \; F$

16

11 27

12

15

22 13

8 2

14

☆ r's Complement

Complement

r's compliment (Raddix Comp.)

(r-1)'s compliment (Diminished Radix Comb')

Compliment of $(7)_{10}$

Ans : 3

N = 7

n = 1.

r = 10

$r^n - N$ r's compliment

N: 5690

determine 10's compliment

r = 10

N = 5690 n = 4

$10^4 - 5690$

4310

	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 comp. $= r^n - N$

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

(r-1)'s comp + 1 = r's comp.

No borrow

Borrow

7's compliment of Octal no. 5674

8's 5674 + 1 → 10000 - 5674 - 1

$(4096)_8 - 5674 - 1$

7777

Q 9's compliment of 5634

↳ 7's compliment of 5634 → 2103 + 1 = 2104

1's compliment of 1101

1111 - 1101

= 0010

+ 1

0011 → 2's compliment

Shortcut method for 2's compliment

Step-1 write down the given no.

Step-2 Starting from LSB, copy all the zeroes till the first 1

Step-3 Copy the first 1

Step-4 compliment all the remaining bits

Data representation Using Signed Magnitude

Data Representation

- Magnitude
  - Unsigned (+ve) binary no.
  - Signed both (+ve & -ve)
- Complement
  - 1's complement (+ve & -ve)
  - 2's complement (+ve & -ve)

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

Unsigned:

+6 110

−6 → cannot represent

Signed Mag. (0 means +ve)

+6 = 0 110 → Magnitude.

−6 = 1 110 → Sign bit

Sign bit = 0 ⇒ no is +ve

= 1 ⇒ no is −ve

(+13) (−13)

01101 11101

Range of sign magnitude (n → no. of variable)

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

n = 4

$-(2^3-1)$ to $+(2^3+1)$

−7 to 7

1's complement

Here to represent any −ve no. just take one's complement of (+ve) no.

i.e to represent -6, take One's complement of +6

+6 = 0110
-6 = 1001

+0 = 0000 (+ve zero)
-0 = 1111 (negative zero) } problem

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

for 4 variable

$n = 4$

$-(2^3-1)$ to (2^3-1)

☆ Using 2's compliment

+6 = 0110

to represent -6, take 2's compliment of 6

1's complement = 1001

+1

1010 2's comp.

→ only have One zero

0000

1's → 1111

+1

2's → 10000

Binary subtraction using 1's compliment

$A - B \rightarrow A + (-B)$

↑ 1's compliment of B: $-B$

Perform the addn

if the final carry is 1, then add 1 to result

if the final carry is 0, result obtained is -ve and in 1's compliment form

$(1100)_2 - (0101)_2$

A = 1100

B = 0101

1's of B = 1010

12
−5
7

1100
1010
(1) 0110

9 2 1

9 1 0
+ 1
0 1 1

~~0100~~

$(0101)_2 - (1100)_2$

A = 0101 B = 1100

-B = 0011

1 0111

0101
0011
01000

final carry is 0

Binary subtraction using 2's complement

A - B = A + (-B)

↑ 2's complement of B

⇒ If final carry is generated then result is +ve and is in pure form

If no final carry is -ve and is in 2's complement

final carry → is always neglected in 2's complement

Ex: $(1011)_2 - (0100)_2$

A = 1011 B = 0100

2's of B = 1011 + 1 = 1100

A + (2's of B): 1011 + 1100 = (1)0111

result is +ve & neglect it

0111 is our answer

(5) //

Condⁿ for over flow

x and y are sign bit of two no.

x is the sign bit of result

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

$= 1$ (overflow)

$(0110)_2 - (1011)_2$ (11)

A = 0110 B = 1011

$-B$: 0100 → +1 → 0101

0110 (6)

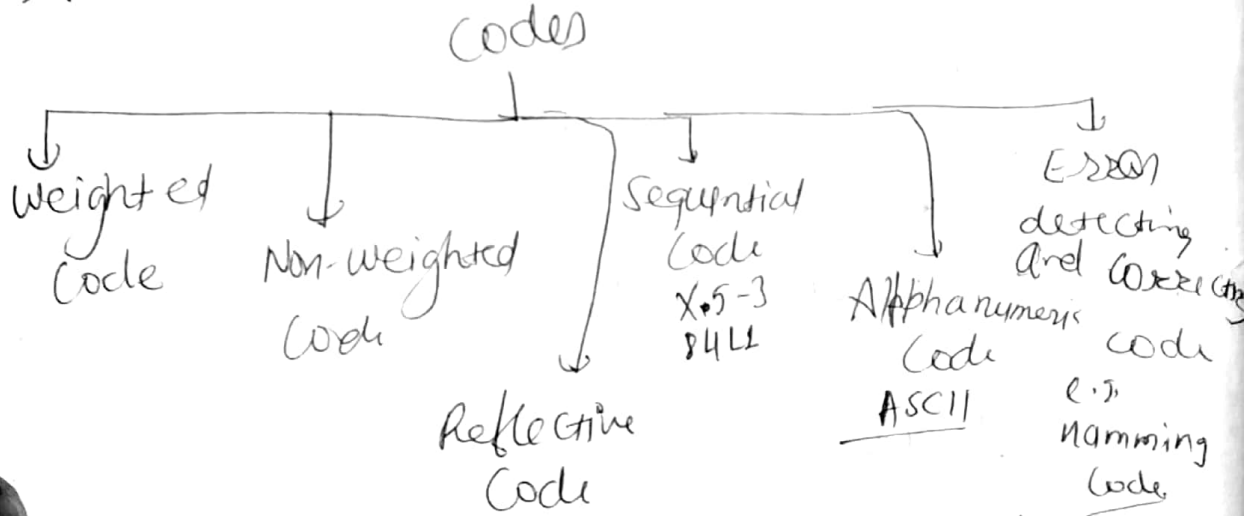
0100

1011 ← (5)

1's comp 0100

0101 (5)

☆ classification of codes



Reflective code → 9 → compl. of 0

8 ⟶ 1, 1

7 ⟶ 1, 1, 2

Decimal	2421 Code
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	1 0 0 1
6	1 0 1 0
7	1 0 1 1
8	1 1 0 0
9	1 1 1 1

(any mistake acceptable)

9 → Comp. of 0
8 → Comp. of 1

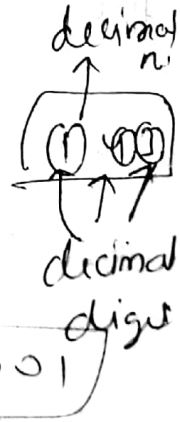
☆ ASCII code

↳ 7 bit code (learn everything)

☆ Binary Code Decimal (BCD) Code

→ In this code each decimal digit is represented by a 4 bit binary no.

→ positional weight are 8-4-2-1



Decimal	BCD 8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0

10 to 15 are invalid as these are decimal no not a decimal digit

② Conversion of decimal no to BCD

(1) $(17)_{10}$ → (00010111) BCD

(2) $(156)_{10}$ → 000101010110)

③ Convert BCD to decimal

(00100100) BCD

$(14)_{10}$

④ Comparision b/w binary and BCD

	binary	BCD
$(10)_{10}$ →	1010	00010000
$(12)_{10}$ →	1100	00010010

* BCD is less efficient than binary (require more bits)

BCD Addn

① Sum ≤ 9, Final carry = 0 Answer is correct

② Sum ≤ 9, Final carry = 1 Ans. is incorrect
add 6 (0110)

③ Sum > 9, Final carry = 0 Ans. Incorrect (add 0110)

binary addn

	S	C
0 + 0 =	0	0
0 + 1 or 1 + 0 =	1	0
1 + 1 =	0	1

e.g $(2)_{10} + (6)_{10}$ BCD addn.

```
  0010
+ 0110
------
  1000 (8)
```

Sum = 8 Sum<9 F.C=0

Ans

e.g (2) $(3)_{10} + (7)_{10}$

```
  0011
+ 0111
------
  1010
```

8+2=10

binary rep. Sum>9

F.C=0

6 is added as 0-9, 0-15

6 cases are considered as invalid

```
 1010
 0110
------
10000
```

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

```
 1000
 1001
------
10001
```

F.C=1

Sum<9

```
 10001
 0110
------
 10111
```

(1) (7)

e.9 $(57)_{10} + (26)_{10}$

(0 1 0 1 0 1 1 1) (57)

+ (0 0 1 0 0 1 1 0) (26)

0 1 1 1 1 1 0 1

7 9

7 13 79

0 1 1 0

1 0 0 1

0 1 1

0 1 1 1 1 1 0 1

0 0 1 0

1 0 0 0 0 0 1 1

8 3

1 0 0 0 0 0 1 1

8 3

$(83)_{10} + (34)_{10}$

8 3

3 4

1 1 7

1 0 0 0 0 0 1 1

+ 0 0 1 1 0 1 0 0

1 0 1 1 0 1 1 1

+ 0 1 1 0

1 0 0 0 1 0 1 1 1

1 2 7

Shift ADD-3 Method

Operation	Tens	ones	Decimal
original no.			1111
Shift		1	111
Shift		11	11
Shift		111	1
Add-3		+011	
		1010	1
Shift	1	0101	
			Stop.

16

it is 4 bit no.

☆ 2421 Code → BCD Code

Decimal Digit	Set 1 (2 4 2 1)	Set 2 (2 4 2 1)
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

2421
(1) 1001

(we will take low significant bit)

Self Complementing Code is also called reflective code

Decimal	7421	5421	3321	8421	7421
0	0000	0000	0000	0000	0000
1	0001	0001	0001	0111	0111
2	0010	0010	0010	0110	0110
3	0011	0011	0100	0101	0101
4	0100	0100	0101	0100	0100
5	0101	1000	0110	1011	1010
6	0110	1001	1100	1010	1001
7	0111	1010	1101	1001	1000
8	1001	1011	1110	1000	1111
9	1010	1100	1111	1111	1110

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

2421

0111

1101

(3)

00111101

0100 1100 1110

2421

468

✡ Excess-3

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

$\underbrace{011}_{3}$

A ⟶ BCD ⟶ A+3

5 → 0101

0011

(8) 1000

XS-3 or X-3

XS-3 code is unweighted code

⟶ 4 bit code

Decimal	8421 BCD	XS-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

XS-3 code is only unweighted code which is self complimen

XS-3 code for decimal no:-

(124)

00100100 →

0010 0100

0011 0011

0101 0111 → XS-3 code for 24

(4) 6 5 3

0110 0101 1000
0011 0011 0011
1000 1000 1011

(2)

(*) w_1, w_2, w_3, w_4

$w_1+w_2+w_3+w_4 = 9$ → self complementing

$\neq 9$

2-4-2-1

$2+4+2+1 = 9$ ✓

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

Excess-3 code addⁿ

Ex 1 $(2)_{10} + (5)_{10}$

$(2)_{10} = 0010 \rightarrow 0101$ XS-3

$(5)_{10} = 0101 + 1000$ XS-3

1101

⊗ → XS-6

−0011

1001

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

↓

0010 0111
0011 0011
0101 1010

$(39)_{10}$ → 0011 1001
0011 0011
0110 1100

① → Finde carry!

0101 1010
0110 1100
1100 1110

G-1 ⇒ F.C = 1 → we will add 0011

G-2 ⇒ F.C = 0 → we will substract 0011

1100 0110

− 0011 + 0011

1001 1001

99

27

39

66

33 (69)

99

$(99)_{10} + (9)_{10}$

1001 1001

+ 0011 0011

1100 1011

1001

0011

1100

(6) 1

11001011

00001100

11010111

−0011 1001

10001000

8

99

9

108

333

441

☆ Introduction to Grey code

⇒ Also known as Reflected binary code :-

⇒ we call it gray code after Frank Gray

⇒ Unweighted code

⇒ Unit distance & Minimum Error code

⇒ Cyclic code

Two ~~Success~~ succ. value differ in only 1 bit. Binary no. is converted to gray code to reduce switching.

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

☆ Binary

Step 1

Step 2

Step 3

MS

Eg (2)

☆ G

Step

Step 2

Step

☆ Binary to Grey Code Conversion

Step 1 Record the MSB as it is

Step 2 Add the MSB to the next bit, record the sum and neglect the carry (X-OR)

Step 3 Repeat the process

MSB $b_3\ b_2\ b_1\ b_0$ LSB

1 0 1 1

1 1 1 0 Grey code

$g_3\ g_2\ g_1\ g_0$

$g_3 = b_3$

$g_2 = b_2 \oplus b_3$

$g_1 = b_1 \oplus b_2$

$g_0 = b_0 \oplus b_1$

Eg ②

1 1 1 0

1 0 0 1 → Grey

☆ Grey code to Binary Conversion

Step 1 Record the MSB as it

Step 2 Add MSB to the next bit of Grey code, record the sum and neglect the carry

Step 3 Repeat the process.

MSB → 1 1 1 0

1 0 0 1

MSB → 1 1 1 0

1 0 1 1

$g_3 g_2 g_1 g_0$ ← Gray code

1 0 0 1

1 1 1 0 → BCD

$b_3 b_2 b_1 b_0$

$b_3 = g_3$

$b_2 = b_3 \oplus g_2$

$b_1 = b_2 \oplus g_1$

$b_0 = b_1 \oplus g_0$

What is parity

→ It is concepts to detect errors

→ A single bit error is detected by it only

We send a bit stream and also an extra bit tells us total 1's

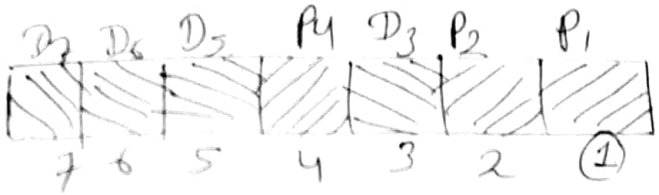
two types → (1) even
(2) odd

	original signal	parity bit
even parity	0 1 0 0 (total odd) so add 1 in parity bit)	1
	1 1 0 0 (total even)	0
odd parity	0 1 0 0	0
	1 1 0 0	1

if noise has been added (suppose one 0 ≠ 0 become 1, hence, it will disturb the no. of 1's, so contradicts the parity condn, hence, we can detect error).

✡ Hamming Code – Error Detection

→ Given by R.W. Hamming
→ Easy to implement
→ 7-bit hamming code is used commonly



Rule

Data bits :- 4
parity bits :- 3

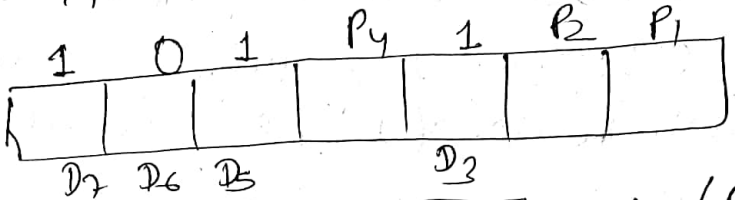
2^n (where $n = 0, 1, \ldots n$) will be position for ~~parity bits~~

$2^0 = 1$ $2^1 = 2$ $2^2 = 4$

$P_1 \rightarrow D_3\ D_5\ D_7$
$P_2 \rightarrow D_3\ D_6\ D_7$
$P_3 \rightarrow D_5\ D_6\ D_7$

} given by Hamming (learn)

1011



$P_1 \rightarrow D_3 D_5 D_7 | 111 \Rightarrow 1$ (even parity)

$P_2 \rightarrow D_3 D_6 D_7 0 | 01 \Rightarrow$

$P_4 \rightarrow D_5 D_6 D_7 0 | 101 \Rightarrow$

∴ transmitter bit stream will be

1	0	1	0	1	0	1

Hamming Code – Error Correction

Q. If the 7-bit hamming code word received by a receiver is 1011011. Assuming the even parity state whether the received code word is correct or wrong. If wrong locate the bit having error.

1011011

$D_7 D_6 D_5 P_4 D_3 P_2 P_1$

$D_3 D_5 D_7$

P_1 (1) 0 1 1 P_1

P_2 (1) 0 0 1 P_2

(1) 1 0 1 P_4

$D_5 = 0$ is a correction.

$P_1P_2P_3$: $(101)_2 = (5)_{10}$

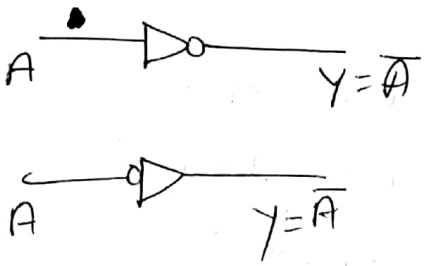
5th bit bcos error

★ logic gate

⇒ physical device → performs logic operation on one or more inputs and produce a single ~~input~~ outputs.

① Basic gate → NOT, AND, & OR
② Universal gates → NAND, NOR
③ Arithmatic gates → X-OR & X-NOR

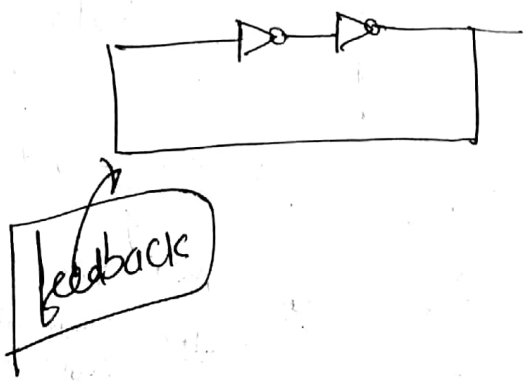
★ NOT gate



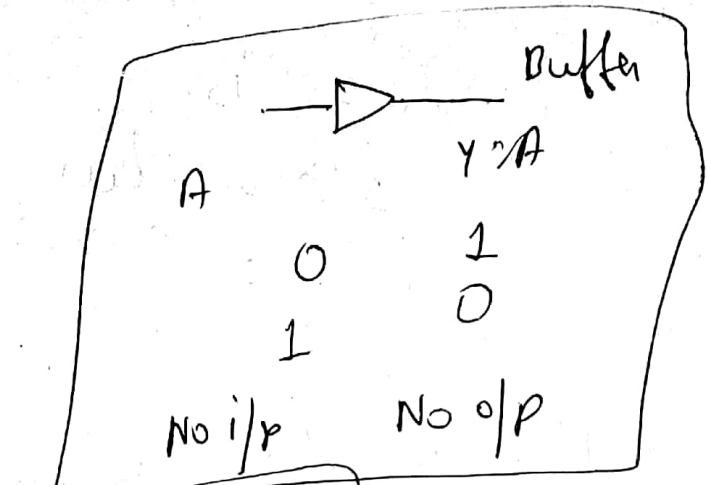
Truth Table
⇓
Table having outputs from all the possible comb^n of inputs

A	Ā
0	1
1	0

Q.1 IES 2010 [Gate 2010]



Y ~~Logic~~ Buffer
(a) Bistable MV
(b) Astable MV
(c) Square wave generation



Question expln

if input = 1 output is 1

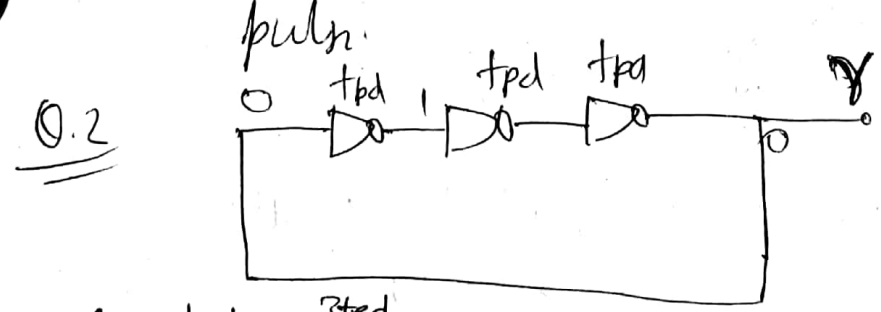
because of feedback

∴ it is not for all input

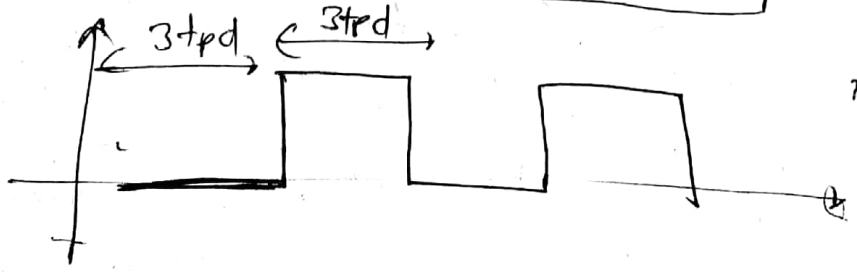
Astable → in which the circuit is not stable in either state, it continually switches from one state to the other.

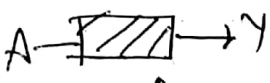
Bistable → in which the circuit is stable in either state. In can be flipped from one state to the other by an external trigger pulse.

0.2



t_{pd} ↓ propagation delay





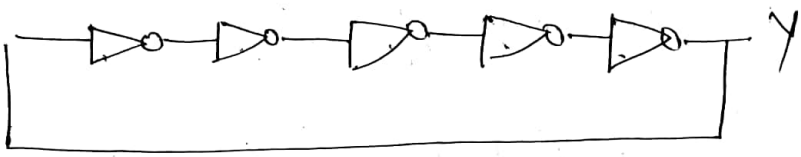
↑ Not give simultaneous results

Time period: $2 \times n \times t_{pd}$

(n ↑ no. of not gate)

∴ Y is sq. wave function

Q In the circuit shown, the propagation delay of each NOT gate is 100 psec. Then frequency of generated square wave is.

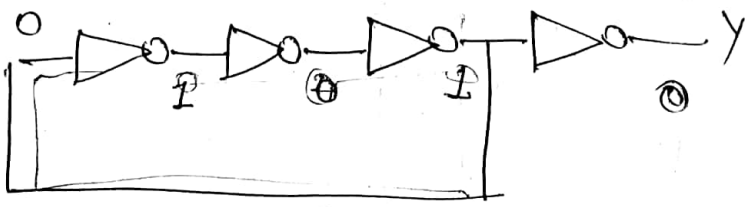


$$\text{frequency} = \frac{1}{\text{Time period}} = \frac{1}{100 \times 2 \times 5 \times 10^{-12}}$$

$$\frac{10^2}{10^3} = 10^9$$

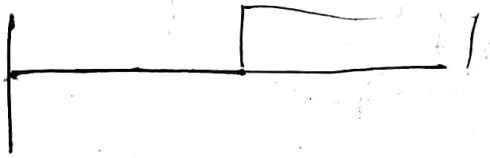
$$= 1\text{GHz}$$

Q In the circuit shown, the propagation delay of each NOT gate is 2nsec. then time period of generated square wave is



$t_{pd} = 2 \times n$ ns

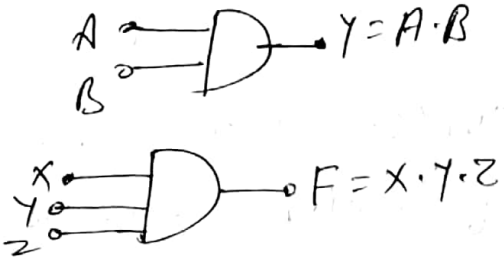
$T = 2 \times 4 \times 1$



☆ AND operator

Output is high, when all the Inputs are high

① Symbol



② Truth table

A	B	Y = A·B
0	0	0
0	1	0
1	0	0
1	1	1

③ Associative Law :-

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

④ Commutative Law :-

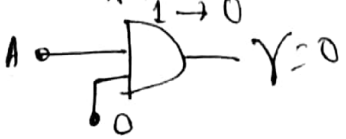
$A \cdot B = B \cdot A$

⑤ Enable & Disable

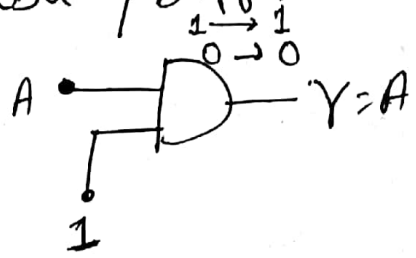
	A	B	Y
disable o/p	0	0	0
	0	1	0
Enable buffer	1	0	0
	1	1	1

"0" → disable with o/p = 0

A 0 → 0, 1 → 0



'1' → enable / buffer

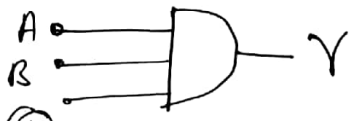


⑥ Unused enable

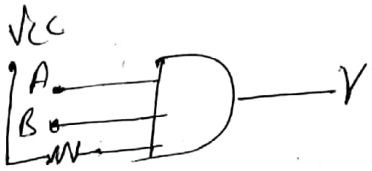
(*) In TTL logic, if any i/p is open or floating it will act as "1".

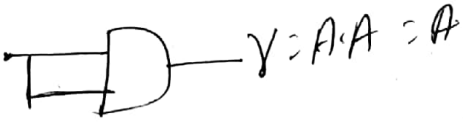
(*) ECL → "0"

(i)

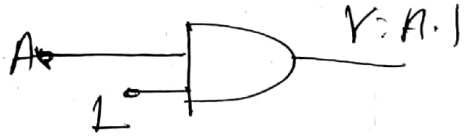


(1) is act as enable for AND gate



(ii) 

(iii) In TTL logic



✡ logic gate (OR)

① 

② Associative law

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

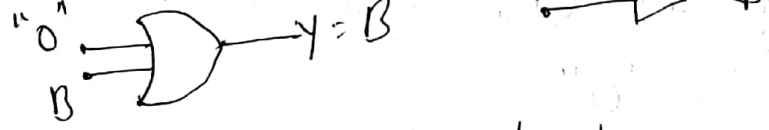
③ Commutative law

A+B = B+A

(5) Enable & Disable

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

"0" → enable / Buffer



1 → Disable with o/p = 1

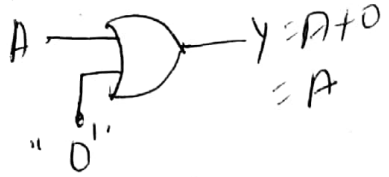
(6) Unused input

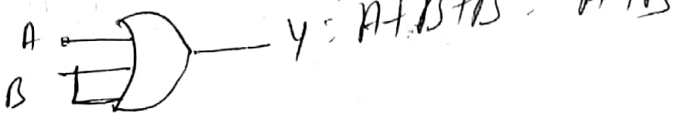
TTL → "1"

ECL → "0"

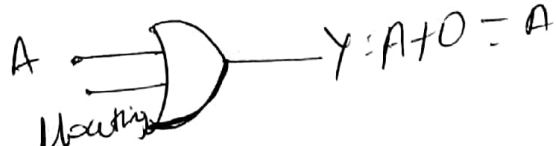
imp.

(i) Connect to "0"

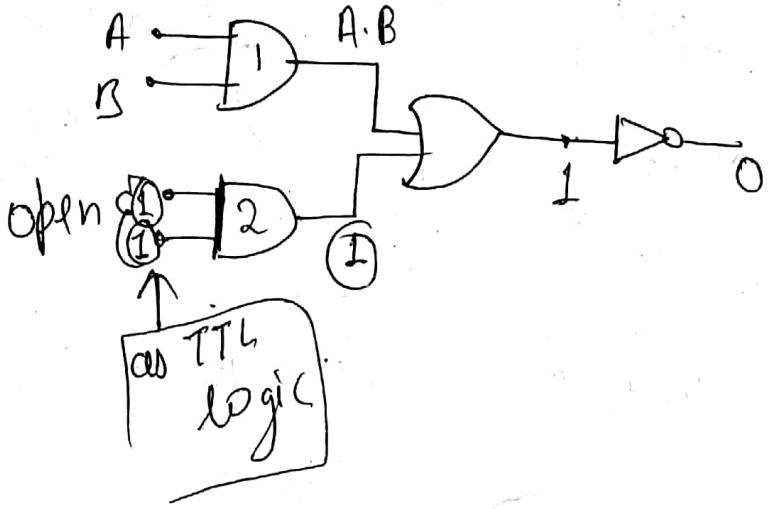




(3) ECL logic

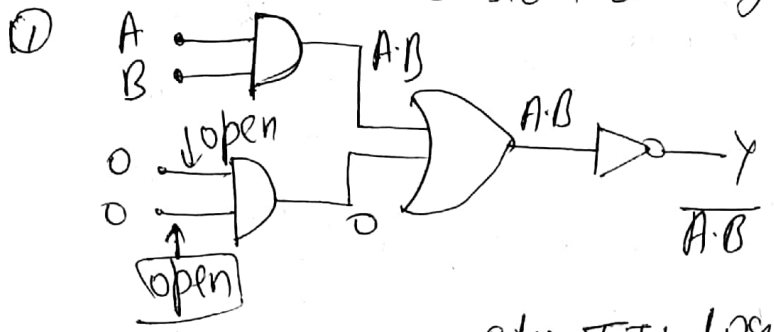


Problem Consider TTL logic

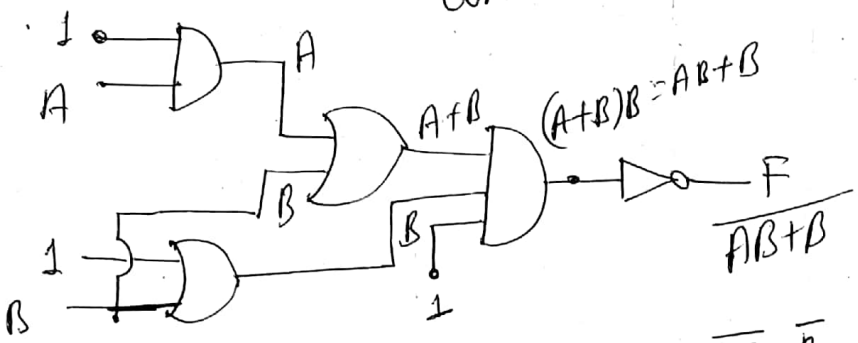


H.w

Consider ECL logic



Consider TTL logic



$\overline{AB}\ \overline{B}$

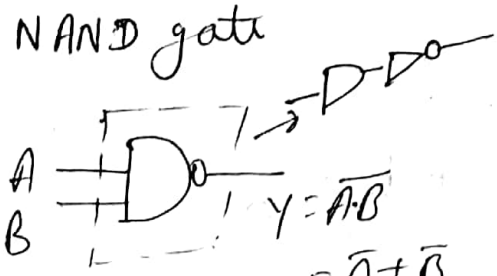
$(\bar{A}+\bar{B}) \cdot \bar{B}$

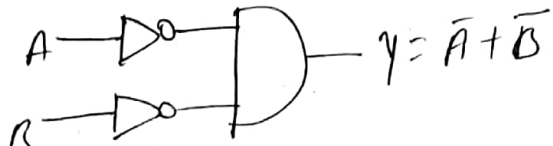
$\bar{A}\bar{B}+\bar{B}$

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

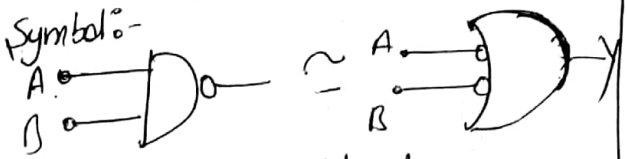
Logic gate

NAND gate





Symbol :-

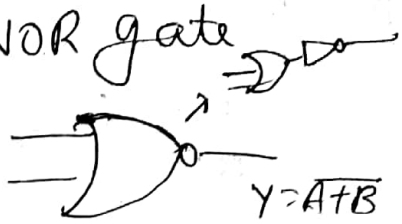


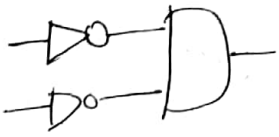
it is also called bubbled OR gate

T.T

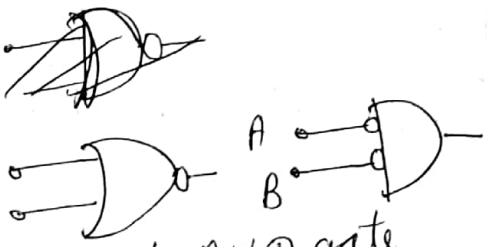
A	B	Y	
0	0	1	disably inverted
0	1	1	
1	0	1	enable inverted
1	1	0	

NOR gate





Symbol :-



Bubbled AND gate

T-T

A	B	Y	
0	0	1	enable inverted
0	1	0	
1	0	0	disably inverted
1	1	0	

(*) Association law (X)

(*) Commutative law ✓

(*) Unused input
Same as AND gate

(X) Commutative law ✓

(X) Same as OR gate

① Prove NAND & NOR doesn't follow the association rule

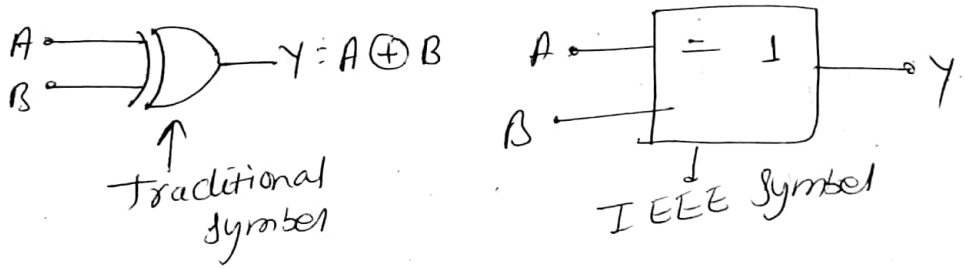
$\overline{(A \cdot B) \cdot C} = A \cdot B + \bar{C}$

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

→ both are diff.

★ Logic gate

Exclusive OR gate Ex-OR / XOR



Internal circuit

$Y = A \oplus B$

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

Truth table

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

Output is high when odd no. of 1

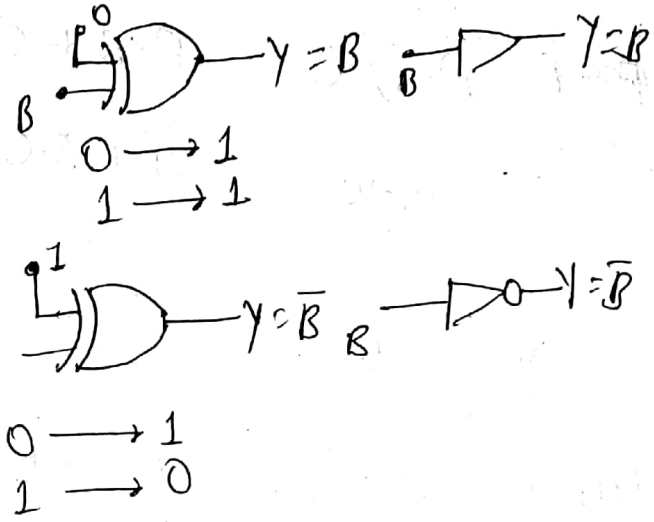
⇒ odd 1's detector

nable and disabl

imp

Controlled inverter

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



properties

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

$A + \bar{A} = 1$

(i) $A \oplus A = 0$

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

(iii) $A \oplus 0 = A$

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

(v) If $A \oplus B = C$, then

$B \oplus C = A$

$A \oplus C = B$

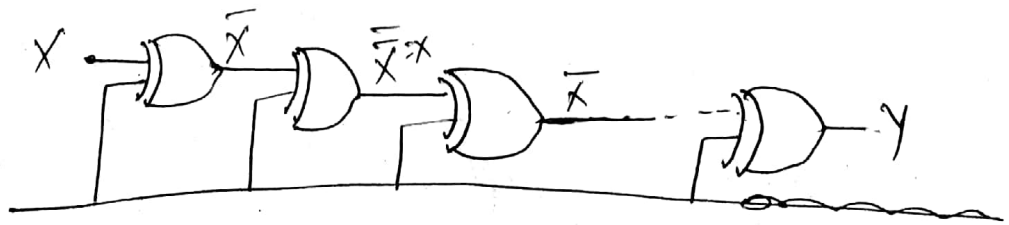
$A \oplus B \oplus C = 0$

(vi) $A \oplus A = 0$

$A \oplus A \oplus A = 0 \oplus A = A$

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

$= 0$ n is even



if odd no. of XOR then $\bar{X}$
if even no. of XOR then X

(11) $Y = A \oplus \bar{A} \oplus \bar{A} \oplus A \oplus A \oplus \bar{A} \oplus \bar{A} \oplus A \oplus A$

$1 \oplus \bar{A} \oplus$ ———

$A \oplus A \oplus A$

$0 \oplus A \oplus \bar{A} + - -$

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

$\bar{A} \oplus A \oplus A$

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

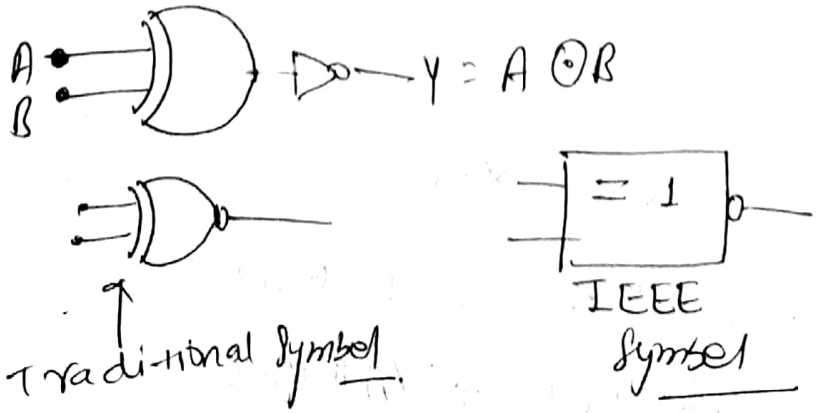
(11) $f(A,B) = A \oplus B$

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

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

$x \oplus y \oplus z \oplus w$

★ Exclusive NOR gate Ex-NOR



Truth table

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

(enable inverter ← rows 0 0 and 0 1; enable buffer ← rows 1 0 and 1 1)

when A = B o/p = 1

when A ≠ B o/p = 0

XNOR follow the commutative & Associative law

Properties

(i) $A \odot A = 1$

(ii) $A \odot \bar{A} = 0$

(iii) $A \odot 0 = \bar{A}$

(iv) $A \odot 1 = A$

(v) $A \odot A \odot A = 1 \odot A = A$

$A \odot A \odot A \odot A \cdots = A$, n is odd

$= 1$, n is even

(vi) X-OR & XNOR are complement → even i/p

X-OR & XNOR are equal → odd i/p.

$AA + \bar{A}\bar{A}$

$A + \bar{A} = 1$

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

$A \cdot 0 + \bar{A} \cdot 1$

A

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

$A \oplus B \oplus C \oplus D = \overline{A \odot B \odot C \odot D}$

$A \odot B \odot C = \otimes \odot C$

$= X \cdot C + \bar{X} \cdot \bar{C}$

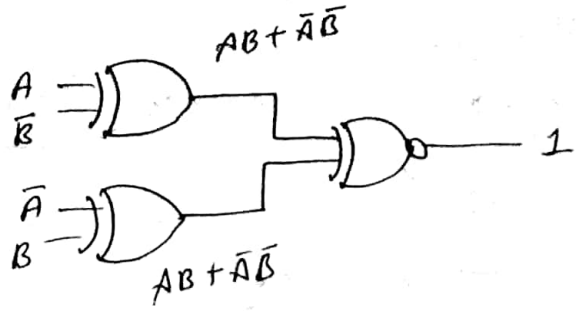
$(A \odot B) \cdot C + (\overline{A \odot B}) \cdot \bar{C}$

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

$\bar{Y} \cdot C + Y \cdot \bar{C}$

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

Q



A⊙A=

(2) $Y = A \odot A \odot A \odot A . \odot \bar{A} . \odot A$

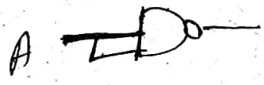
$= 1 \odot \bar{A} \odot A$

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

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

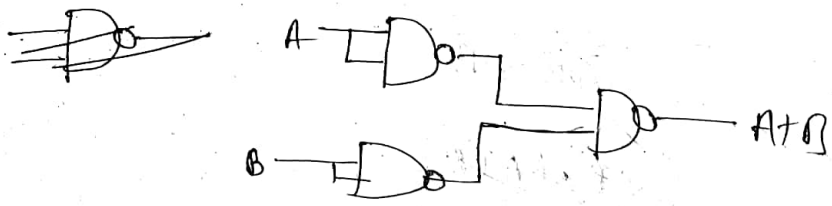
☆ NAND as Universal Gate

NAND as NOT

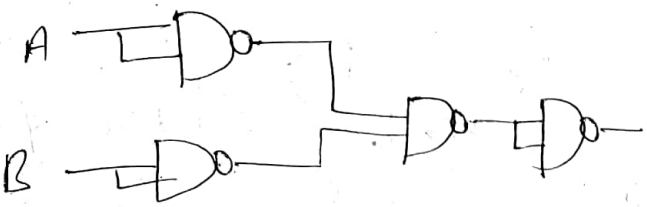


NAND as OR $\quad \overline{A\cdot B}$ to $A+B$

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

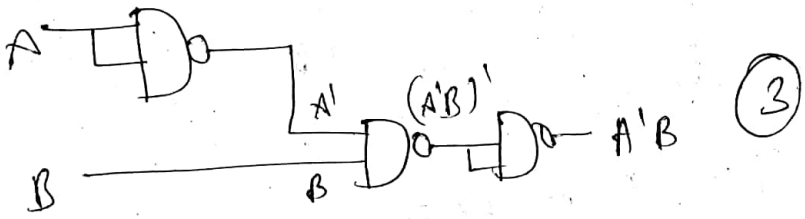


NAND as ~~EXOR~~ NOR

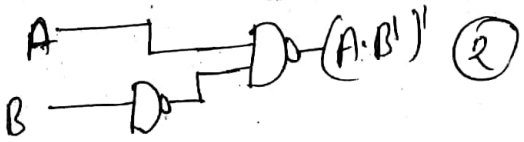


4 NAND gate

Q Find minimum no. of two i/p NAND gates required to implement $Y = A'.B$



Q $\boxed{Y = A'+B} = \overline{A''B'} = \overline{A\cdot B'}$



NAND OR EXOR

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

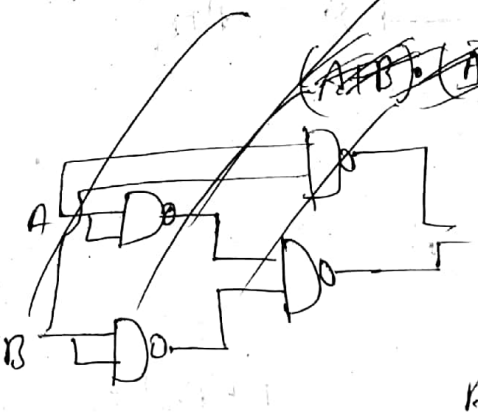
$Y = A\bar{B} + \bar{A}B$

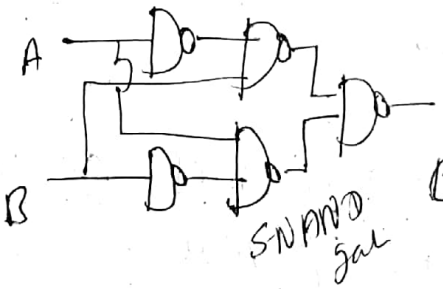
$\overline{A\bar{B} \cdot \bar{A}B} = \overline{(\bar{A}+B)\cdot(A+\bar{B})}$

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

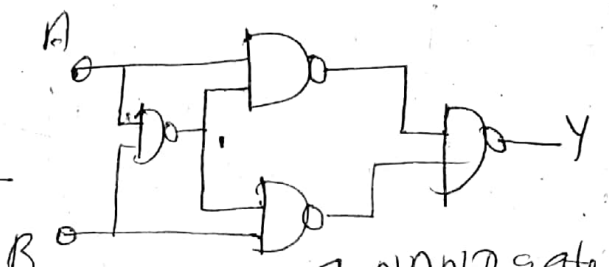
$= \overline{\bar{A}\bar{B}} \cdot \overline{AB}$

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





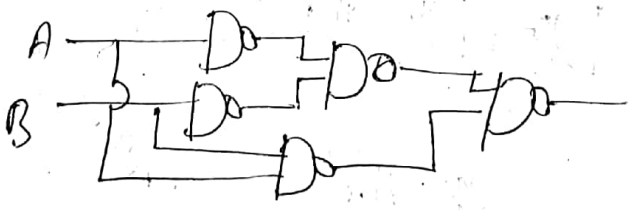
5 NAND gate



4 NAND gate

NAND OR EX NOR

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



XOR → NOR gate

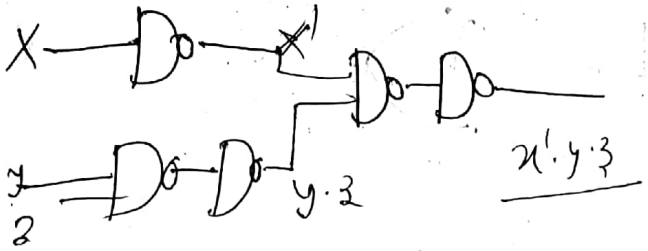
5 NAND gate

Q Minimum no. of two i/p NAND gate require to implement $\bar{x} \cdot y \cdot \bar{z}$

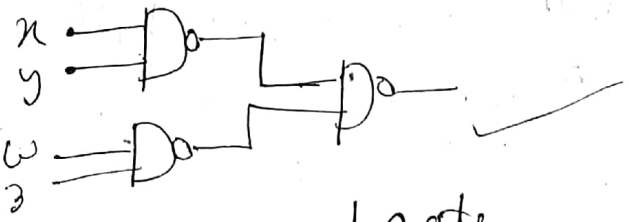
$$\overline{\overline{\bar{x} \cdot y \cdot \bar{z}}} = \overline{\overline{x + y \cdot \bar{z}}}$$

$$x \cdot \overline{y\bar{z}}$$

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

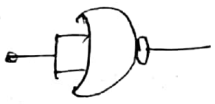


Q $\overline{\overline{xy + wz}} = \overline{\overline{xy} \cdot \overline{wz}}$

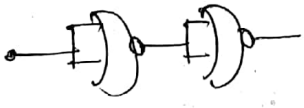


★ NOR as Universal gate

(i) NOT —



(ii) OR —

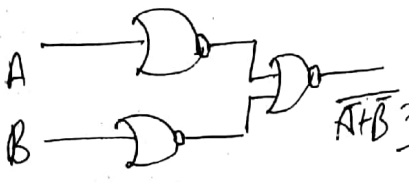


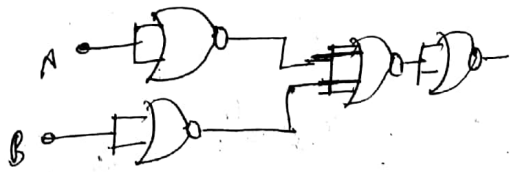
(iii) AND $\quad \overline{A+B} = \bar{A} \cdot \bar{B}$

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

(iv) NAND $\quad \overline{A+\bar{B}} = \overline{\bar{A}B}$

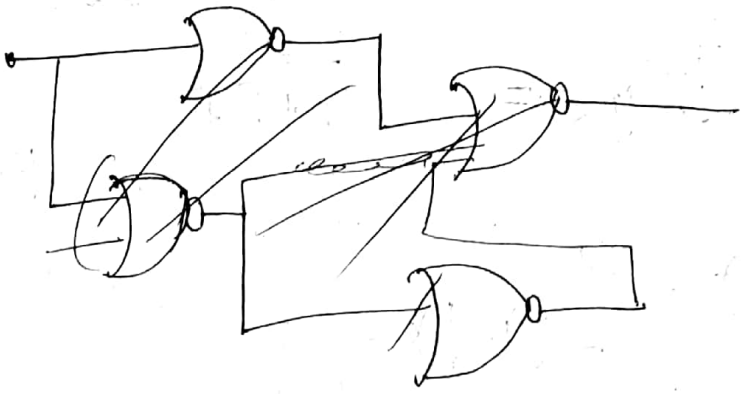
$$\overline{AB} = \bar{A} + \bar{B}$$

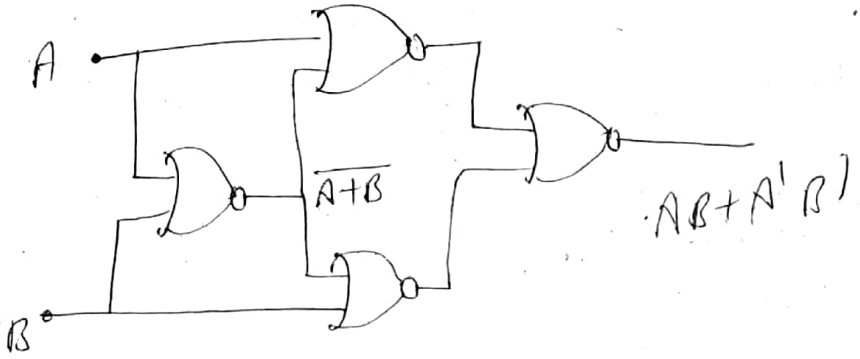




(4) $A \odot A = 0$

(5) X-NOR Using NOR



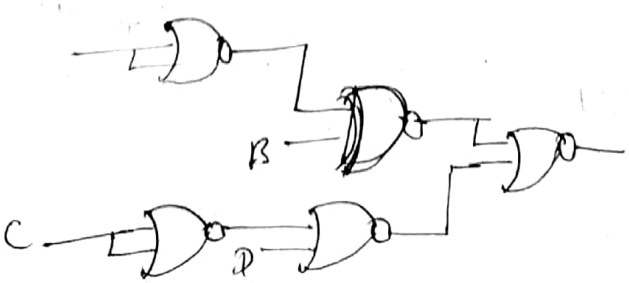


(6) XOR ⇒ XNOR — XOR

⊖ No. of two input NOR gate to implement

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

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



☆ K'MAP (Karnaugh Map)

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

0 + 01

0

0 + 0+

	A	B	C		BC/00
M_0	0	0	0	0	m_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	

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

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

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

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

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

☆ k'map example

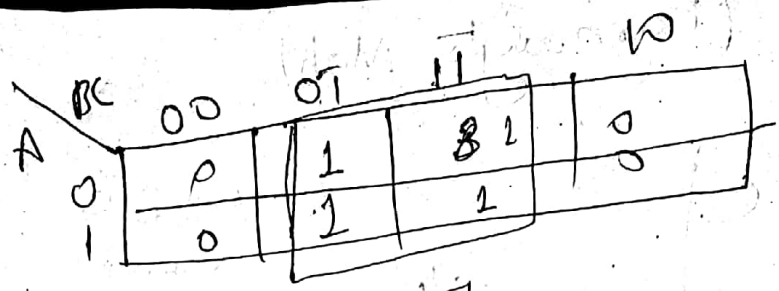
(i) $f(A,B,C) = \Sigma m(1,3,5,7)$

MSB, LSB

SOP

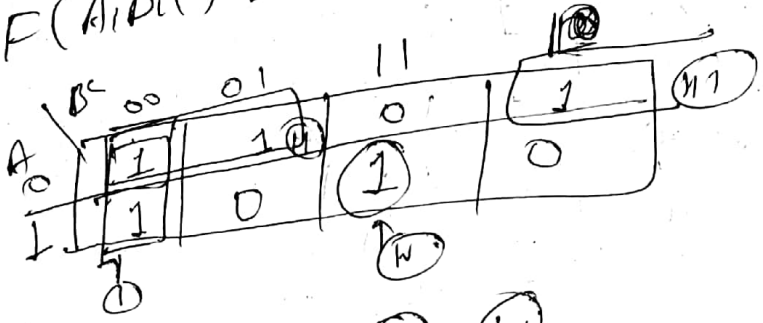
① Find out no. of cell in k'map.

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



$F = I = C$

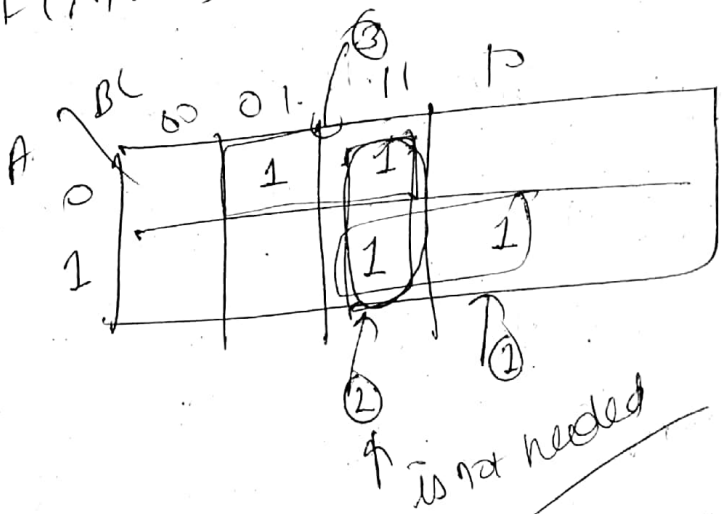
(11) $F(A,B,C) = \Sigma m(0,1,2,4,7)$



$F = (I) + (II) + (III) + (IV)$

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

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



(2) is not needed

$F = (1) + (III)$ [(II) is redundant]

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

(*) The result is minimum but may not be same or unique.

☆ Implicants :-

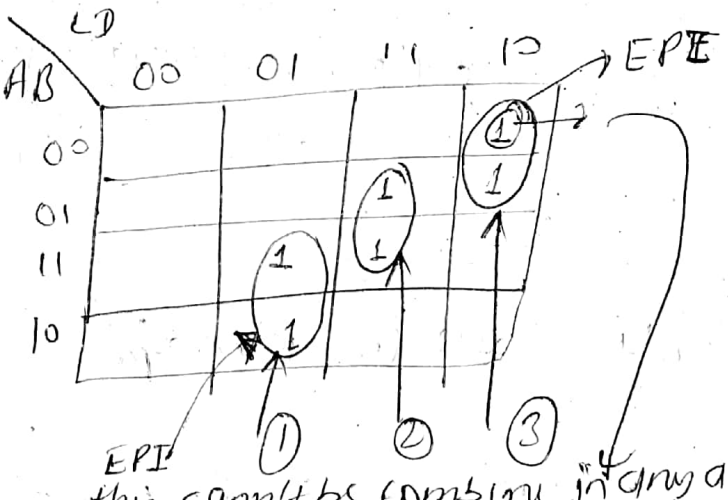
The group of 1's is called a implicants

e.g. — 1, 2, 4, 8, 16 --- etc.

prime Implicants :- It is the largest possible group of 1

Essential Prime Implicants :- atleast there is single 1 which can't be combined in any other way.

e.g

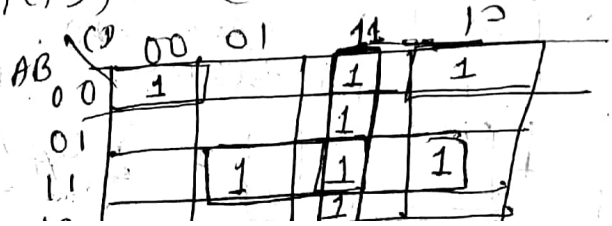


this can't be combine in any another way

it is essential prime implicant

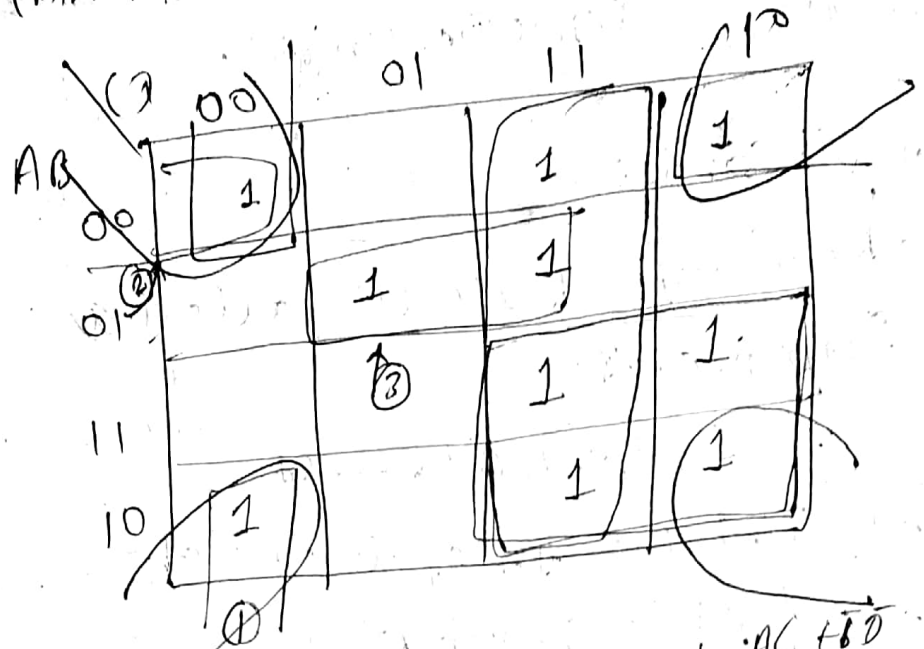
☆ K' Map with 4 variable

① $f(A, B, C, D) = \Sigma m(0, 2, 3, 7, 11, 13, 14, 15)$

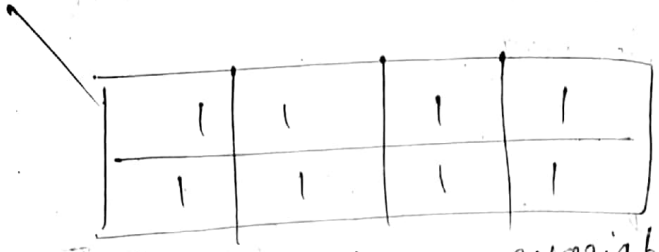


$CD + ABD + ABC + \bar{A}\bar{B}\bar{D}$

$F(A,B,C,D) = \Sigma m(0,2,3,5,7,8,10,11,13,15)$



$\cancel{\bar{A}\bar{B}\bar{C}\bar{D}} + A\bar{B}\bar{D} + \cancel{\bar{A}BD} + CD + \bar{A}C + \bar{B}D$



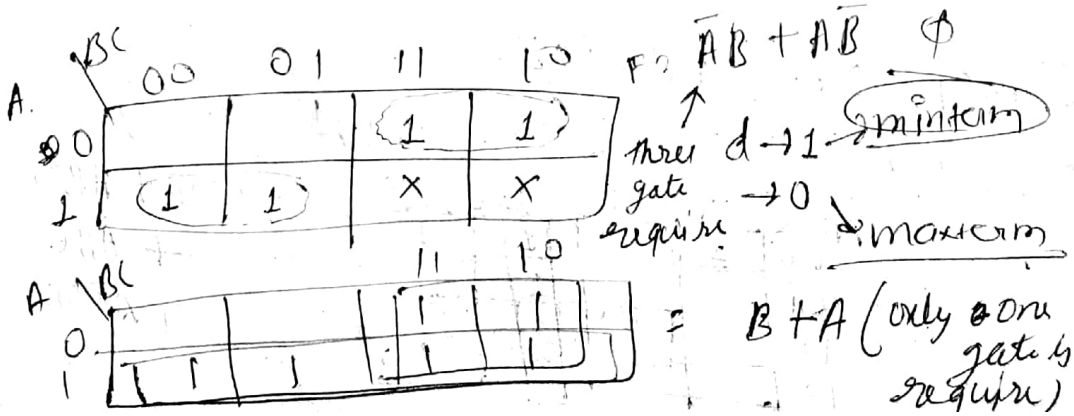
8 1's → 3 variable (reduce)

1 1 1

$= 1$

☆ Don't care in K-map.

e.g. $F(A,B,C) = \Sigma m(2,3,4,5) + \Sigma d(6,7)$



$F = \bar{A}B + A\bar{B}$ ↑ three gate require

d → 1 → minterm

→ 0 → maxterm

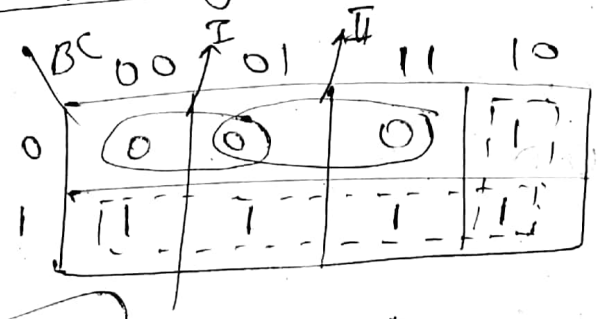
$= B + A$ (only one gate is require)

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

	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	1
m_4	1	0	0	1
m_5	1	0	1	1
m_6	1	1	0	*
m_7	1	1	1	*

X-OR is odd 1's detecta (no. of 1 is odd)

☆ K' map using Max term



$F = A + BC'$

Sometime it will be very helpfull to combine only

*$\bar{F}$ = I + II

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

Demorgan law

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

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

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

(POS)

$F = A + BC'$

$F(A,B,C,D) = \Sigma m(1,3,5,9,11,14,15)$ [7M]

C mqdes.

	00	01	11	10
00	0	1	1	0
01	0	1	0	0
11	0	0	1	1
10	0	1	1	0

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

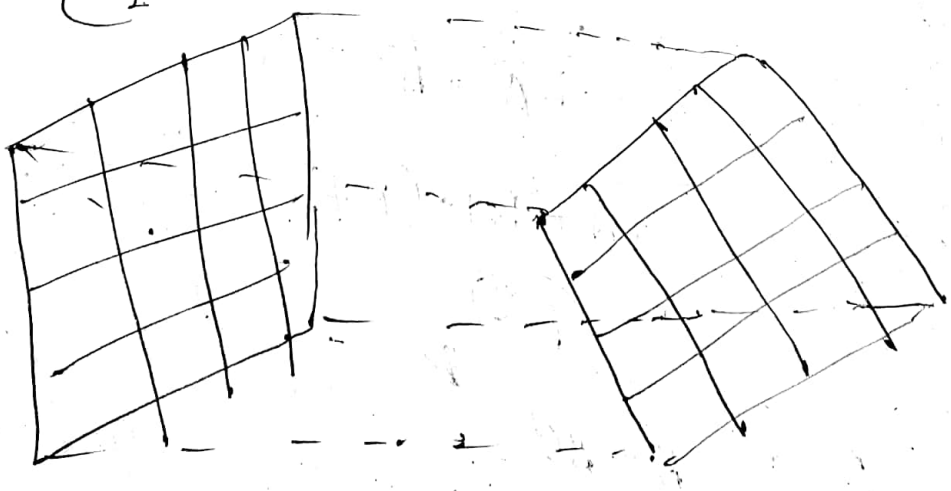
★ K' map using 5 variable.

$F(A,B,C,D,E)$

A B C D E

16 { 0 0 0 0 0
0
⋮
0 1 1 1 1

16 { 0 0 0 0 0
1
⋮
1



★ Quine -

e.g. Y(

0-000
1-000
2-001
3-001
7-011
8-10
9-100
10-101
15-111

☆ Quine - Mc Cluskey minimization Technique (Tabular method)

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

0-0000
1-0001
2-0010
3-0011
7-0111
8-1000
9-1001
10-1011
15-1111

Step 1

Group	minterm	Bin Rep A B C D
0	m_0	0 0 0 0 ✓
	m_1	0 0 0 1 ✓
	m_8	1 0 0 0 ✓
2	m_3	0 0 1 1 ✓
	m_9	1 0 0 1 ✓
3	m_7	0 1 1 1 ✓
	m_{11}	1 0 1 1 ✓
4	m_{15}	1 1 1 1

m_0 & m_1 → one variable is changing hence matched

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
		1 – 1 1

Step 3

Group	M.P	Binary A B C D	
0	~~$m_0 = m_8 - m_8 - m_9$~~		
	$m_0 - m_8 - m_1 - m_9$	1 - 00 -	$\bar{B}\bar{C}$
		- 00 -	
1	$m_1 - m_3 - m_9 - 11$	- 0 - 1	$\bar{B}D$
	$m_1 - m_9 - m_3 - m_{11}$	- 0 - 1	
2	$m_3 - m_7 - m_{11} - m_{15}$	- - 11	CD
	$m_3 - m_{11} - m_7 - m_{15}$	- - 11	

Check the PIV table any M.P is Uncheck, then we should consider it also

P.I	Min-term involve	0	1	3	7	8	9	11	15
$\bar{B}\bar{C}$	0, 1, 8, 9	⊗	×			⊗	×		
$\bar{B}D$	1, 3, 9, 11		×	×			×	×	
CD	3, 7, 11, 15			×	⊗			×	⊗

Circle the one cross

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

☆ LED (seven segment decoder)

b_3 b_2 b_1 b_0	a	b	c	d	e	f	g
0 0 0 0 (0)	1	1	1	1	1	1	0
0 0 0 1 (1)	0	1	1	0	0	0	0
0 0 1 0 (2)	1	1	0	1	1	0	1
0 0 1 1 (3)	1	1	1	1	0	0	1
0 1 0 0 (4)	0	1	1	0	0	1	1
0 1 0 1 (5)	1	0	1	1	0	1	1
0 1 1 0 (6)	1	0	1	1	1	1	1
0 1 1 1 (7)	1	1	1	0	0	0	0
1 0 0 0 (8)	1	1	1	1	1	1	1
1 0 0 1 (9)	1	1	1	1	0	1	1
1 0 1 0	} → don't care						
⋮ ⋮							
1 1 1 1							

for a

b_3b_2 \ b_1b_0	00	01	11	10
00	1		1	1
01		1	1	1
11	X	X	X	X
10	1	1	X	X

$$a = b_3 + b_1 + \bar{b_2}\bar{b_0} + b_3\bar{b_2}\bar{b_0} + b_0b_2$$

$$b_3 + b_1 + b_0 \odot b_2$$

$$b = b_3 + b_1b_0 + \bar{b_2} + b_1\bar{b_0}$$

similarly, c, d, e, f, g ...

☆ Combinational circuit & Sequential circuit

Combinational circuit

- output is only depend on present i/p

ex. Add

1
+0
1

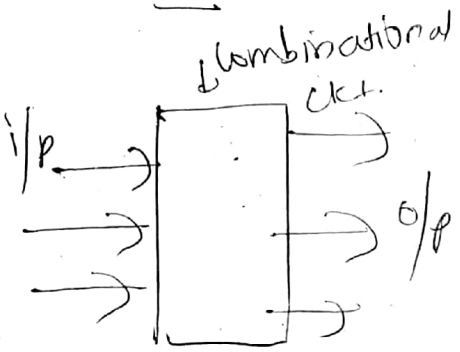
Sequential

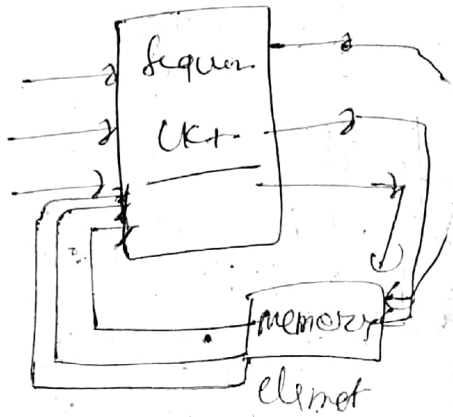
- o/p depends on the present i/p as well as previous output

ex. Counter

+1 increment 1 to prv. output

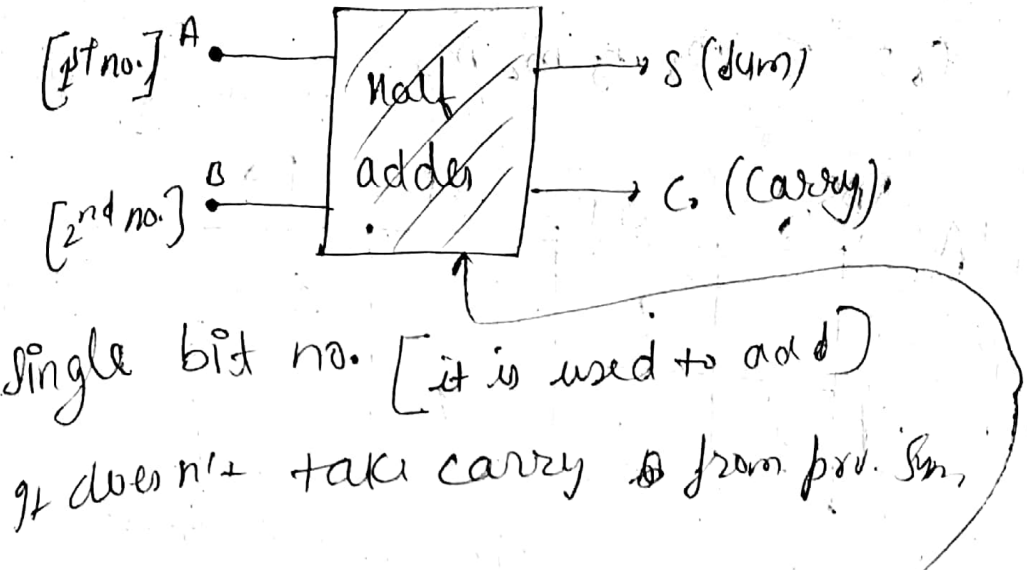
4+1 = 5





① feedback

✡ Half adder [Comb^{n} ckt]

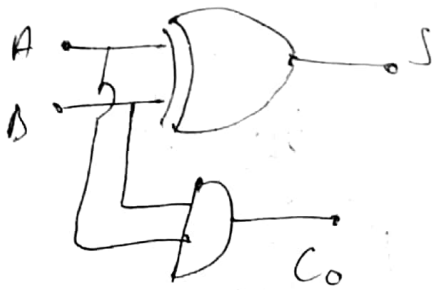


* Single bit no. [it is used to add]
* It doesn't take carry from prv. Sm.

A	B	S	C$_o$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

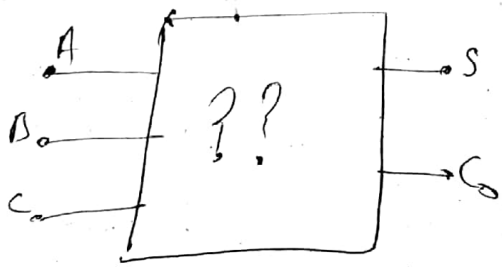
S: A ⊕ B

C$_o$: AB



✡ Full adder

A	B	C$_i$	S	C$_o$
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



$S = \Sigma(m_1, m_2, m_4, m_7)$

$C_0 = \Sigma\, m_3\, m_5\, m_6\, m_7$

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

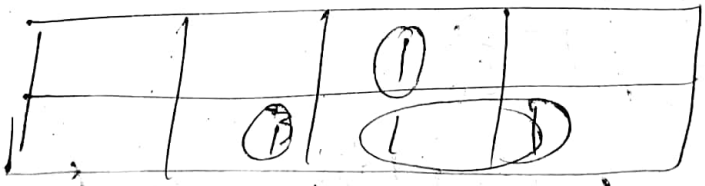
Check board combⁿ

$S = A \oplus B \oplus C_i$

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

$C_0 = BC_i + AC_i + AB$

$C_0 = AB + C_i(A \oplus B)$ (this can be obtained by taking diff. mapping)



↑ take this mapping

☆ 4-bit even parity generator

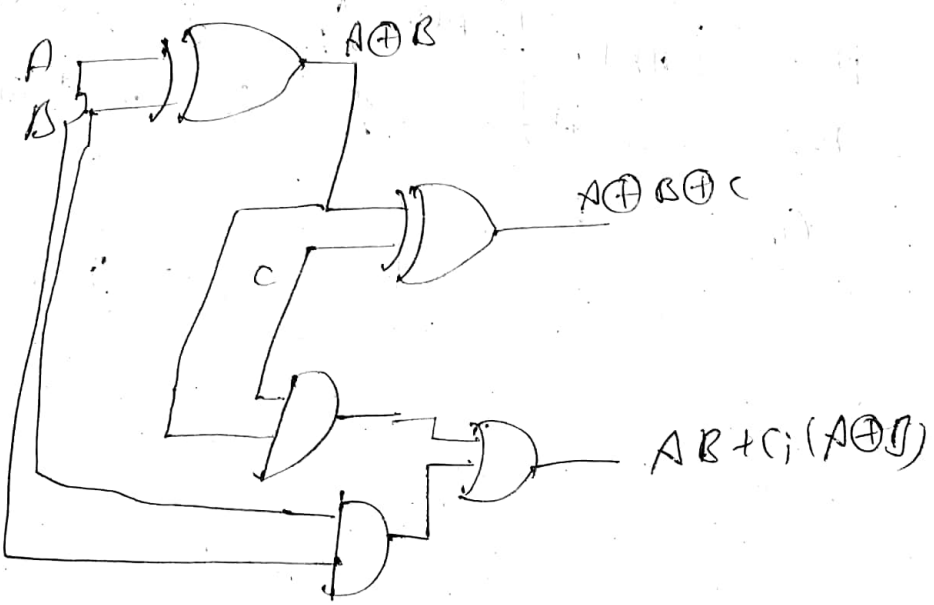
Check-board conf.

Note

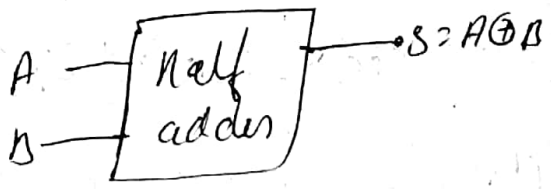
$$P_0 = \bar{b}_3 \bar{b}_2 \bar{b}_1 b_0 + \bar{b}_3 \bar{b}_2 b_1 \bar{b}_0 + \bar{b}_3 b_2 \bar{b}_1 \bar{b}_0 + \bar{b}_3 b_2 b_1 b_0 + b_3 \bar{b}_2 \bar{b}_1 b_0 + b_3 \bar{b}_2 b_1 \bar{b}_0 + b_3 b_2 \bar{b}_1 \bar{b}_0 + b_3 b_2 b_1 b_0 + \text{⊕} \cdots$$

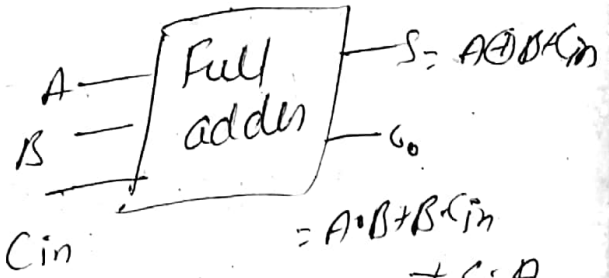
$$P_0 = b_0 \oplus b_1 \oplus b_2 \oplus b_3$$

Circuit



★Full adder using half adder

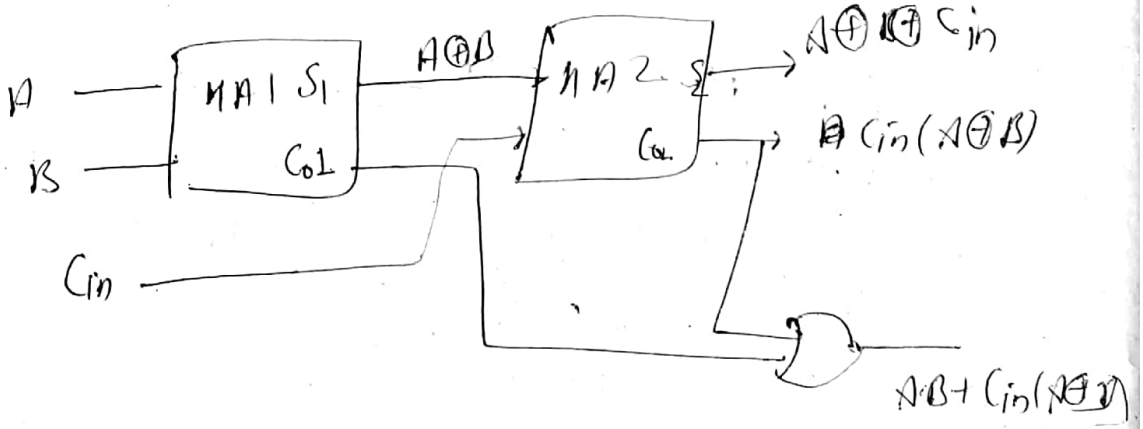




$= A \cdot B + B \cdot C_{in} + C_{in} A$

or

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



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

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

$A \cdot B + C_{in}A\bar{B} + C_{in}\bar{A}B$

$A \cdot B + A\bar{B}C_{in} + \bar{A}BC_{in}$

$A(B + \bar{B}C_{in}) + \bar{A}BC_{in}$

$A(B + C_{in}) + A'BC_{in}$

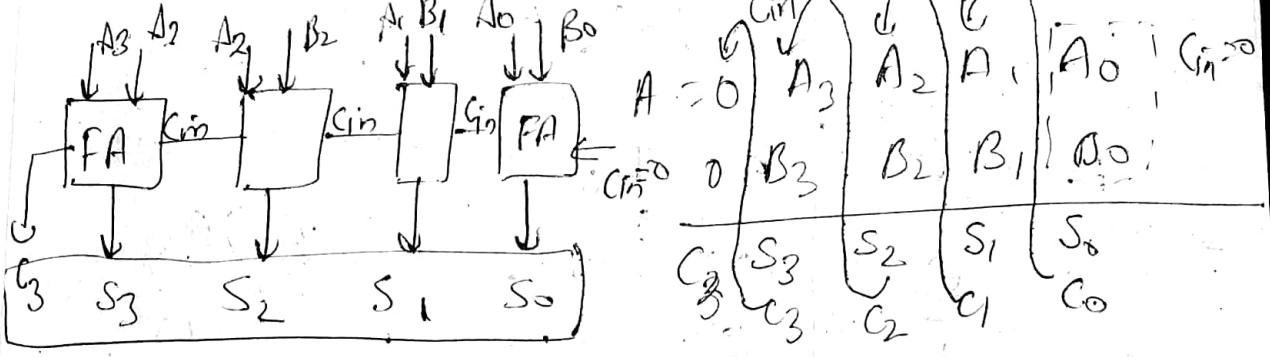
$AB + AC_{in} + A'BC_{in}$

$AB + C_{in}(A + A'B)$

$AB + C_{in}(A + B)$

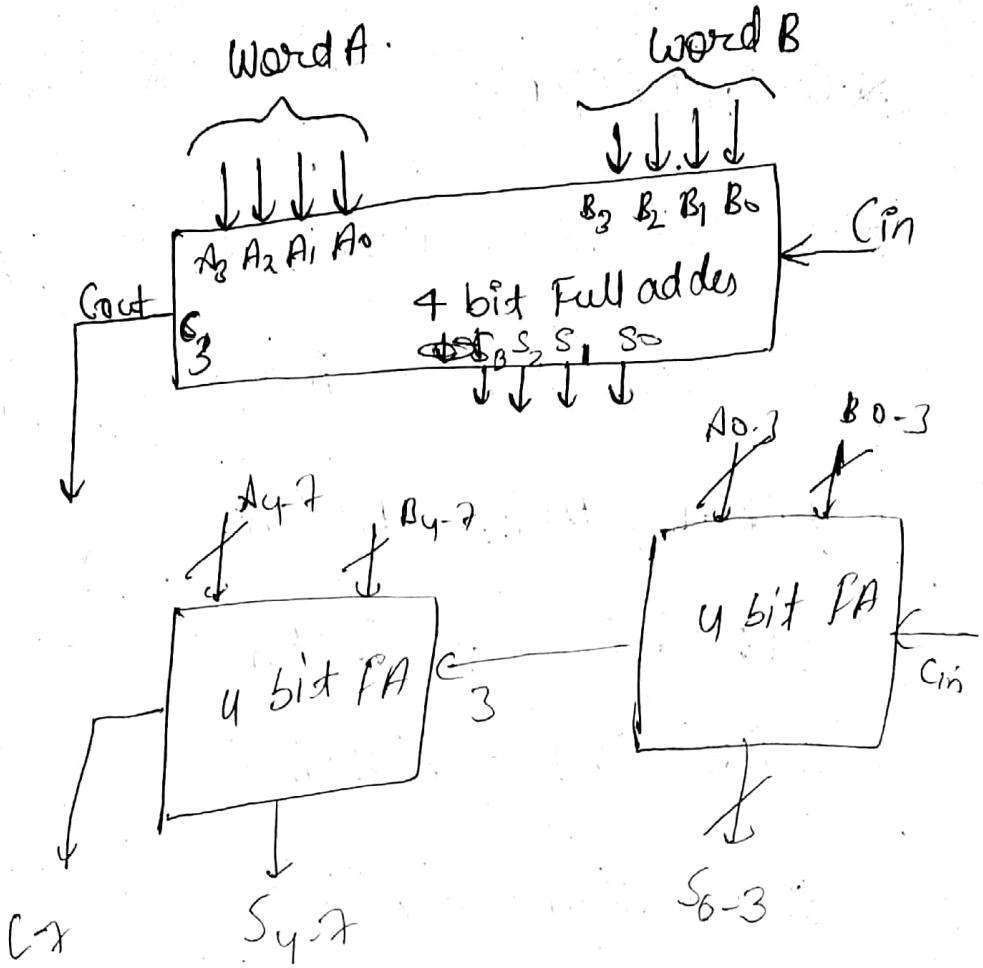
$AB + ABC_{in} + AC_{in}$

★ 4 bit parallel adder using full adder

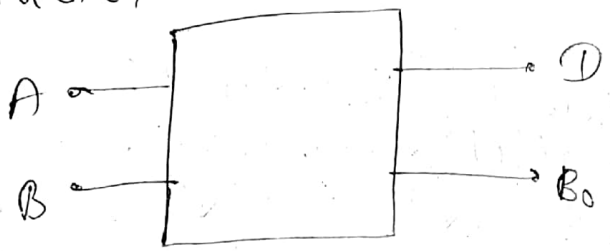


find sum

74H(283) => IC for 4 bit full adder



☆ Half subtractor

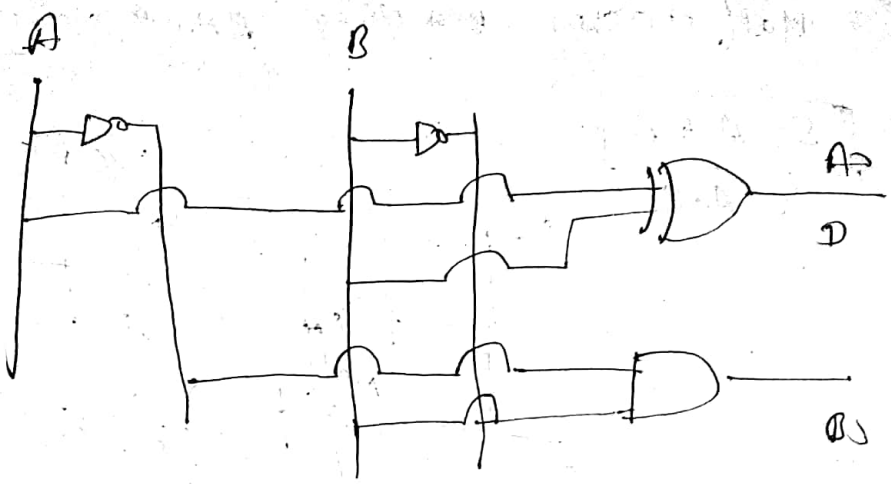


Half subtractor

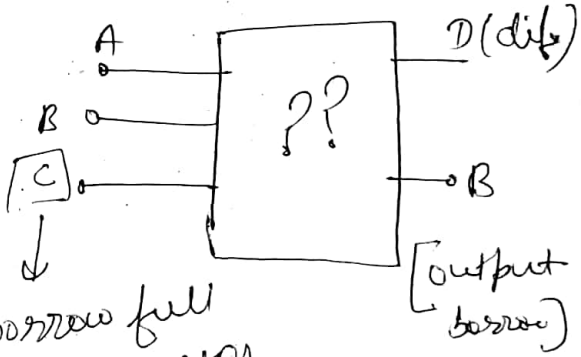
A	B	D	B_0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$D = A \oplus B$ (odd)

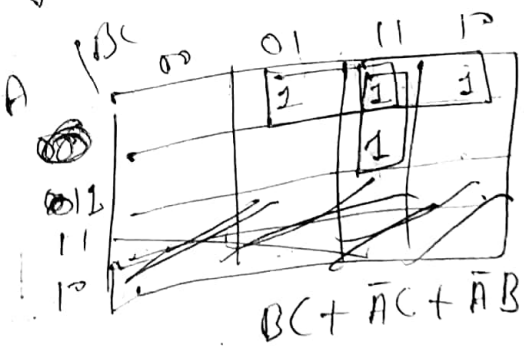
$B_0 = \bar{A}B$



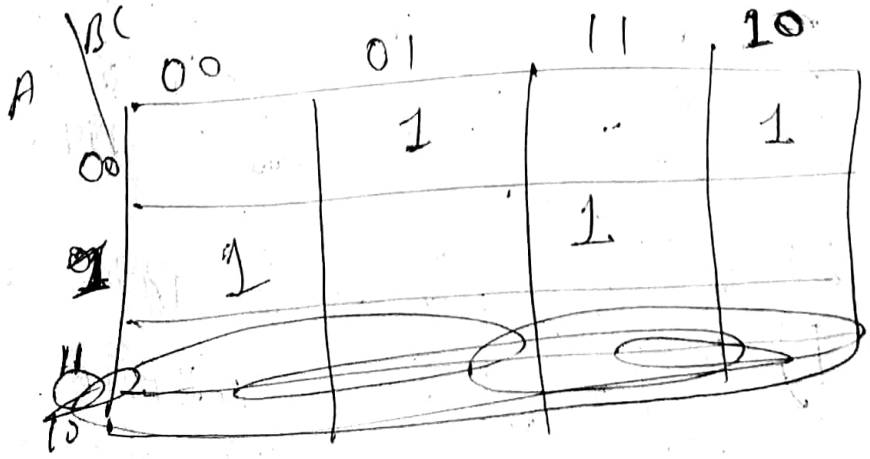
☆ Full Subtractor



A	B	C	D	B_0
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



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



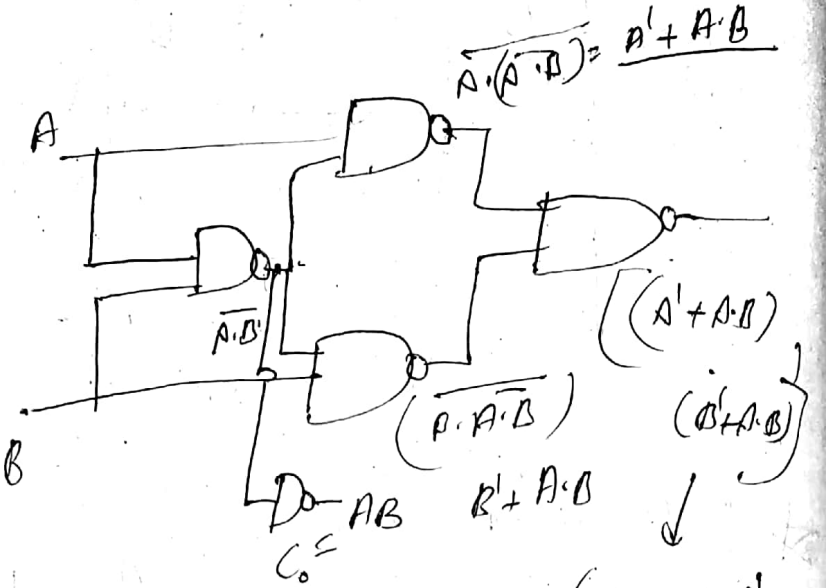
$A \oplus B \oplus C$

Checkboard Conf.

Half adder Using NAND Gate Only

$S = A \oplus B$

$C_0 = A \cdot B$



$(A'B' + AB)$

↓

$\overline{A \odot B}$

↓

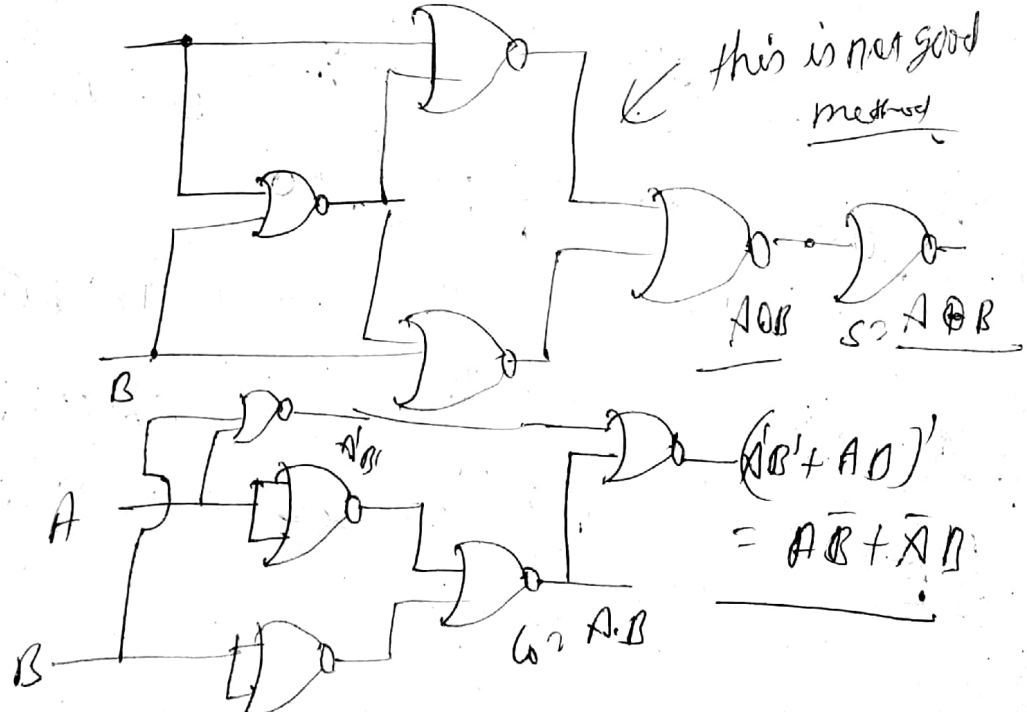
$A \oplus B$

Half adder Using NOR gate only

$S = A \oplus B$

$C_0 = A \cdot B$

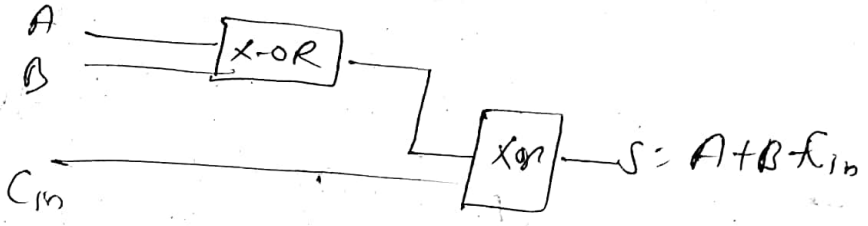
$A \oplus B \longrightarrow$

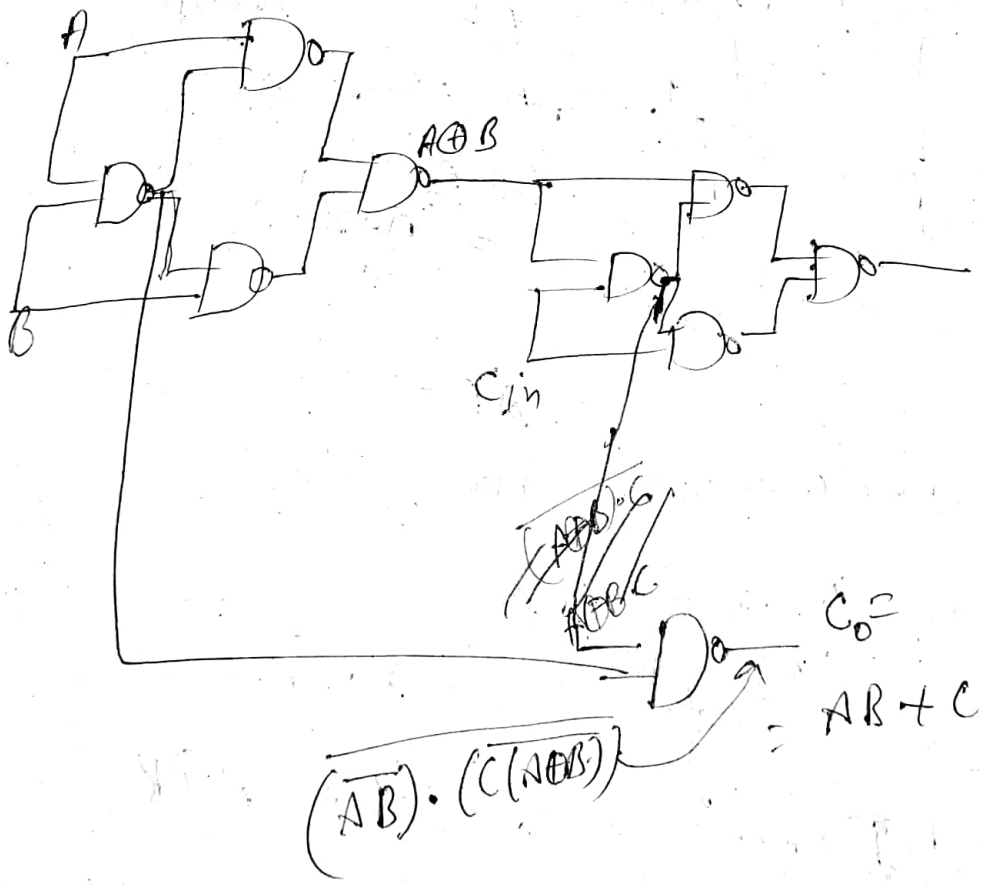


☆ Full adder using NAND gate

$S = A \oplus B \oplus C$

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

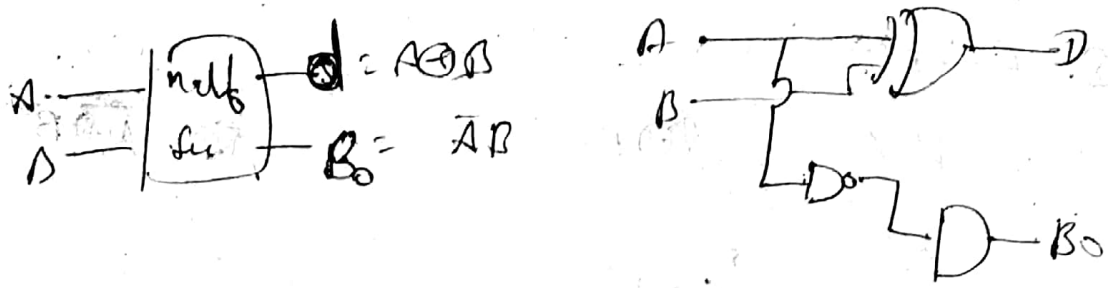




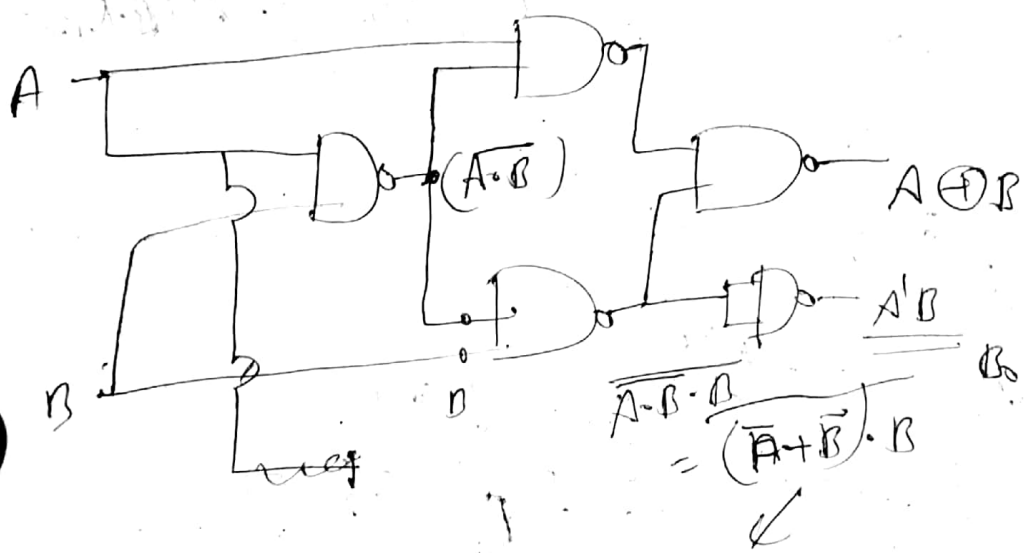
$\overline{(AB)} \cdot \overline{(C(A \oplus B))}$

$C_o = AB + C(A \oplus B)$

☆ Half Subtractor using NAND gate only

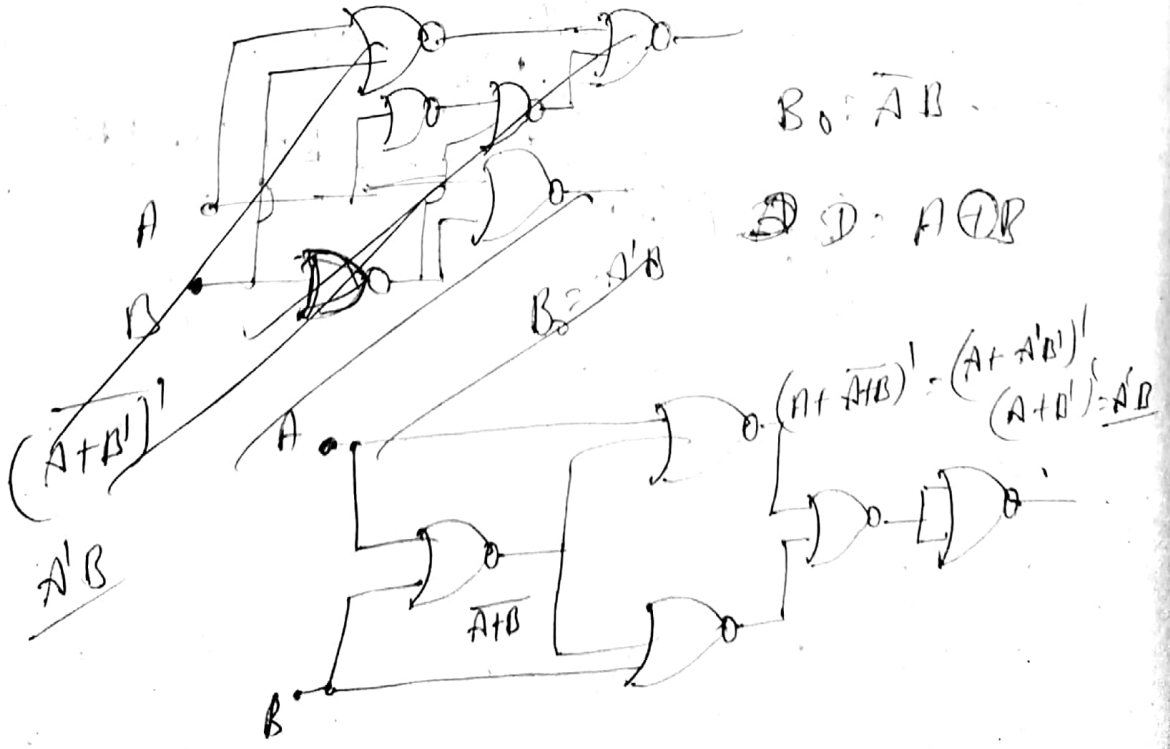


A⊕B



$\overline{A'B} =$

Half Subtractor using NOR gate

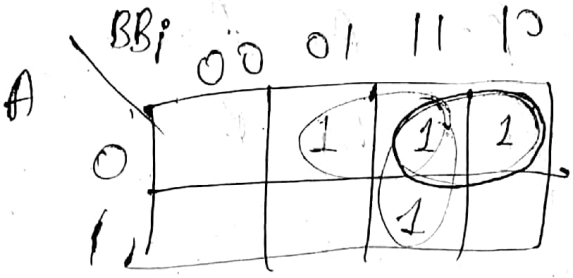


✱ Full subtractor using NAND gates only

A (Minuend)	B (Subtrahend)	B_i (Borrow input)	D	B_o
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

D depends only on present i/p

$D = A - B - B_i$

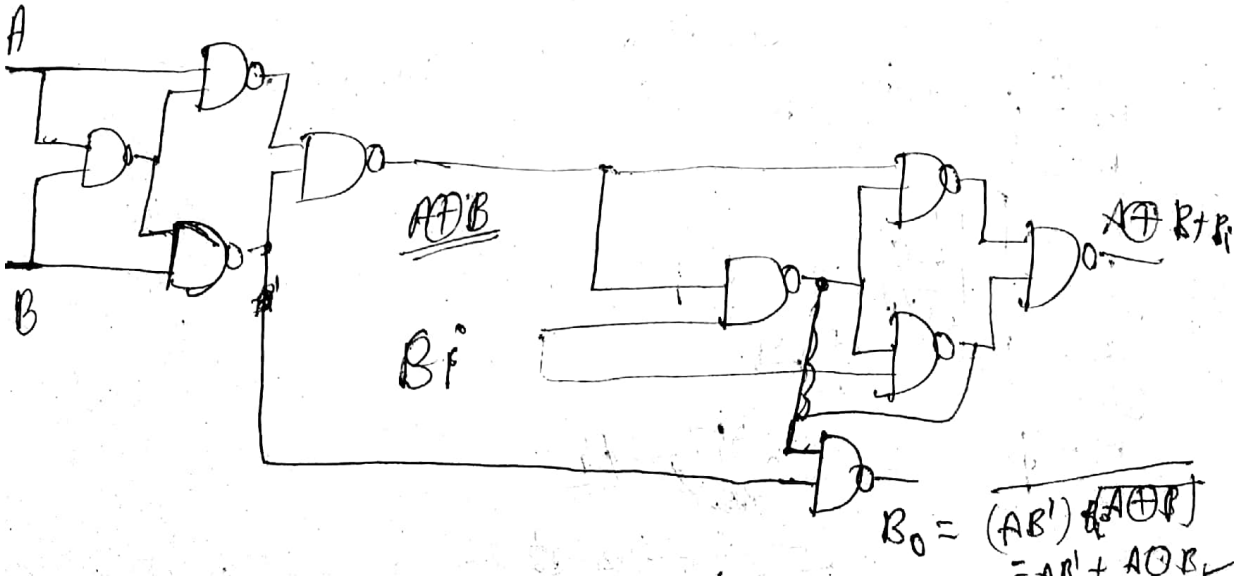


$A\bar{B} + \bar{A}B_i + \bar{A}B$

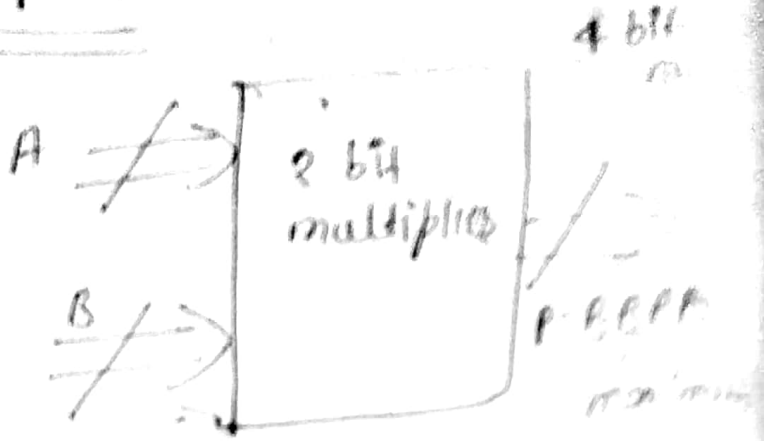
~~$B\bar{B}_i + \bar{A}B_i B + \bar{A}B_i$~~

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

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

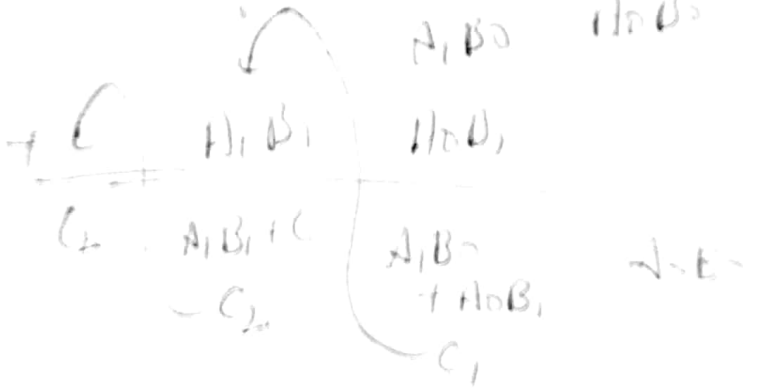


☆ 2-bit multiplier

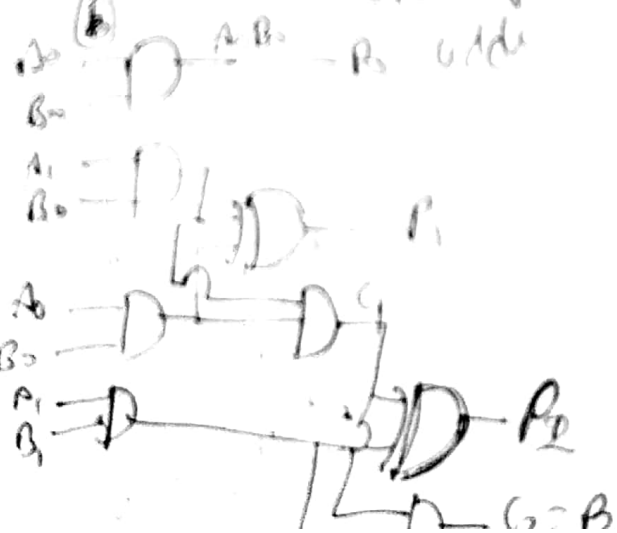


$A = A_1 \; A_0$

$B = B_1 \; B_0$



$P_0 = A_0 B_0$



A: 1 1 (3)

B: 1 0 (2)

0 0

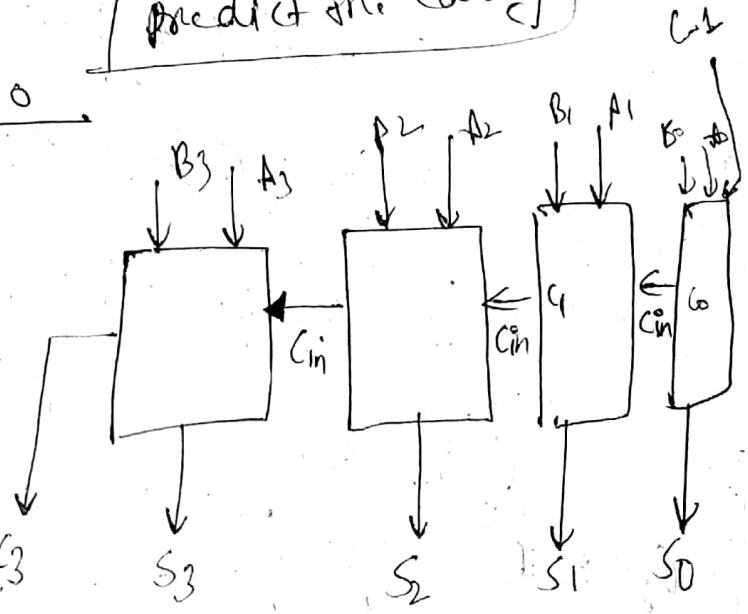
1 1

1 1 0

☆ Carry look Ahead Adder (CLA adder)

Predict the carry

A	B	C_i	C_o
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



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

Carry generation — Carry propogator

$C_o = G_1 + P C_{in}$

G Pi

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

i=0

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

i=1

$C_1 = G_1 + P_1 C_0$

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

$C_1 = G_1 + P_1 G_1 + P_1 P_0 C_{-1}$

i=2

$C_2 = G_2 + P_2 C_1$

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

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

$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_0 G_0$

$+ P_3 P_2 P_1 P_0 C_{-1}$

$G_3 = A_3 \cdot B_3$

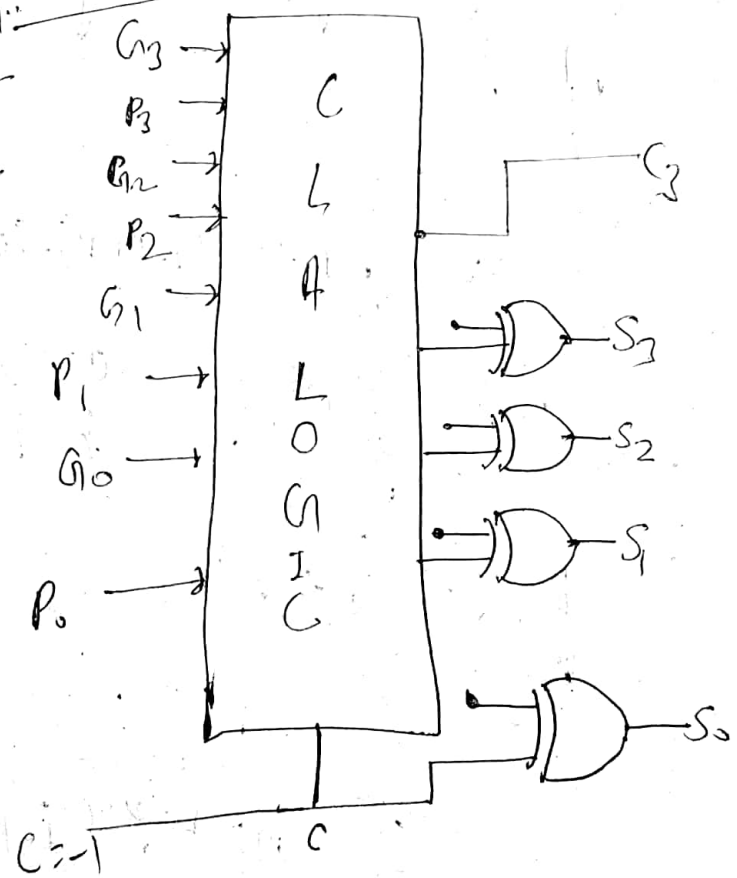
$G_2 = A_2 \cdot B_2$

$P_3 = A_3 \oplus A_2$

$P_2 = A_2 \oplus A_1$

$P_1 = A_1 \oplus B_1$

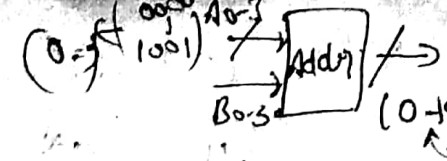
$P_0 = A_0 \oplus B_0$



Decimal

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

☆ BCD adder

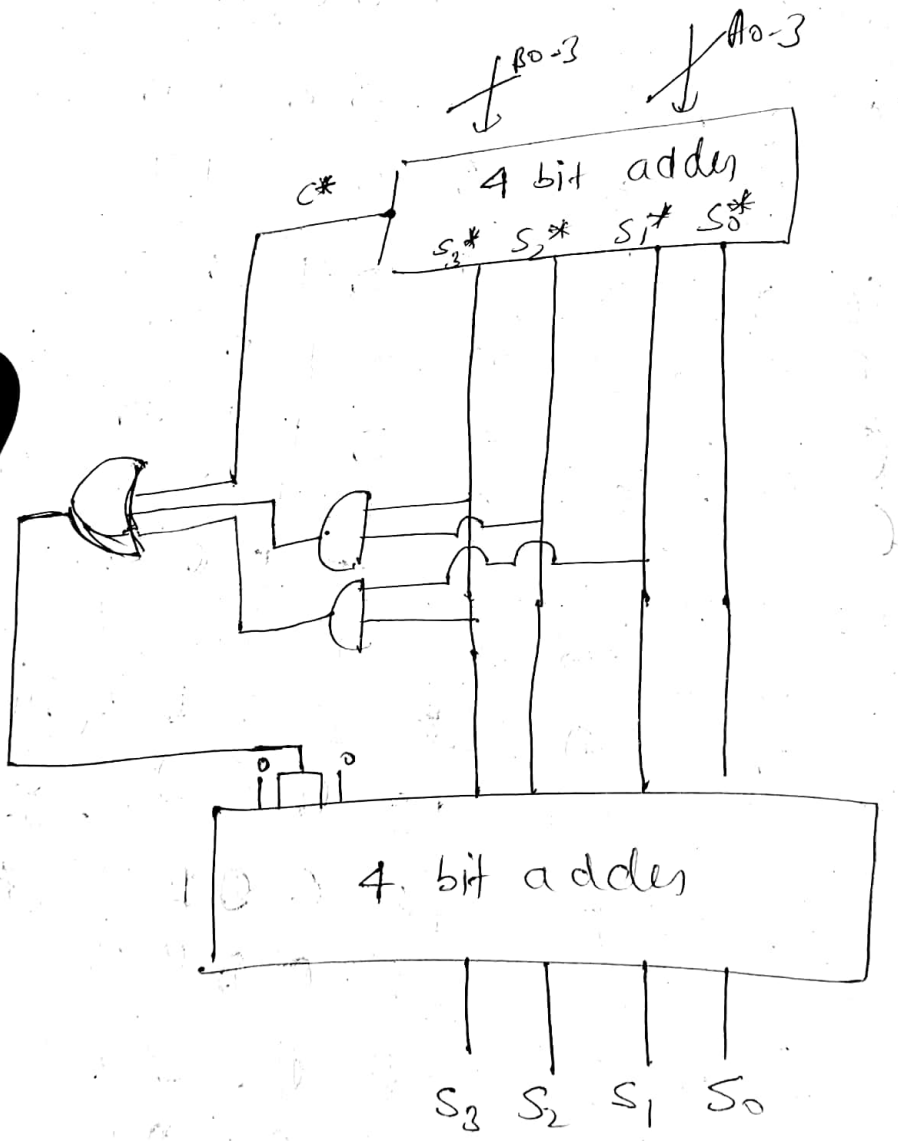


Decimal	Binary Sum					BCD Sum					
	C	S_3^*	S_2^*	S_1^*	S_0^*	C	S_3	S_2	S_1	S_0	
0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	1	0	0	0	0	1	
2	0	0	0	1	0	0	0	0	1	0	
3	0	0	0	1	1	0	0	0	1	1	
4	0	0	0	1	0	0	0	1	0	0	
5	0	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0	
7	0	0	1	1	1	0	0	1	1	1	
8	0	1	0	0	0	0	1	0	0	0	
9	0	1	0	0	1	0	1	0	0	1	
10	0	1	0	1	0	1	0	0	0	0	0110
11	0	1	0	1	1	1	0	0	0	1	↓ add
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	1	0	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	

When (i) $S_3^* \cdot (S_2^* + S_1^*) = 1$

(ii) $C^* = 1$

$= C^* + S_3^* S_2^* + S_3^* \cdot S_1^* =$ (1) thus only



⭐ Mul

⇒ It is
info
and

⇒ It

I0 —
I1 —
I2 —
I3 —

Advanta

(1) Re

(2) Red

(3) Imp

Types

(1) e.g. IM

2° 1

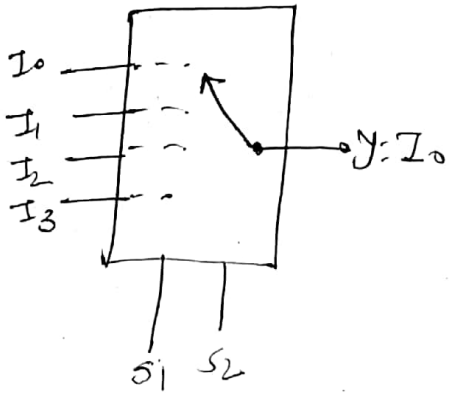
I

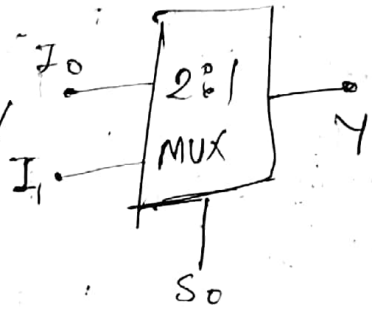
I1

☆ Multiplexers

⇒ It is combinational circuit that selects binary information from one of many input lines and direct it to o/p line

⇒ It is simply a data selector

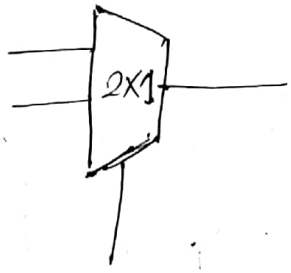




Select line / Selection value

S_1	S_0
0	0
0	1
1	0
1	1

n = no. of selection lines
m = no. of input lines
$m = 2^n$



Advantages

(1) Reduce no. of wires (as gate ↓)

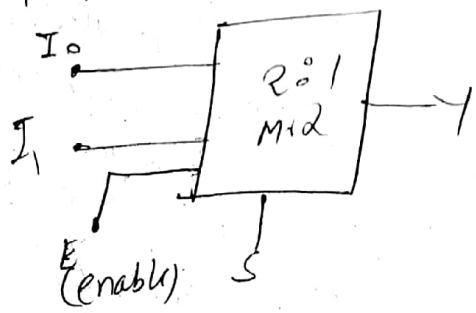
(2) Reduces circuit complexity & cost

(3) Implementation of various circuit using MUX

Types

(1) 2:1 MUX, 4:1 MUX, 8:1 MUX, 16:1 MUX & 32:1 MUX

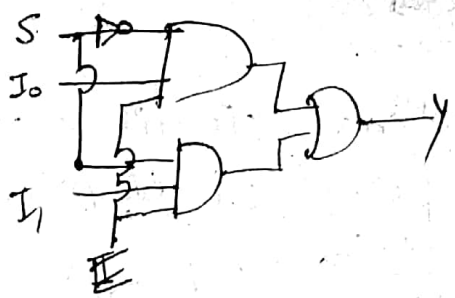
2:1 MUX:



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

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

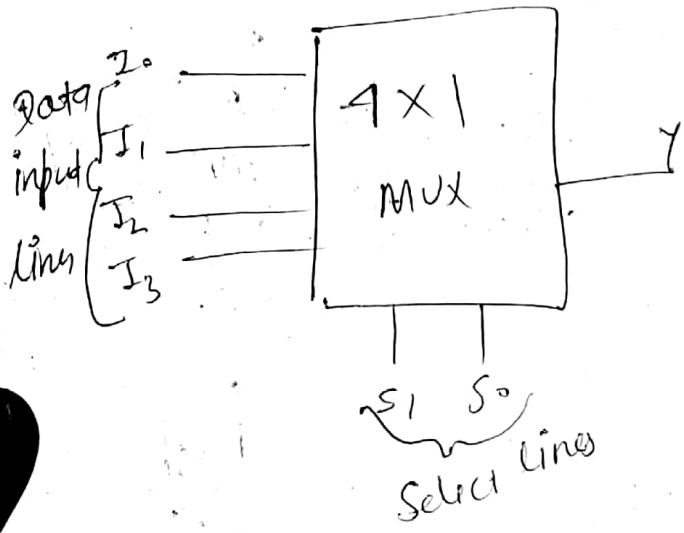
$= E\cdot(\bar{S}I_0 + S\cdot I_1)$



n = 4

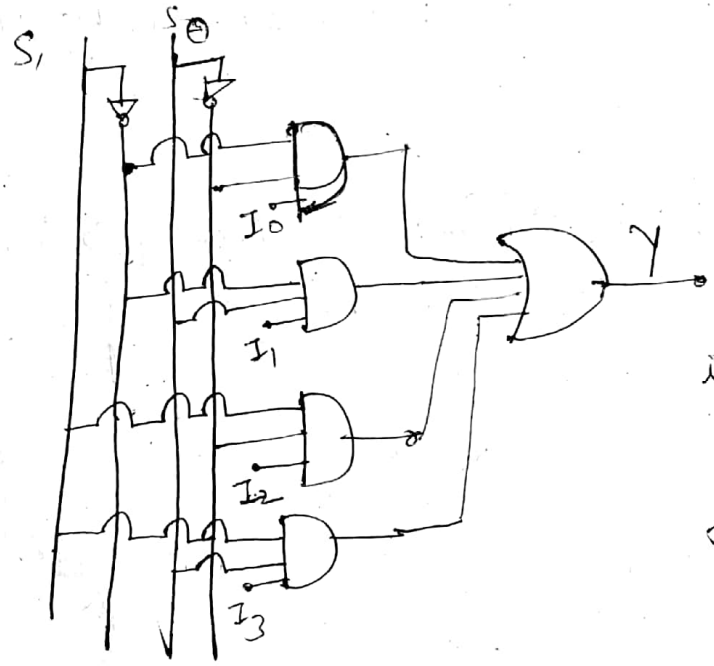
no. of selection lines $= \log_2 4 = 2$

☆ 4×1 MUX



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

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



if enable is present then take OR operator to get real Y

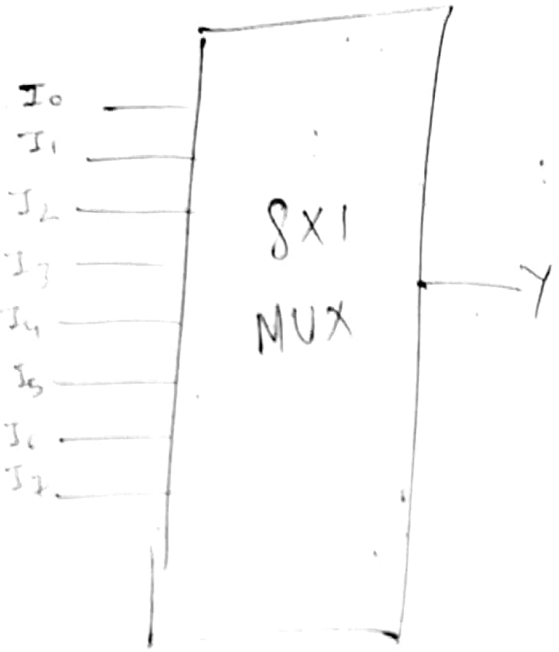
$a_1 \quad a_2 \quad a_3$

$b_1 \quad b_2 \quad b_3$

$$\frac{a_1 + a_2 + a_3}{b_1 + b_2 + b_3}$$

☆ 8 × 1 Multiplier

no. of select line $= \log_2^8 = 3$



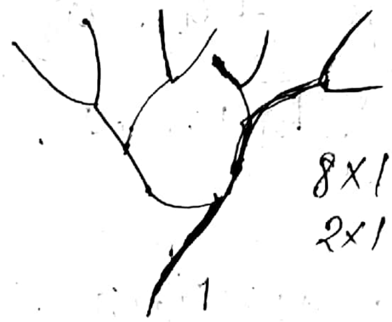
S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

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

☆ MUX tree (obtaining higher order MUX using lower order MUX)

☆ Implement 4x1 MUX using 2x1 MUX

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



8x1 MUX
2x1 MUX

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

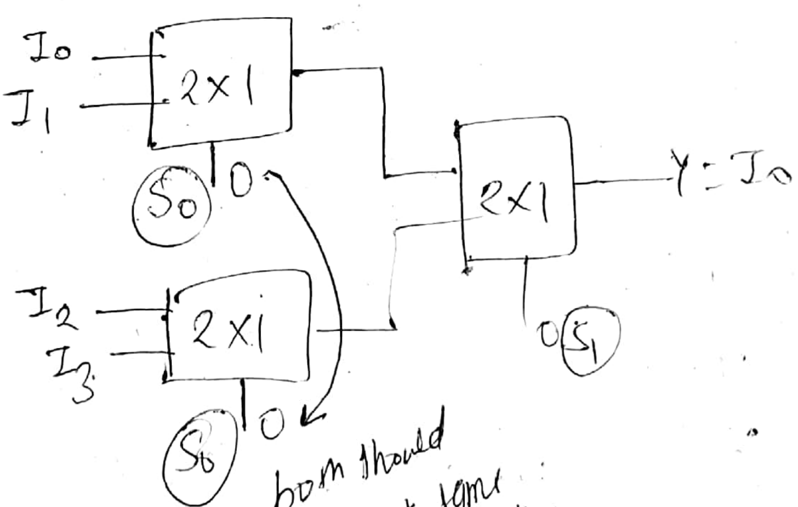
$\frac{2}{2} = 1$

2 + 1 = 3 (2x1)

mrell. 2x1 require

always sequence provide

for



both should select same select line

☆ 8×1 MUX using 2×1 MUX

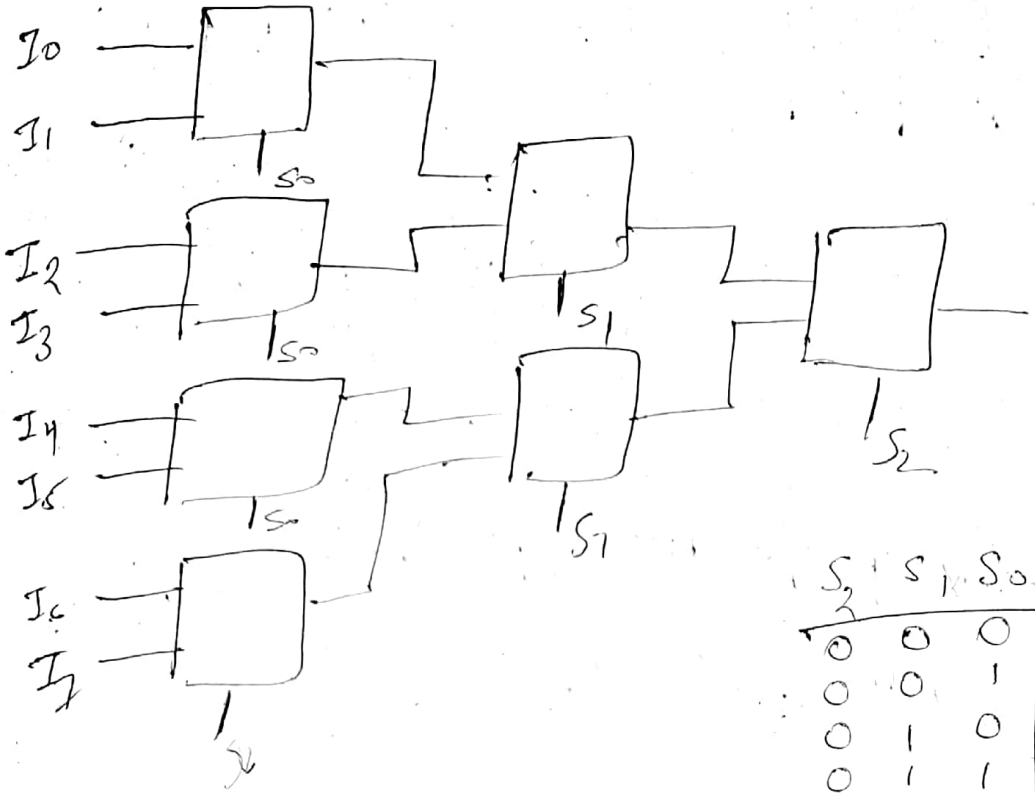
$\frac{8}{2} = 4$

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

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

$4+2+1$

$= 7$ 2×1 MUX will require

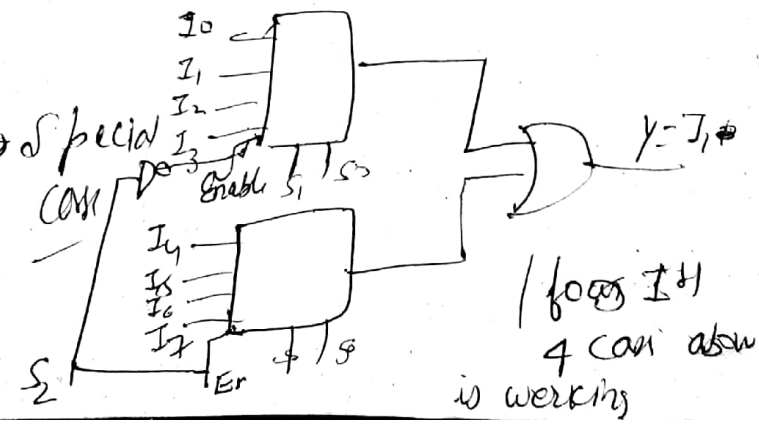


S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

☆ 8×1 MUX using 4×1 MUX (Special case)

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

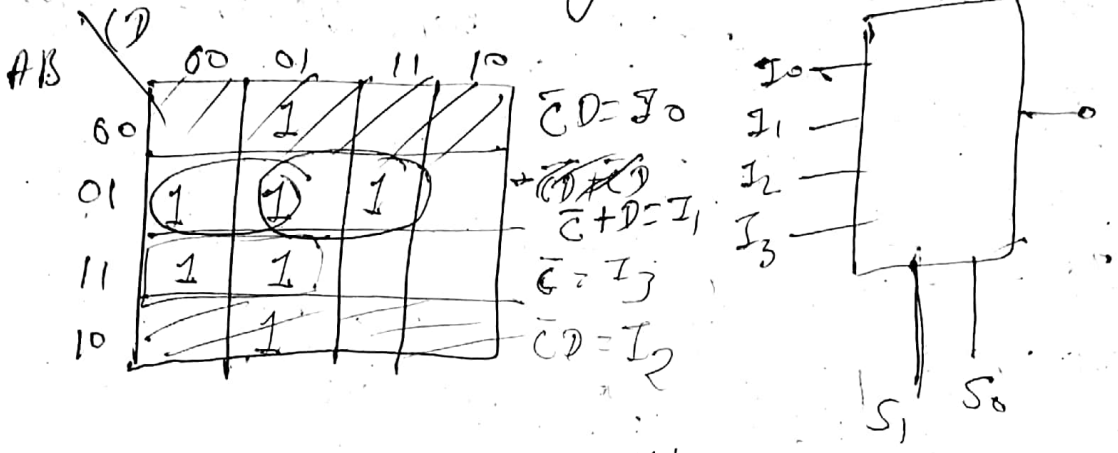
$\frac{2}{4} = 0.5$ (not possible) → Special case



for I1 4 can also is working

☆ Implementation of Boolean Function using Multiplexers

Implement :- $F(A,B,C,D) = \Sigma m(1,4,5,7,9,12,15)$ using 4×1 MUX



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

$S_0 = B$

$S_1 = A$

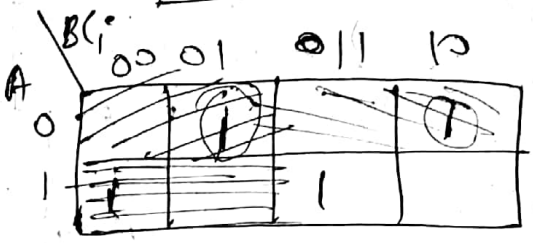
☆ 32×1 MUX using 8×1 MUX

watch lecture

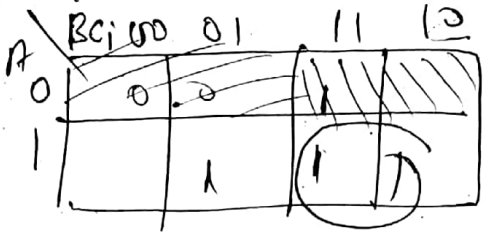
☆ 1-Bit Full adder Using Multiplexer

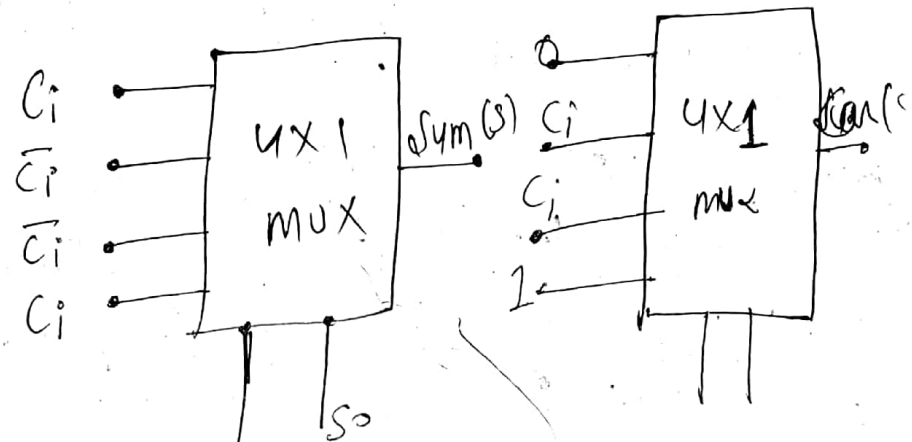
A	B	C_{in}	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

For S



For C_o





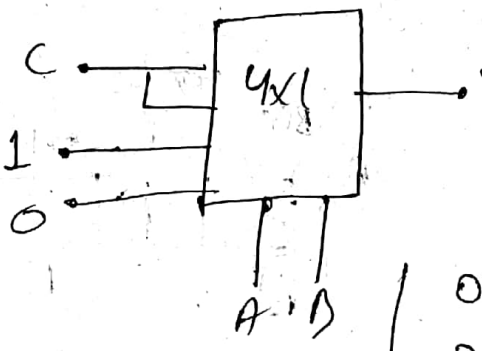
A	B	S	C
0	0	$I_0 = C_i$	$I_0 = 0$
0	1	$I_1 = \bar{C}_i$	$I_1 = C_i$
1	0	$I_2 = \bar{C}_i$	$I_2 = C_i$
1	1	$I_3 = C_i$	$I_3 = 1$

$S_0 = B$

$S_1 = A$

☆ logical Exp. from mux

①



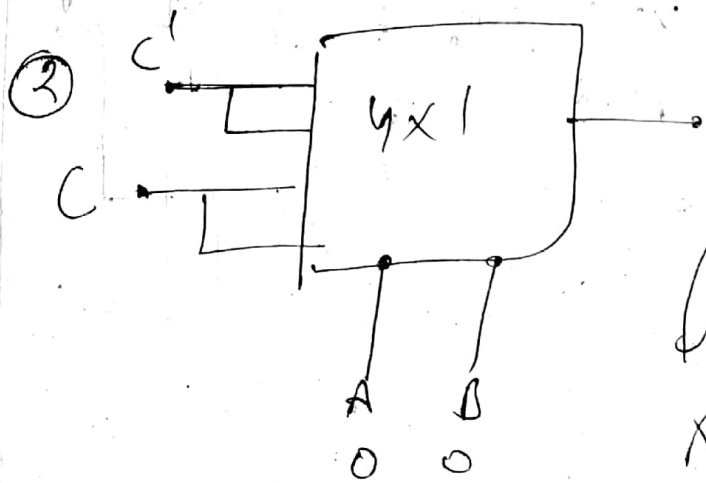
$Y_1 = A'B'C + A'B \cdot C + AB'1 + 0$

A	B
0	0
0	1
1	0
1	1

$A'C(B+B') + AB'$

$Y = A'C + AB'$

②



$Y_2 = A'B'C' + A'BC' + AB'C + ABC$

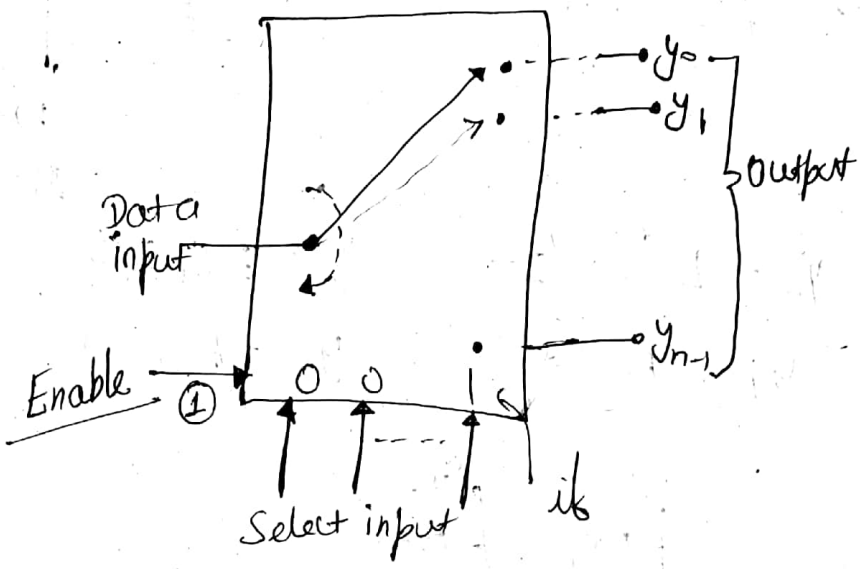
$A'C' + AC$

✡ Introduction to Demultiplexer:- DEMUX

⇒ One input and many output

⇒ Reverse operation of multiplexer

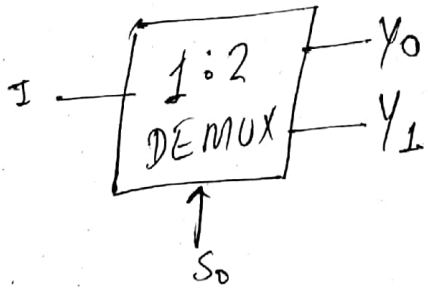
⇒ One to many ckt or data distribution



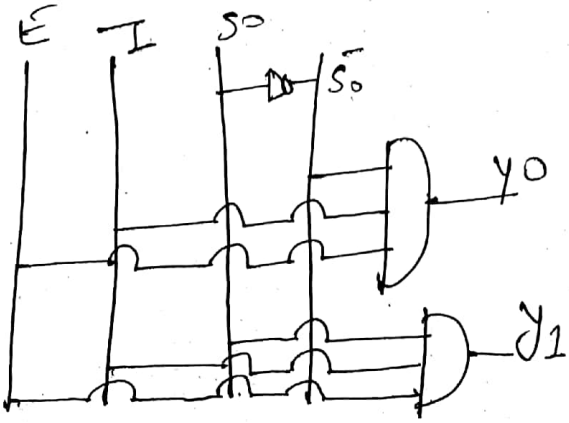
$n \rightarrow$ no. of o/p line m = no. of select line

$n = 2^m \Rightarrow m = \log_2 n$

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

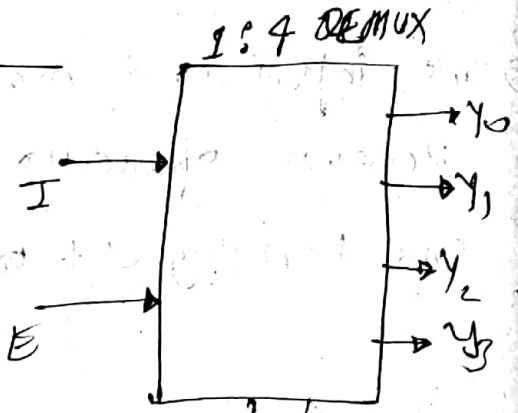


$y_0 = E \bar{S_0} I$

$y_1 = E S I$

☆ 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



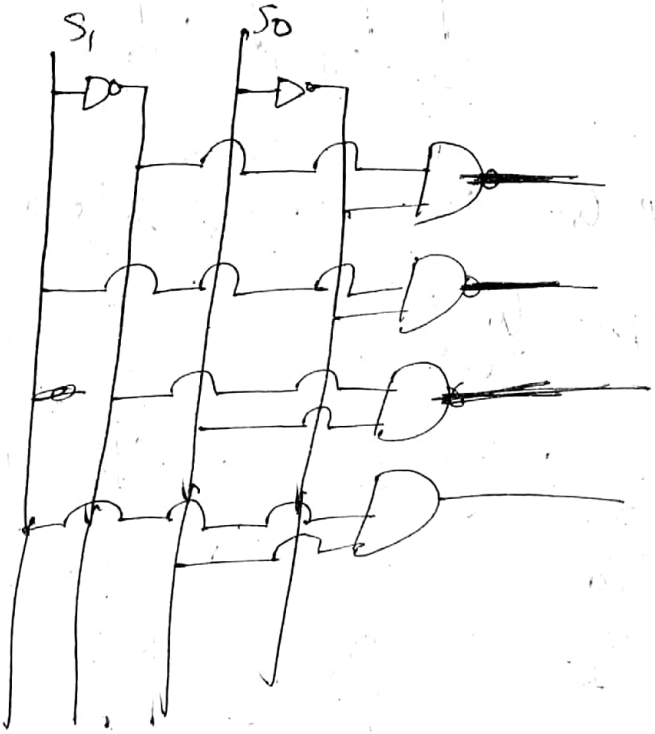
$M = \log_2 4 = 2$

$Y_0 = E S_1' S_0' I$

$Y_1 = E S_1' S_0 I$

$Y_2 = E S_1 S_2' I$

$Y_3 = E S_1 S_2 I$



☆Full subtractor Using 1:8 Demultiplexer

A	B	B_i	D	B_o
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

Standard SOP

$B_o(A, B, B_i)$

$= \sum m_1 m_2 m_3 m_7$

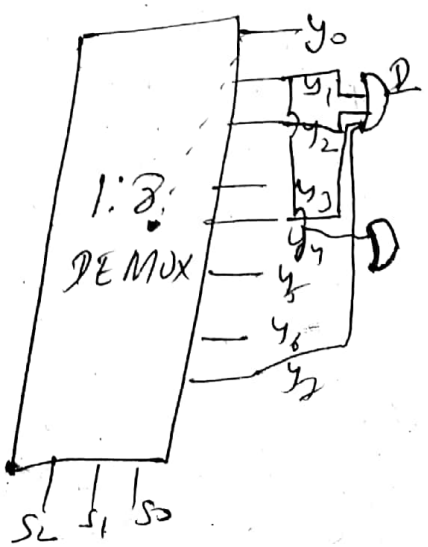
$= \sum m(1,2,3,7)$

$D(A, B, B_i)$

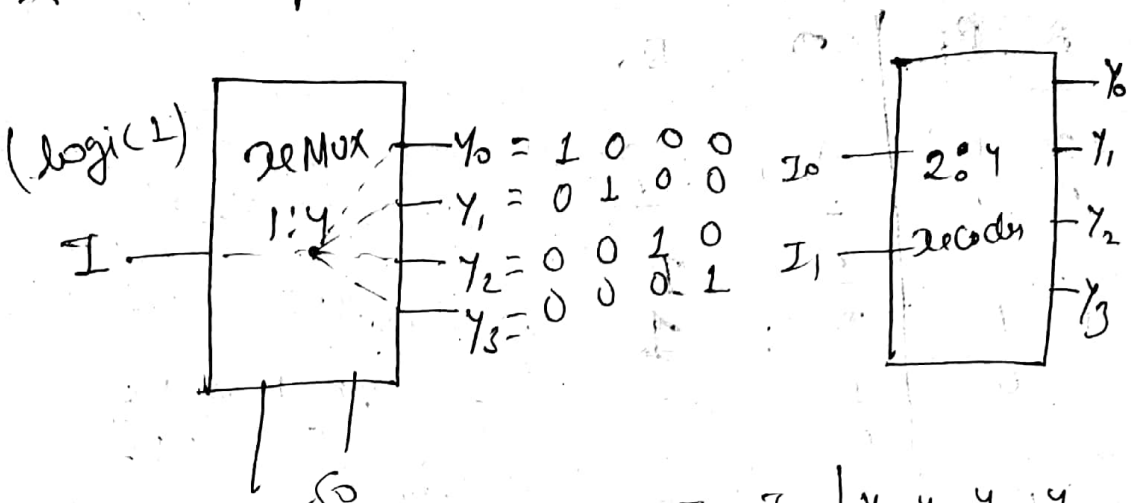
$= \sum m_1 m_2 m_4 m_7$

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

A (S_2)	B (S_1)	B_i (S_0)	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



☆ Demultiplexer as decoder



S_1	S_0
0	0
0	1
1	0
1	1

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

3:8 - decoder as 1:8 demultiplexer

☆ 2-bit comparator

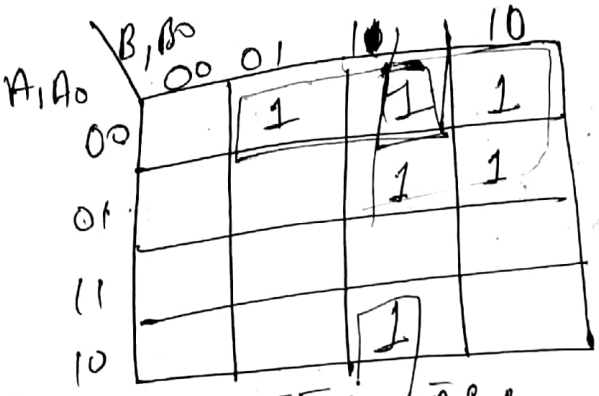
⇒ A digital comparator is a combinational circuit designed to compare two n-bit binary words.

⇒ Comparators has three outputs.

Inputs				Outputs		
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

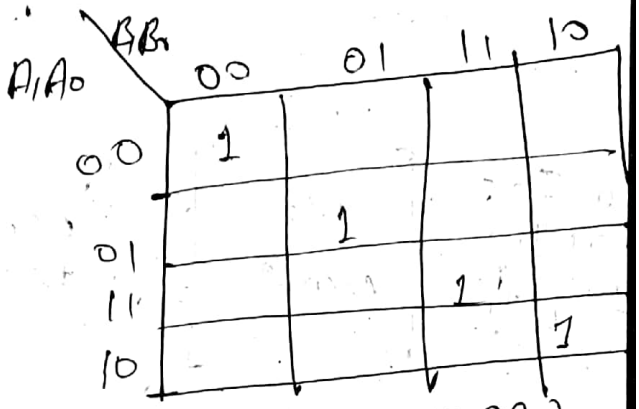
A_1 A_0	B_1 B_0	A<B	A=B	A>B
1 0	0 0	0	0	1
1 0	0 1	0	0	1
1 0	1 0	0	0	1
1 0	1 1	1	0	0
1 1	0 0	0	0	1
1 1	0 1	0	0	1
1 1	1 0	0	0	1
1 1	1 1	0	1	0

For A<B:-

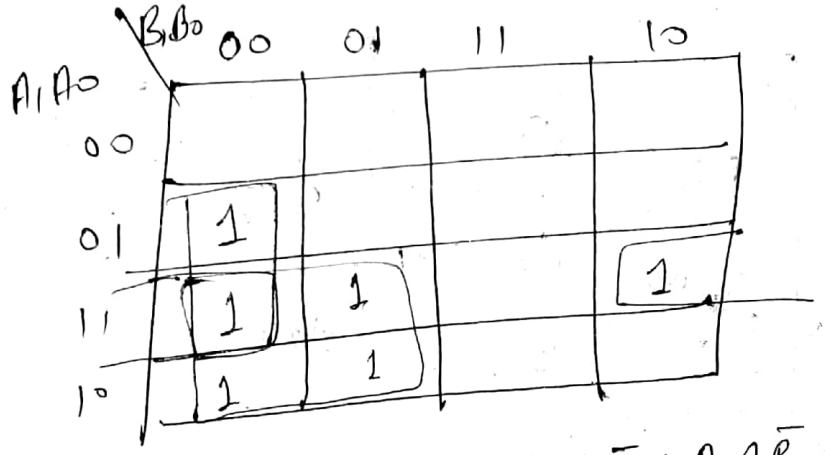


$A<B: \bar{A}_1 B_1 + \bar{A}_1 \bar{A}_0 B_0 + \bar{A}_0 B_0 B_1$

For A=B



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



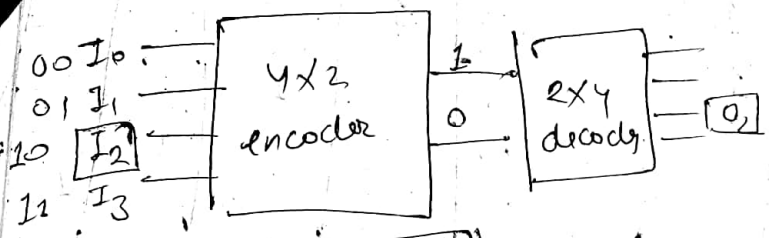
$A>B = A_0 \bar{B}_1 \bar{B}_0 + A_1 A_0 \bar{B}_0 + A_1 \bar{B}_1$

Introduction to Encoders & Decoders

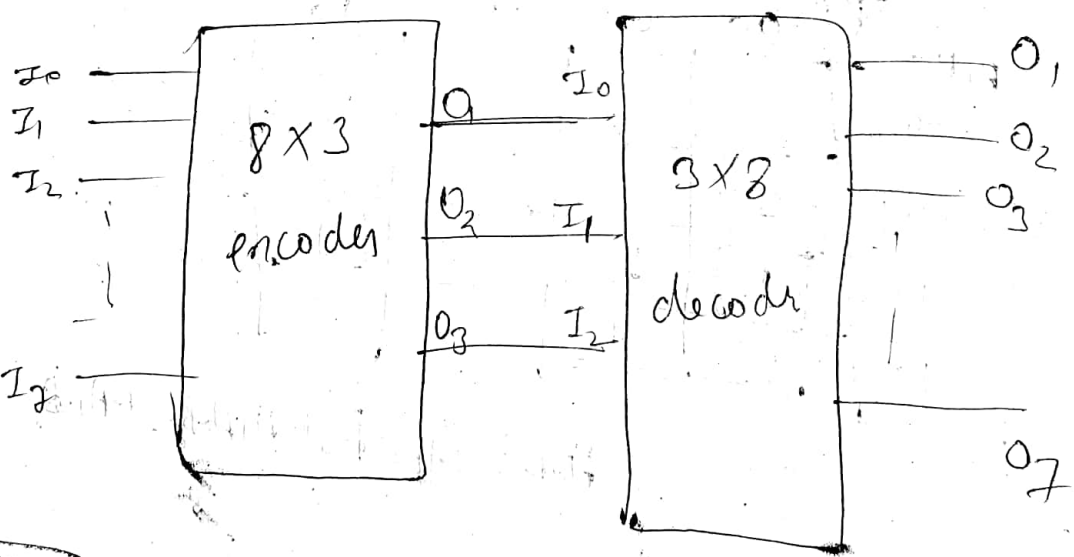
⇒ They are combination circuit

⇒ Encoders have "n" inputs and "m" output

⇒ Function of decoder is opposite to encoder



$n = 2^m$



Types

(I) Priority Encoder

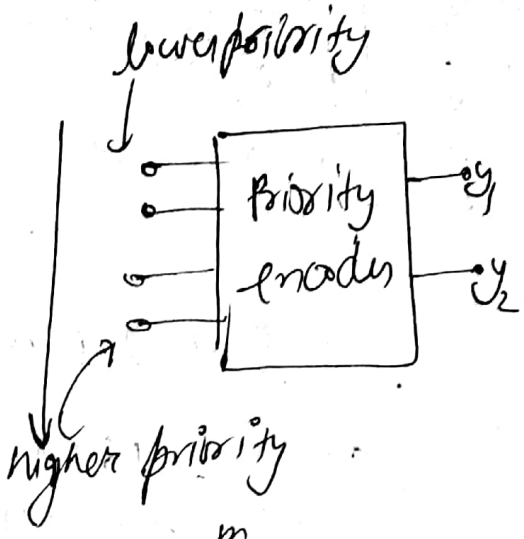
(II) Decimal to BCD encoder

(III) Octal to Binary Encoder

(IV) Hexadecimal to Binary Encoder

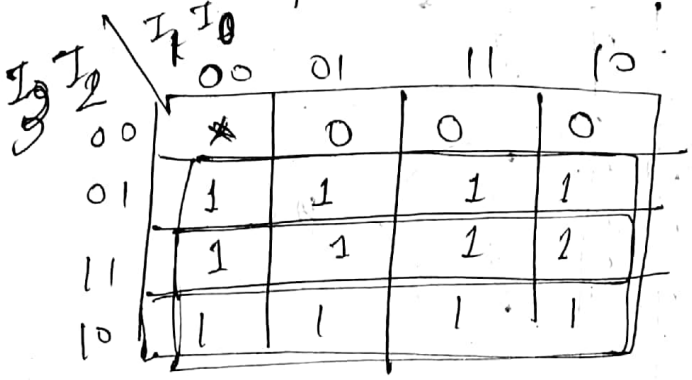
☆ Priority encoder

I_0	I_1	I_2	I_3	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

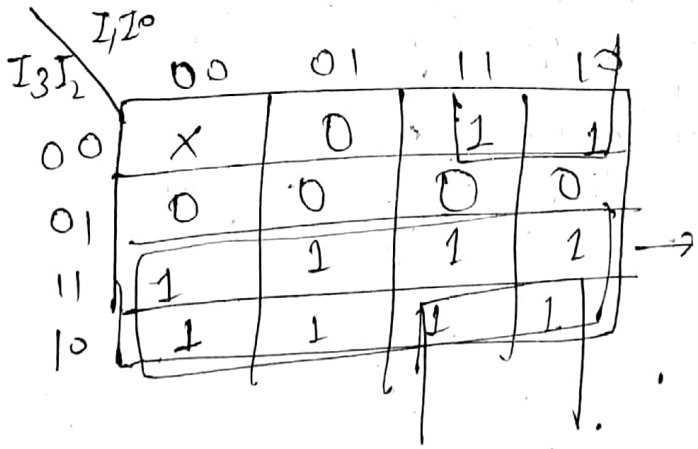


$n = 4 = 2^m$

$m_2 <$



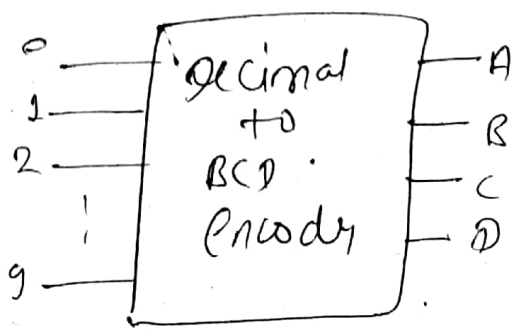
$y_1 = I_2$



$y_0 = I_3 + \bar{I}_2 I_1$

☆ Decimal to BCD encoder

Input	Output 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



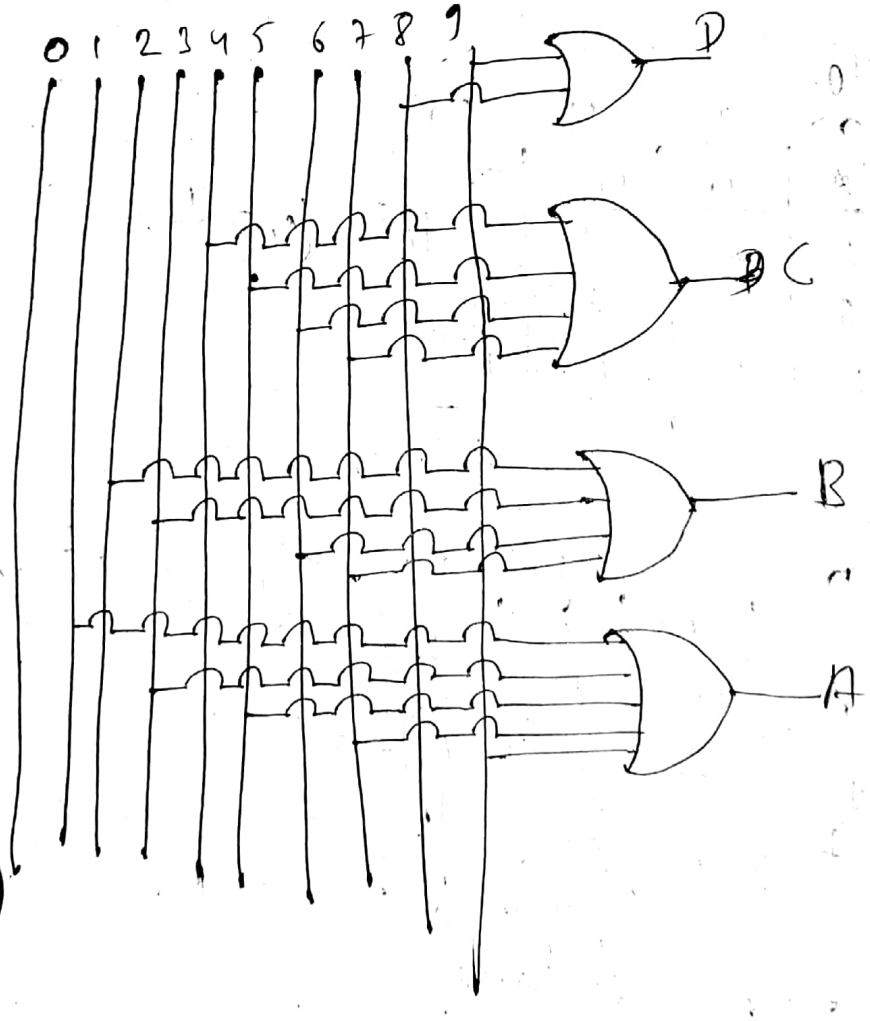
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

D = 8 + 9 (high jaha hai)

C = 4 + 5 + 6 + 7

B = 2 + 3 + 6 + 7

A = 1 + 3 + 5 + 7 + 9



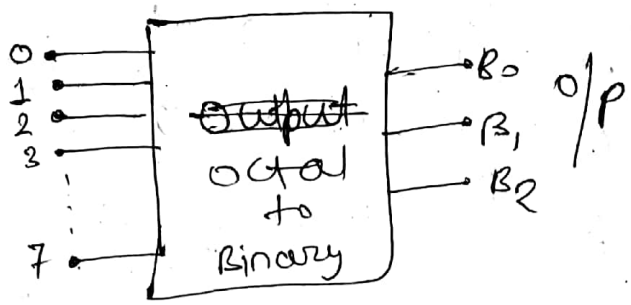
☆ octal to binary encoder

→ 8 input lines 3 output lines

r = 8 (radix = 8)

$(71)_8 \rightarrow (\quad)_2$

111 110 001



Input	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

$B_0 = 1+3+5+7$

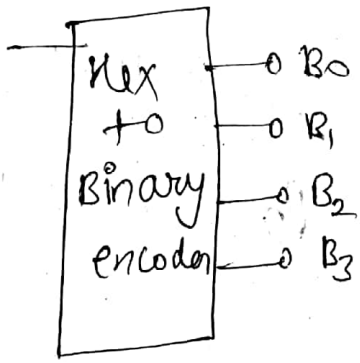
$B_1 = 2+3+6+7$

$B_2 = 4+5+6+7$

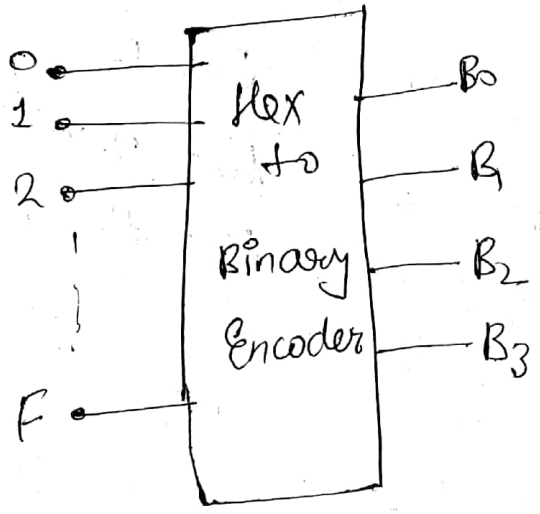
Make implimentation as previous

☆ Hexadecimal to Binary encoder

H	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



	B_3	B_2	B_1	B_0
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



$B_0 = 1+3+5+7+9+B+D+F$

$B_1 = 2+3+6+7+A+B+E+F$

$B_2 = 4+5+6+7+C+D+E+F$

$B_3 = 8+9+A+B+C+D+E+F$

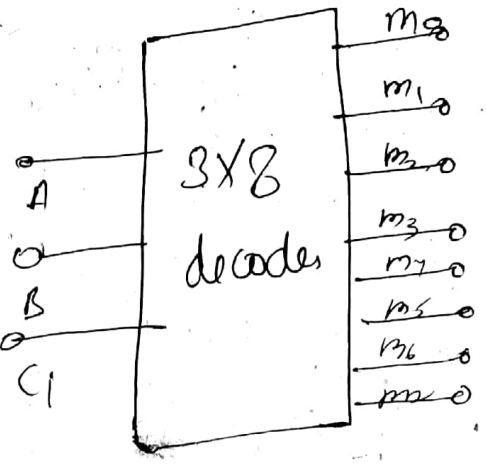
✱ Combination logic design using decoders

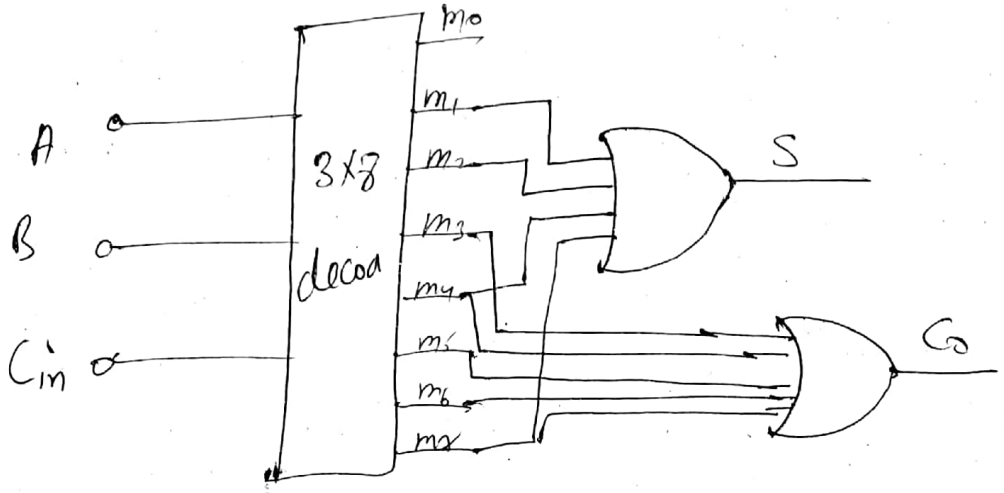
Truth table — Full adder implement

A	B	Cin	S	Co	
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

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

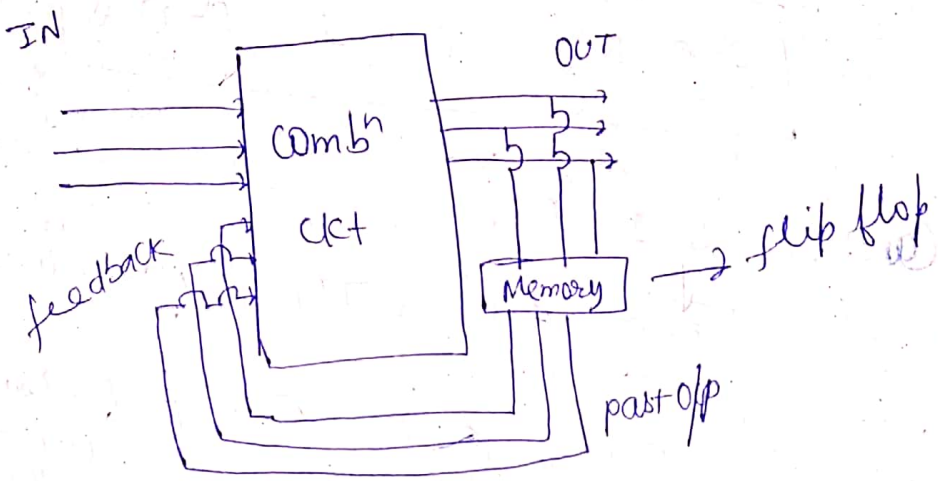
$F_{Co} = \Sigma m(3,5,6,7)$





☆ Introduction to Sequential Circuit.

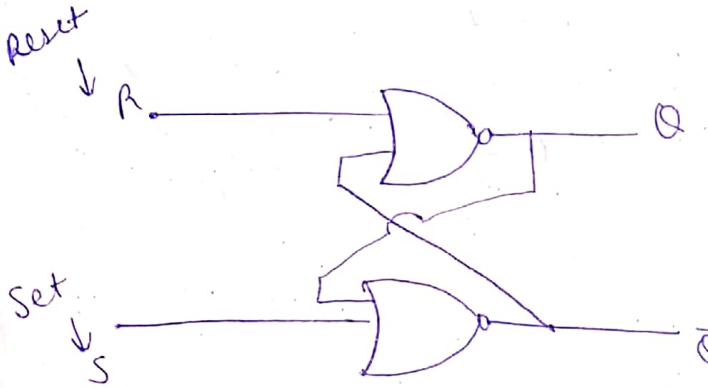
In sequential circuits, the present output depends on the the present input as well as the past outputs.



Sequential circuit

☆ SR Latch

The basic storage element is called LATCH: As the name suggest it latches "0" or "1"



Clock

Reset $Q=0$

Set $Q=1$

Case: 1 $S=0$ $R=1$

if any input is 1, output is 0.

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

Now remove the input

$S=0$ $R=0$ $\Rightarrow$ $Q=0$ $\bar{Q}=1$ ← Memory

Case II S=1 R=0 Q=1 & $\bar{Q}$=0

again remove the inputs

S=0 R=0 Q=1 $\bar{Q}$=0	← Memory

Case III S=1 R=1

$\bar{Q}$=0 Q=0	→ This is contradiction

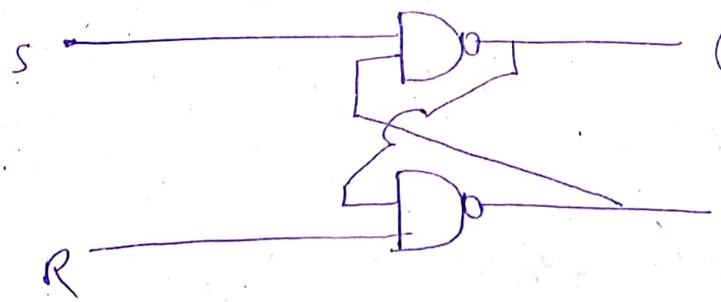
if S=0 R=0, Q=0 & $\bar{Q}$=1 [it is started with $\bar{Q}$]

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

Truth table

S	R	Q	$\bar{Q}$
0	0	Memory (as before)	
0	1	0	1
1	0	1	0
1	1	not used	

★ NAND S-R latch



S	R	Q	$\bar{Q}$
0	0	Not used	
0	1	1	0
1	0	0	1
1	1	Memory	

★ Clock

A digi
voltag
and
diff
The
meani
same.

Uses

A cloc
open
proce
circ
lig
mar
le

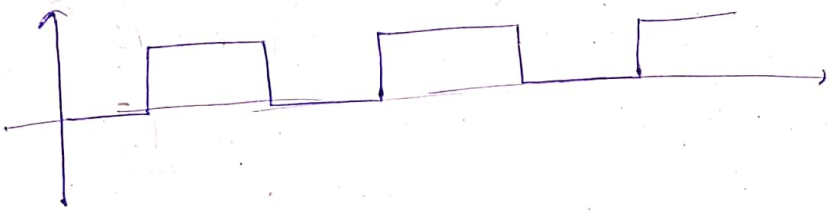
ed

☆ Clock

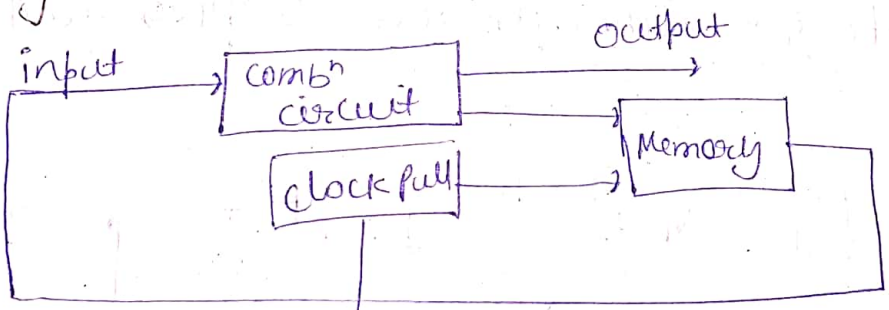
A digital clock signal is basically a square wave voltage, it has only two levels, one is zero and other is high, which the high level can be different according to the requirements of the circuit. The signal mostly has 50% ~~only~~ duty cycle, meaning the duration of high and a zero is the same. The frequency can be anything as needed.

Uses

A clock plays very important role as it is used to open and close digital paths, allow or stop a process and in general provide timing for the circuit. You can compare a clock with the traffic lights. They stop and allow the traffic in a timely manner so that the traffic can flow smoothly with the least delays.



☆ Triggering Methods



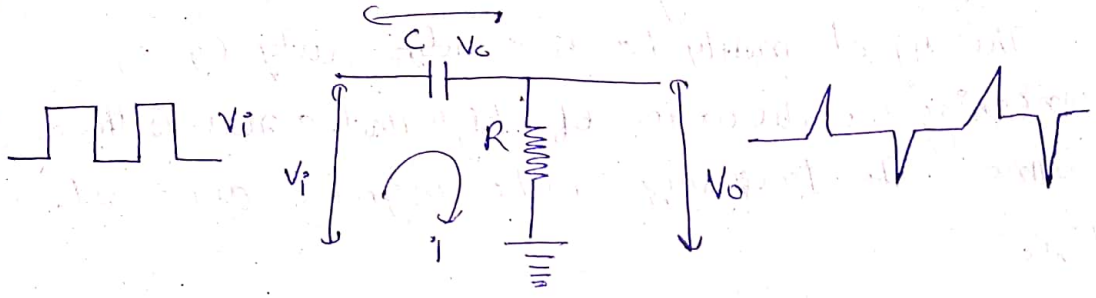
edge (+ve) trigger
↓
When pulse high to low

Control i/p

(-ve) → when high to low

☆ How to get Edge Triggering

⊕ edge ⊖ve edge



$V_i - V_c - V_o = 0$

$V_i = V_c + V_o \rightarrow iR$

$Q = CV_c$

$\int i\,dt = CV_c$

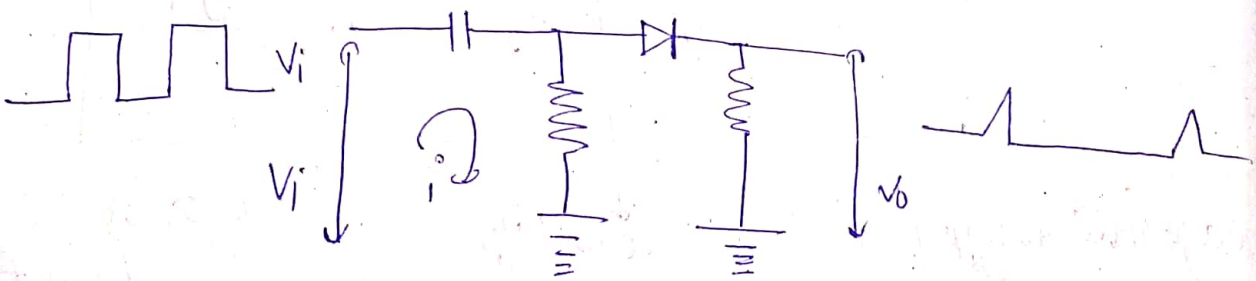
$V_c = \frac{1}{C}\int i\,dt$

$V_i = \frac{1}{C}\int i\,dt + iR$

$V_i = \frac{1}{C}\int \frac{V_o}{R}\,dt = \frac{dV_i}{dt} = \frac{1}{RC}V_o$

$V_o = RC\frac{dV_i}{dt}$

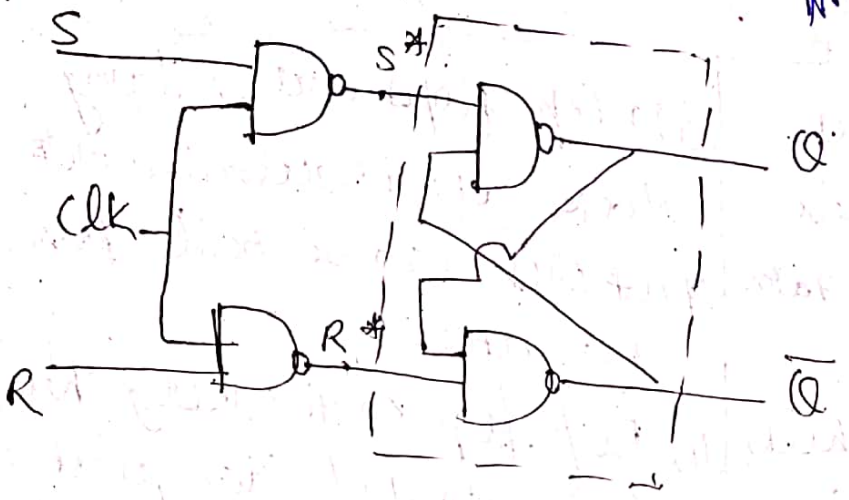
to get either positive or -ve spikes add diode



☆ Diff

Latches	Flip Flops
(I) Latches are building blocks of sequential ckt and these can be built from logic gates.	(I) Flip flops are building blocks of sequential ckt. But these can be built from the latches.
(II) Latches continuously checks its inputs and changes it outputs correspondigly.	(II) Flip flop continuously checks it inputs and change it output Correspondingly only at time determined by clocking signal.
(III) The latch is sensitive to the duration of the pulse and can send or receive the data when switch is on.	(III) Flipflop is sensitive to a signal change. They can transfer data only at the ~~signal~~ single instant and data can't be changed until next signal change. Flip flops are used as registers.
(IV) It is based on the enable function input.	(IV) It works on the basis of clock pulse
(V) It is level triggered, it means that the output of the present state and input of the next steps depends on the level that is binary input 1 or 0.	(V) It is an edge triggered, it means that the output and the next state input changes when there is a change in clock pulse whether it may a +ve or -ve clock pulse.

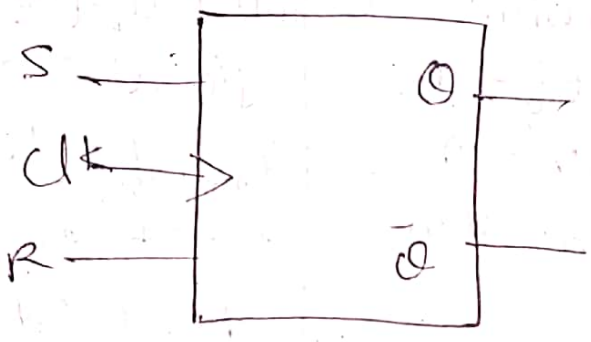
★ Introduction to SR flip flop



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

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

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



Clk	S	R	Q	Q̄	Clk = 0

Characteristic table

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

~~Excitable~~ Excitation table

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

Q_n \ SR	00	01	11	10
0	0	0	X	1
1	1	0	X	1

$Q_{n+1} = I + II$

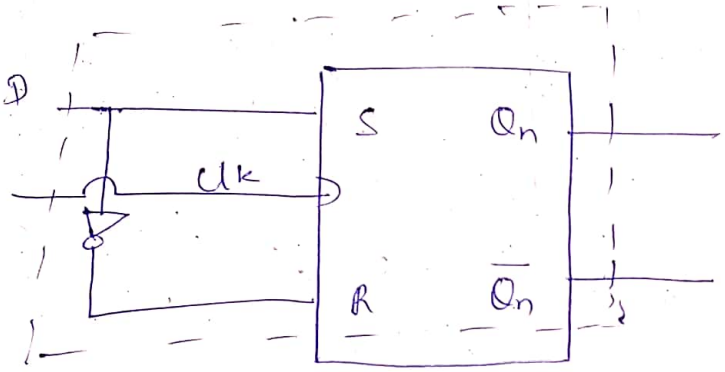
$Q_{n+1} = S + Q_n\bar{R}$

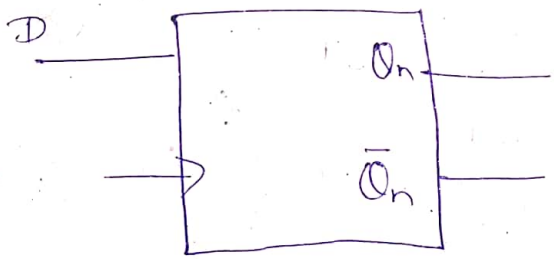
★ Introduction

Data

T.T. for SR flip flop.

Clk	S	R	Q_{n+1}
0	X	X	Q_n / Memory
1	0	0	Q_n
1	0	1	0 → Complimentry input
1	1	0	1
1	1	1	invalid





T.T for D flip-flop

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

Char. table

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

$Q_{n+1} = D$

Excitation table

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

★ J-K

J

clock

K

Clk=0

Clk=1

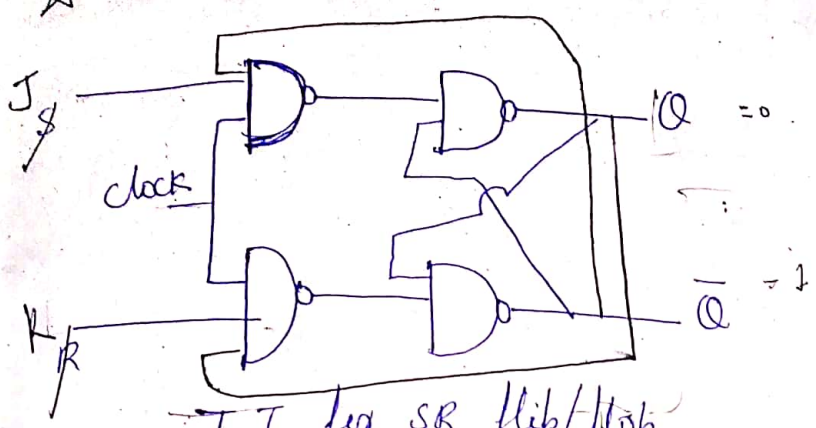
Clk=

Clk=

Char.

Scanned by CamScanner

★ J-K flip-flop



T.T for SR flip/flop

clk	S (J)	R (K)	Q_{n+1}
0	x	x	Q_n (Memory)
1	0	0	Q_n (Memory)
1	0	1	0
1	1	0	1
1	1	1	Not used. Q_n (toggle)

Run

clk=0 - Memory

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

clk=1 J=0 K=1, Q=0, $\bar{Q}$=1 $Q_{n+1} = \bar{Q}_n$

clk=1 J=1 K=1 Q=0,1,0,1 ---

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

Char. table

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	0
1	0	1	0
1	1	0	1
1	1	1	0

✡ Machine arithmatic / Programmable logic devic

✡ Machine arithmatic

Data Representation

1. Fixed point number
2. Floating Point number

(1.) Fixed point no.

$(+7)_{10} = (0111)$ 4-bit representation

$(-7)_{10} = ?$

Step 1: 1's Complement

$\xrightarrow{1's}$ (1000)

Step 2: - 2's complement

$$\begin{array}{r} 1000 \\ +\quad\quad 1 \\ \hline (1001)_2 \end{array}$$

(2) Floating Point representation

It has three parts

a. Mantissa

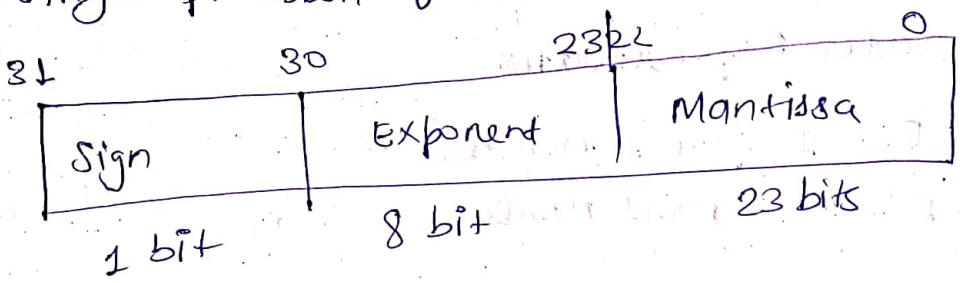
b. Base

c. Exponent

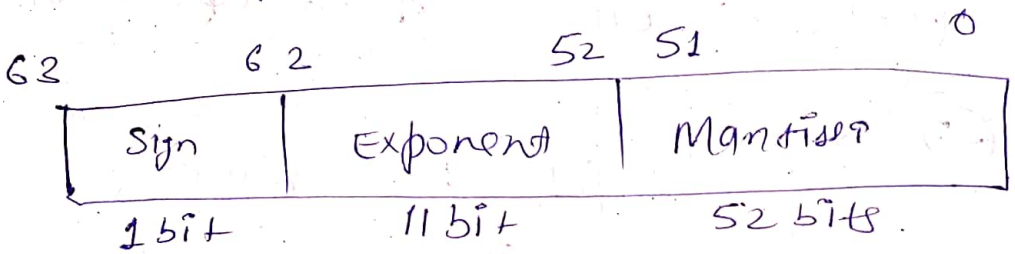
Number	Mantissa	Base	Exponent
1. 3×10^6	3	10	6
2. 110×2^8	110	2	8
3. 6132.784	6132784	10	3

IEEE 754 floating point number representation

(a) single precision format:



(b) Double precision format



Q $(259.125)_{10}$ in single and double precision.

Convert decimal no. to binary no.

$(259)_{10} = (100\,0001\,0011)_2$

$(0.125)_{10}$

$0.125 \times 2 = 0.250$	0
$0.250 \times 2 = 0.500$	0
$0.5 \times 2 = 1.0$	1

$(259.125)_{10} = (100\,0001\,0011.001)_2$

(2) Normalise the no.

$(1.N)\,2^{E-127}$ (single bias)

$(1.N)\,2^{E-1023}$ (double bias)

$(259.125)_{10}$

100111011.001

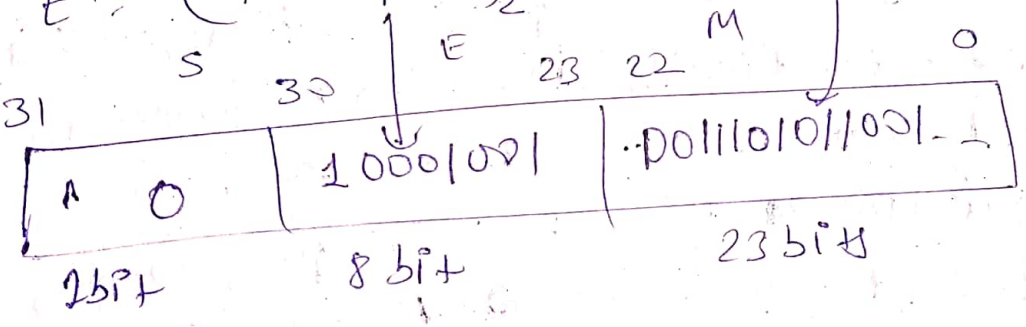
$\underbrace{1.00111011001}_{N} \times 2^{10}$

(3) single precision format

$(1.N)\,2^{E-127}$ $\quad E-127=10$, $E=137$

$(1.00111011001)\,2$

$E = (10001001)_2$



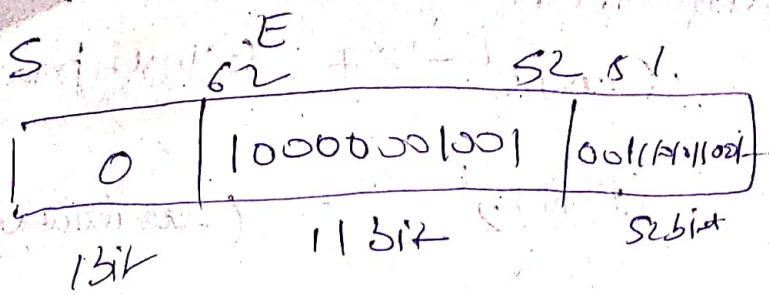
(4) Double precision format

$(1.N)\,2^{E-1023}$ $\quad E-1023$

$(1.00111011001)\,2$

$E-1023=10$

$E=1033$



Double precision format

* Programmable logic device (PLDs)

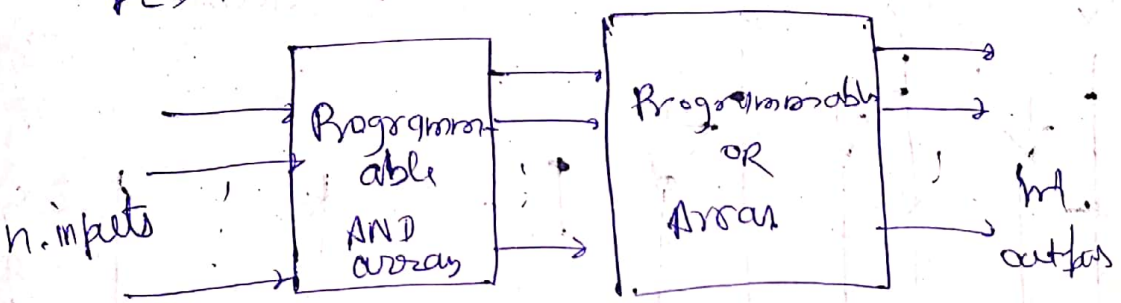
→ PLDs are Integrated circuit. They contain an array of AND gates & another array of OR gates. There are three kinds of PLDs based on the type of arrays which has programming features.

The process of entering the information into these devices is known as programming. Basically, users can program these devices or ICs electrically in order to implement the Boolean function based on the requirements. Here, the term programming refers to hardware programming but not software programming

(1) PROM
(2) PLA
(3) PAL

☆ Programmable Logic Array (PLA)

PLA is a programmable logic device that has both programmable AND array & Programmable OR array, hence it is most flexible PLD.



Here the inputs of AND gates are programmable hence AND gate will get both normal and complemented inputs of variables.

So, we get generate required product terms by using these AND gates.

∴ the inputs of OR gates are also programmable output of AND gate will act as input for OR gate

Therefore the outputs of PAL will be in the form of sum of products (SOP)

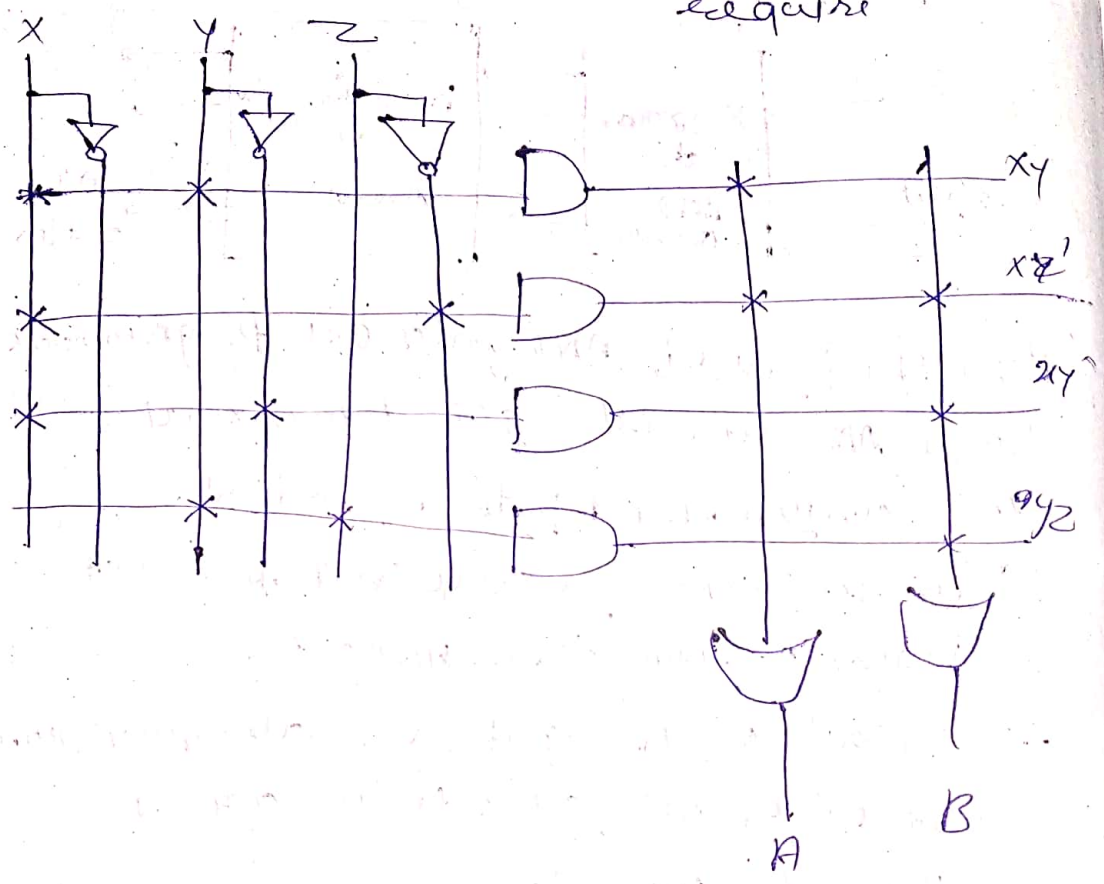
Let us implements following boolean functions using PLA

$$A = XY + XZ'$$

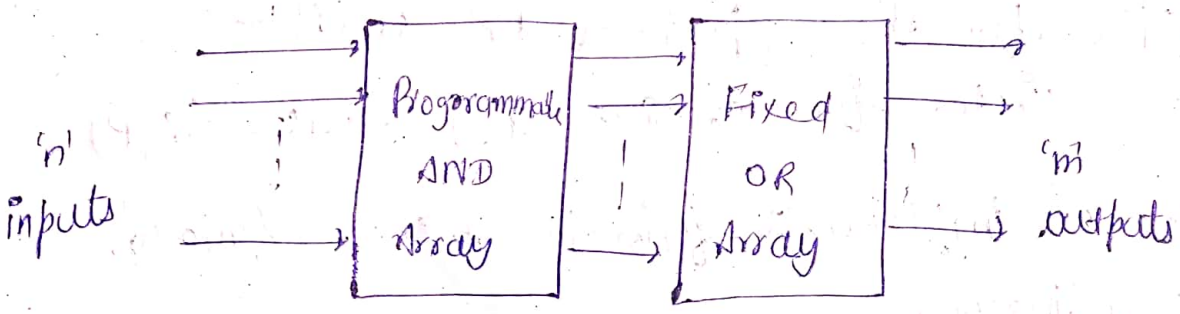
$$B = XY' + YZ + XZ'$$

no of products terms are 5 in which 1 is common. Hence 4 AND gate will required.

As well as, no. of function is 2. Hence two OR gate will require



★ Programming Array Logic (PAL)



→ It has programmable AND array and fixed OR array.

~~Block diagram~~

(1) Boolean function.

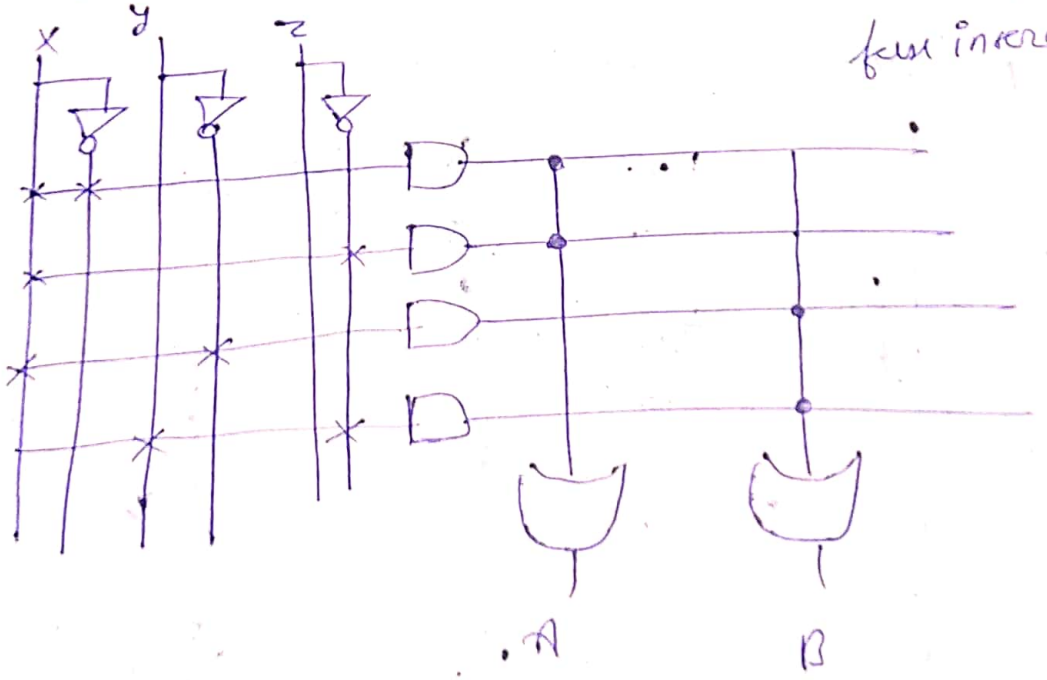
$A = XY + XZ'$

$B = XY' + YZ'$

} find which has max. no. of minimum

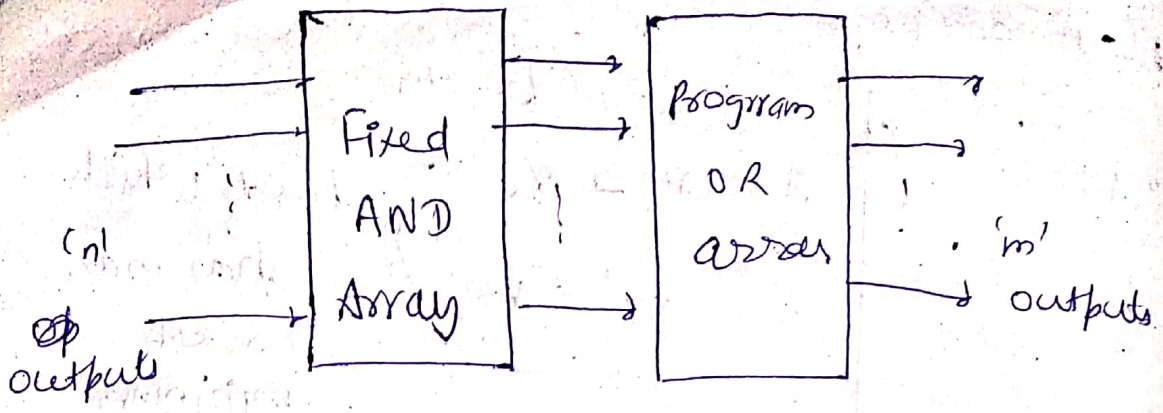
$\therefore$ (2)

Hence we required (2x2) programmable AND gate & two fixed OR gate for producing two function



★ Programmable Read Only.

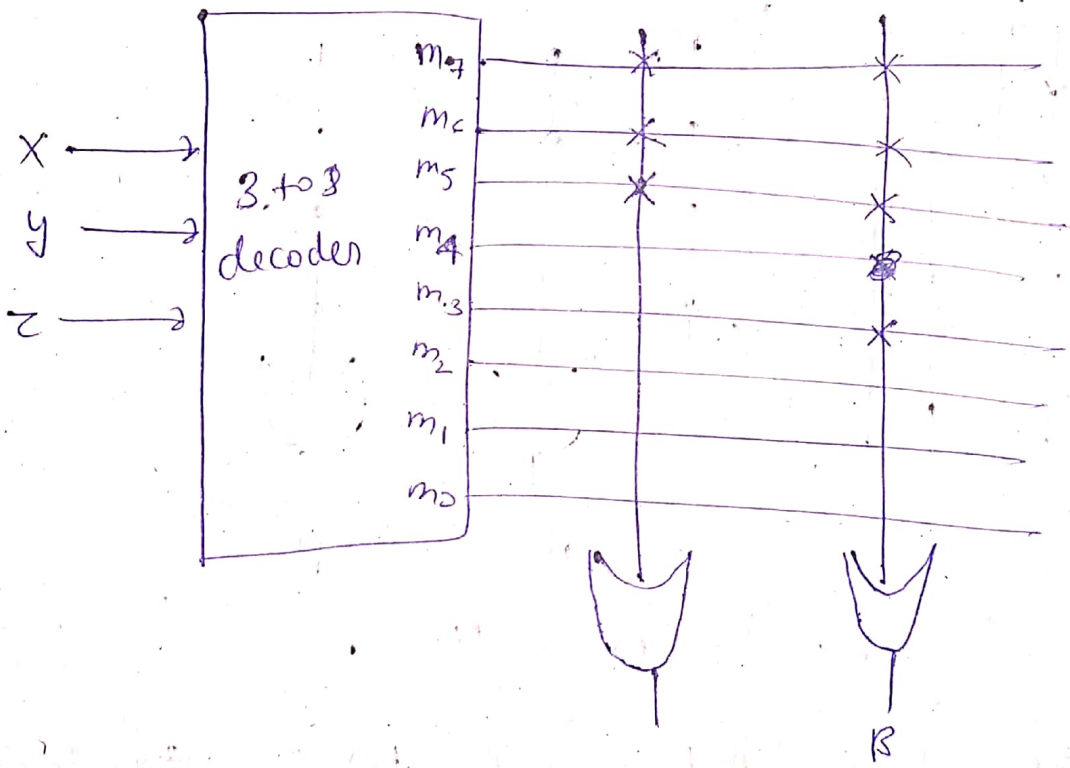
→ Read only Memory (ROM) is a memory device, which stores the binary information permanently. The user has the flexibility to program the binary information electrically once by using PROM programmer.



As AND gates are not of programmable type. So, we have to generate 2^n product terms by using 2^n AND gate having n inputs each.

$$A(X,Y,Z) = \sum m(5,6,7)$$

$$B(X,Y,Z) = \sum m(3,5,6,7)$$



☆ Carry lookhead adder

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

$$C_0 = G_0 + P_0 C_{-1}$$

$$C_1 = G_1 + P_1 C_0 = G_1 + P_1 (G_0 + P_0 C_{-1})$$

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

$$C_2 = G_2 + P_2 C_1$$

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

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

$$C_3 = G_3 + P_3 C_2$$

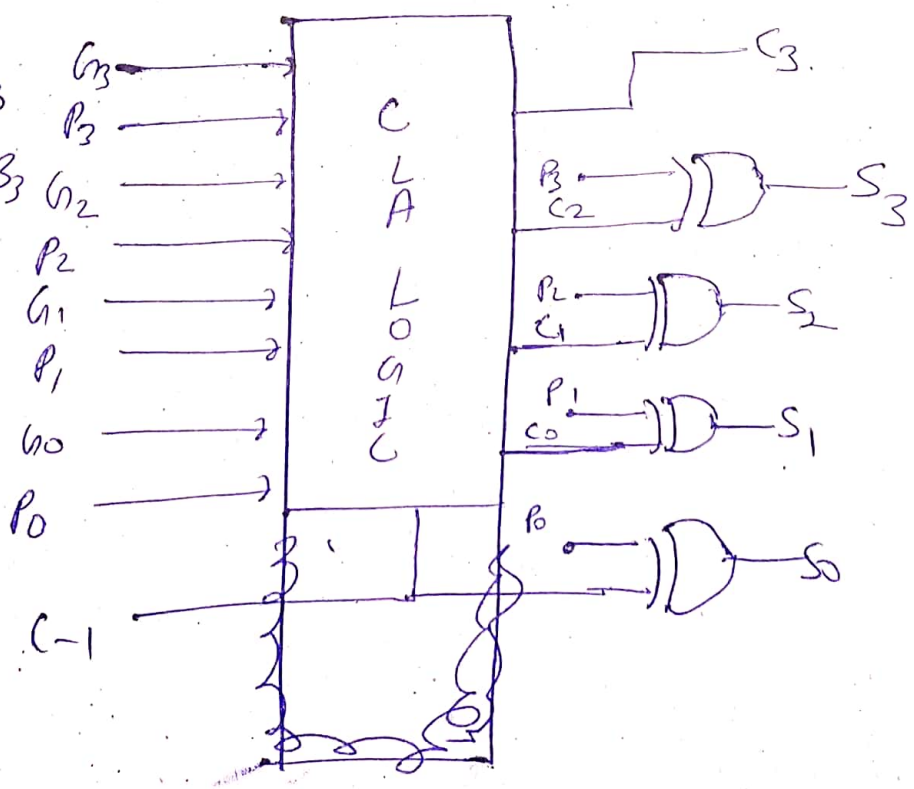
$$= G_3 + P_3 (G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1})$$

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

$G_3 = A_3 \cdot B_3$

$P_3 = A_3 \oplus B_3$

$P_2 = A_2 \oplus A_1$



Truth table for 1-bit

A	B	C_i	C_{i+1}	Condn
0	0	0	0	No carry generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No Carry Propogate
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry generate
1	1	1	1	

AB=0

$C_{i+1} = C_i$

~~A⊕B~~ C_i

~~No carry~~

AB=1

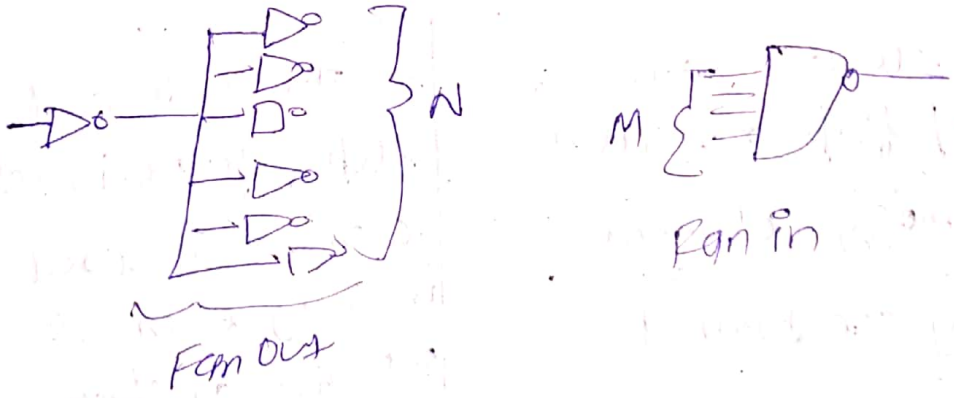
A_1 A_0 B_1 B_0 C_i ← Input

☆ Fan in and Fan-Out

Fan in → The fan-in defined is the maximum no. of inputs that a logic gate can accept. If no. of input exceeds, the output will be undefined or incorrect.

Gates with large fan-in are bigger and slower

Fan-out → Max no. of input (loads) that can be connected to the output of a gate with degrading in normal operation.



☆ Noise immunity

↓

Noise immunity is the ability of a device or component to operate in the presence of noise disturbance. The higher the level of noise at which the equipments its operating ability, the higher its noise immunity

Noise margin

In a digit circuit, the noise margin is the amount by which signal exceeds the threshold for a proper 0 or 1.

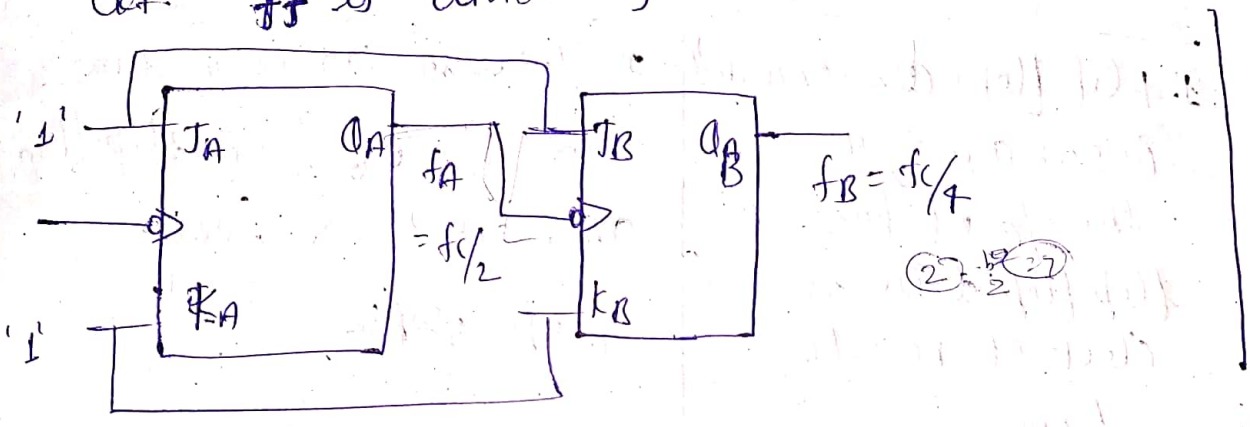
☆ Counters

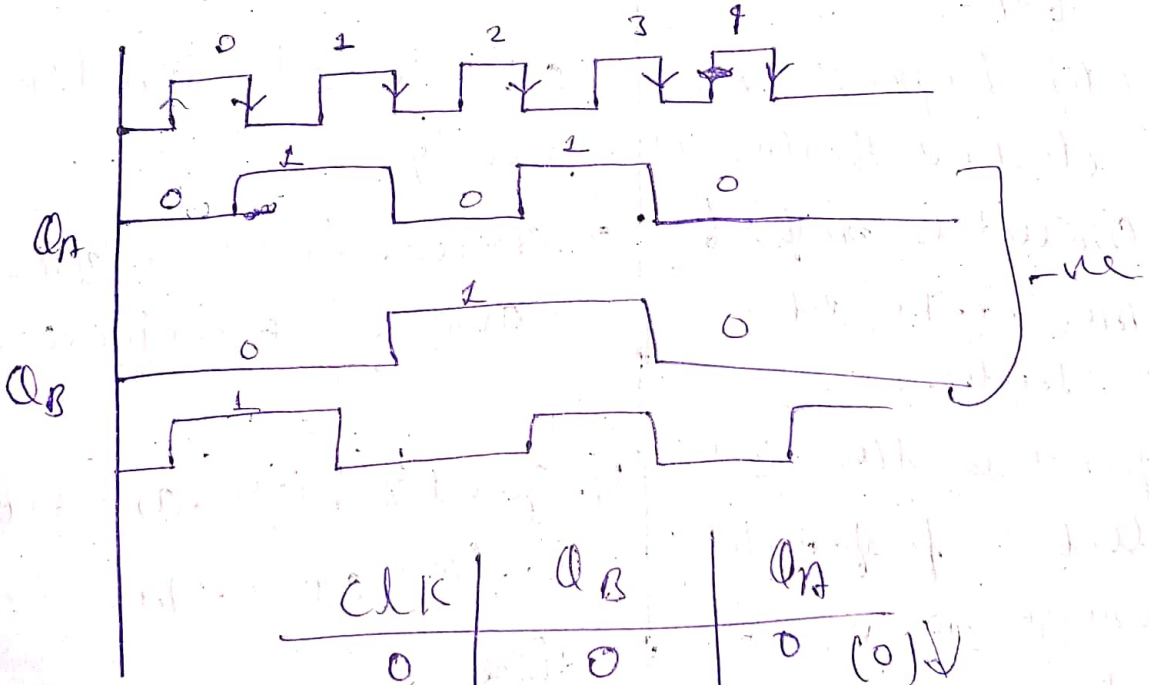
☆ Difference b/w Synch. and Asynch. sequential circuit

Synchronous Seq. circuit	Asynchronous Seq. Ckt.
1. These circuit are easy to design	1. These circuit are difficult to design
2. A clocked ~~element~~ flip flop acts as a memory element	2. An unclocked flip flop or time delay element is used as memory element
3. They are slowed	3. Faster as clock is not present
4. The status of memory element is affected only at the active edge of clock, if input is changed.	4. The status of memory element will change any time as soon as input is changed.
5. ~~Latches~~ flip flops are used in asynchronous circuits.	5. Whereas ~~flip flops~~ latches are used in asynchronous ckt

☆ Introduction to Counters

⇒ A counter is a device which stores (and sometime displays) the no. of times a particular event or process has occured. Counters in digital electronics for counting purpose, they can count specific event happening in Ckt. ff is divided by 2 clk :-





Clk	Q_B	Q_A
0	0	0 (0)
1	0	1 (1)
2	1	0 (2)
3	1	1 (3)

★ Types of Counters

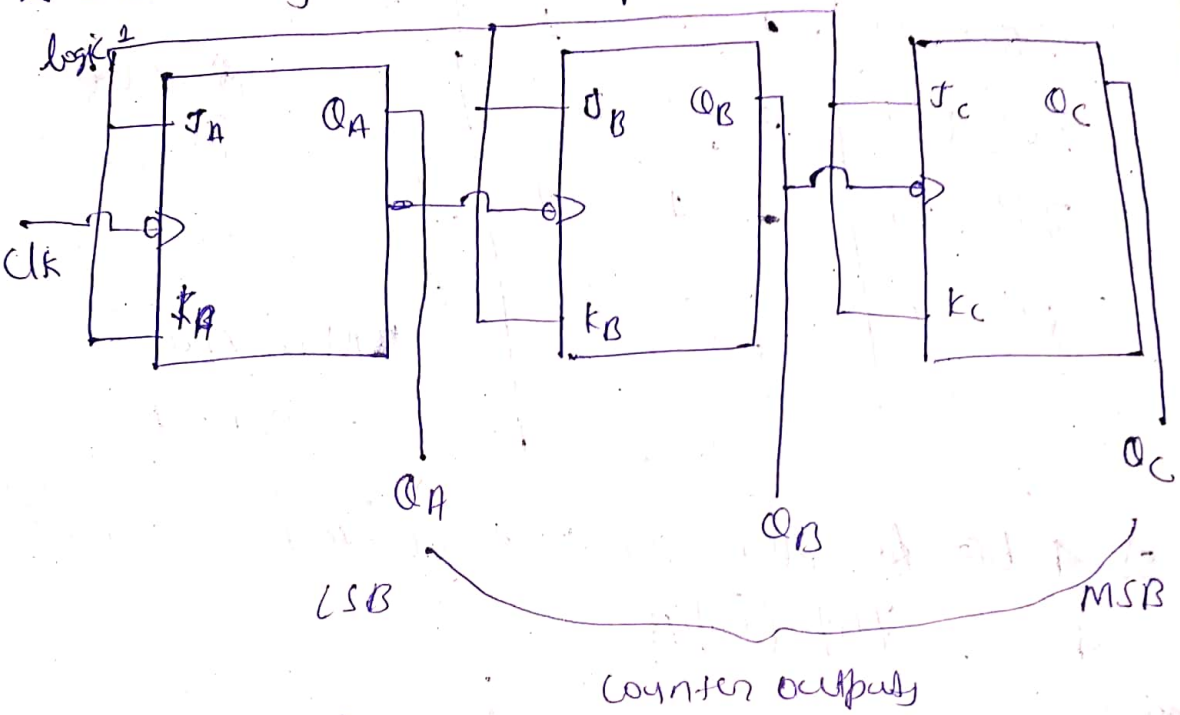
Counter counts the clock pulse

Asynchronous Counters (Ripple)	Synchronous Counters
1. Flip flops are connected in such a way that the o/p of first flip flop drives the clock of next flip flopes.	1. There is no connection b/w o/p of first flip flop and clock of next flip flopes.
2. Flip flopes are not clocked simultaneously	2. are clocked simultaneously
3. Circuit is simple for more no. of states.	2. Circuit becomes complicated as no. of states increases.
4. Speed is slow as clock is propagates through no. of stages	4. Speed is high as clock is given at a same time.

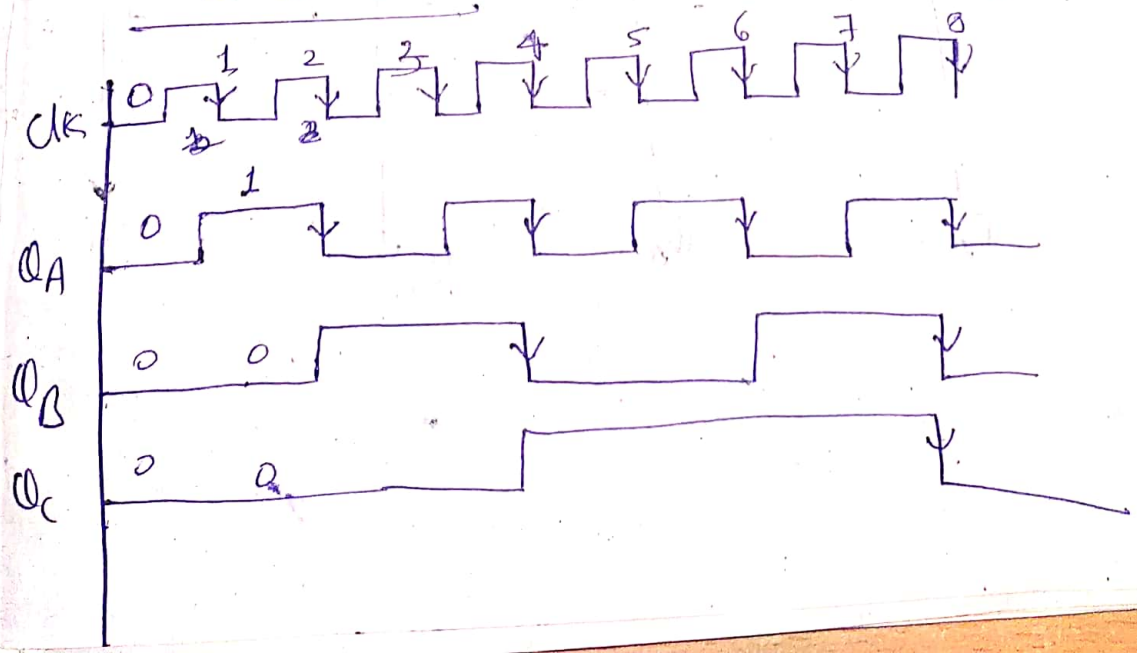
Counters

- Up counter: 0-1-2-3---
- down Counters: 7-6-5--0
- up-down Counters

☆ 3 bit Asynchronous Up Counters.



Q_C	Q_B	Q_A
0	0	0
↓		
1	1	1



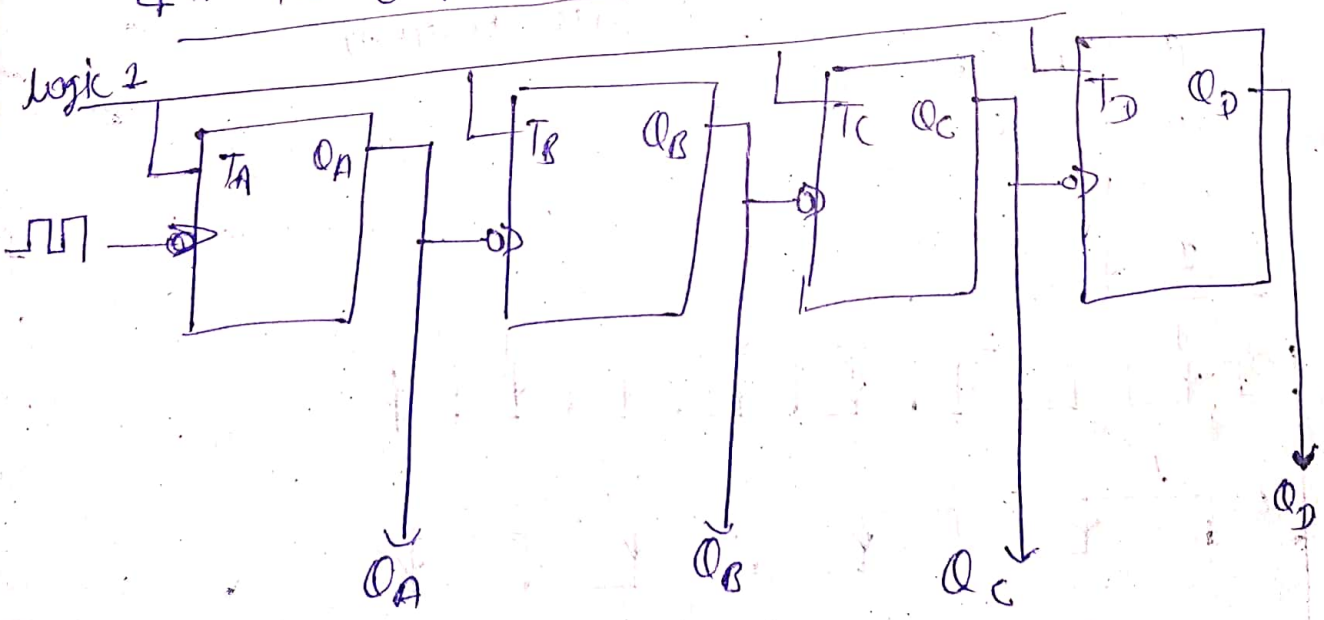
Clk	Q_C	Q_B	Q_A	Decimals Eq.
Intl	0	0	0	0
1st (↓)	0	0	1	1
2nd (↓)	0	1	0	2
3rd (↓)	0	1	1	3
4th (↓)	1	0	0	4
5th (↓)	1	0	1	5
6th (↓)	1	1	0	6
7th (↓)	1	1	1	7
8th (↓)	0	0	0	0

8 states
$2^n = 2^3 = 8$

Max. count
$= 2^n - 1 = 7$

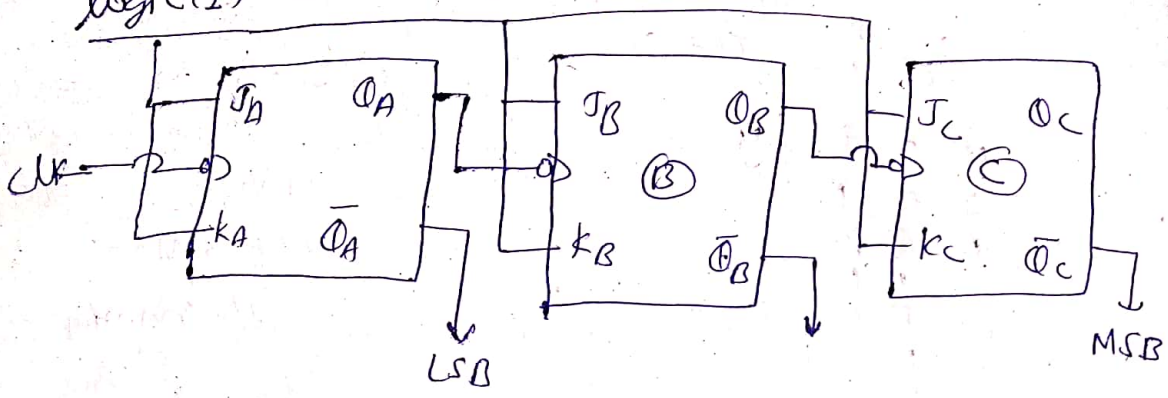
☆ 4-bit Asynchronous UP Counter

4 - T- flip flops

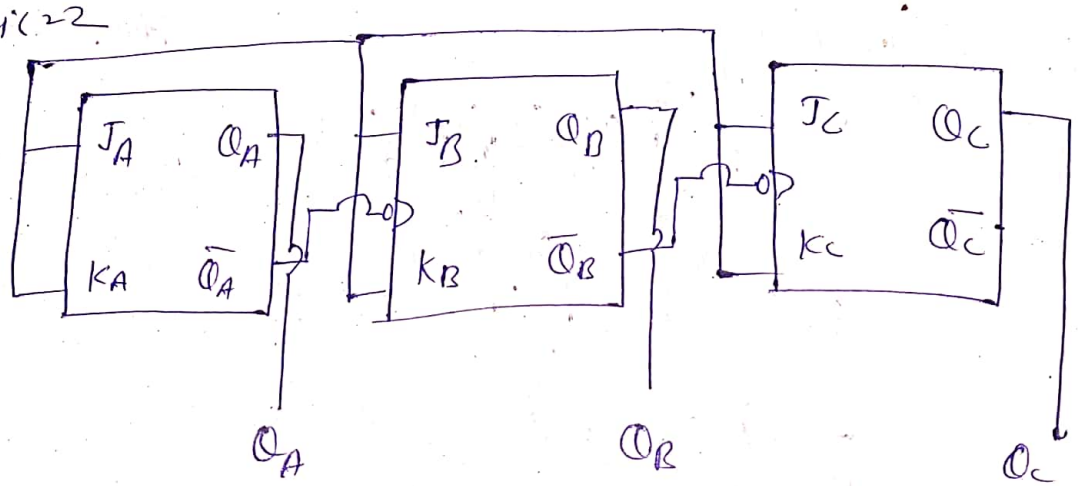


✡ 3 bit & 4 bit Asynchronous Down Counters

logic (1)



logic 2



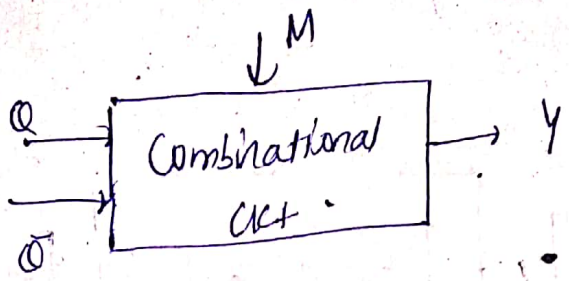
✡ 3-bit & 4 bit UP/Down Ripple (Asynchronous) counters

→ We have designed the up and down counters seperately.

⇒ But in practice, we need both mode combined

⇒ A mode control inputs (M) is used to select either up or down mode.

⇒ A combination ckt. is required b/w each pair of flop.

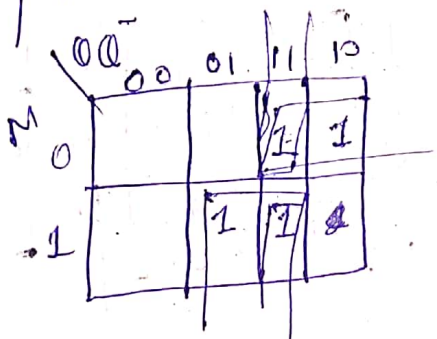


M	Q	Q̄	y
0	0	1	0
0	0	1	0
0	1	0	1
0	1	0	0
1	0	1	0
1	0	1	0
1	1	0	1
1	1	0	1

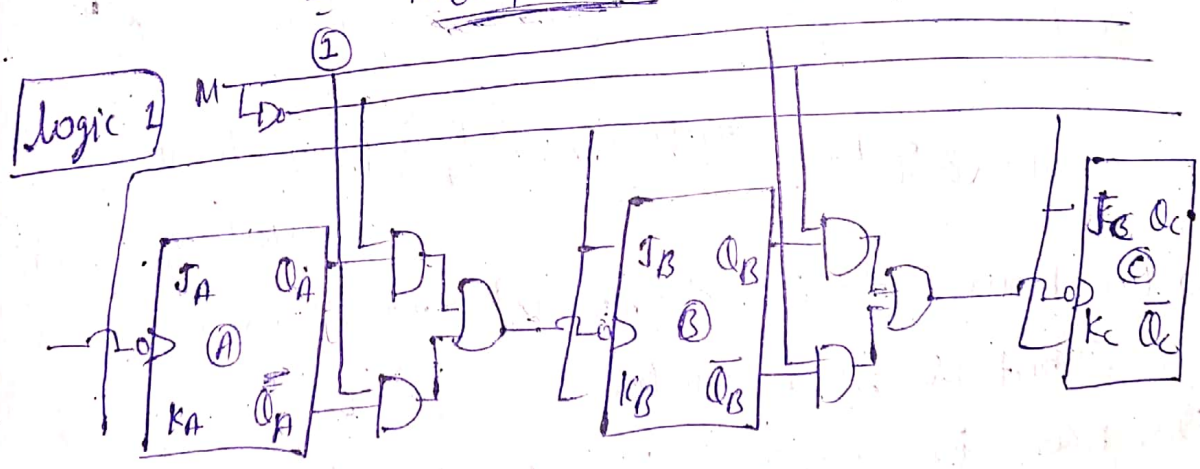
if Cond^n is Given

Suppose when M=0 Up Counting

when M=1 down Ct



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



☆ Modulus of the counter & counting up to particular value

→ it tells the total no. of state of counters or to which value it counts.

⇒ 2-bit ripple counter is called MOD-4 or modulo counter,

n → no. of bit

MOD no. $= 2^n$

Q MOD = 6 counter using MOD-8 - counter

States = (6)

Max - counter : 6-1 = (5)

000 (0)

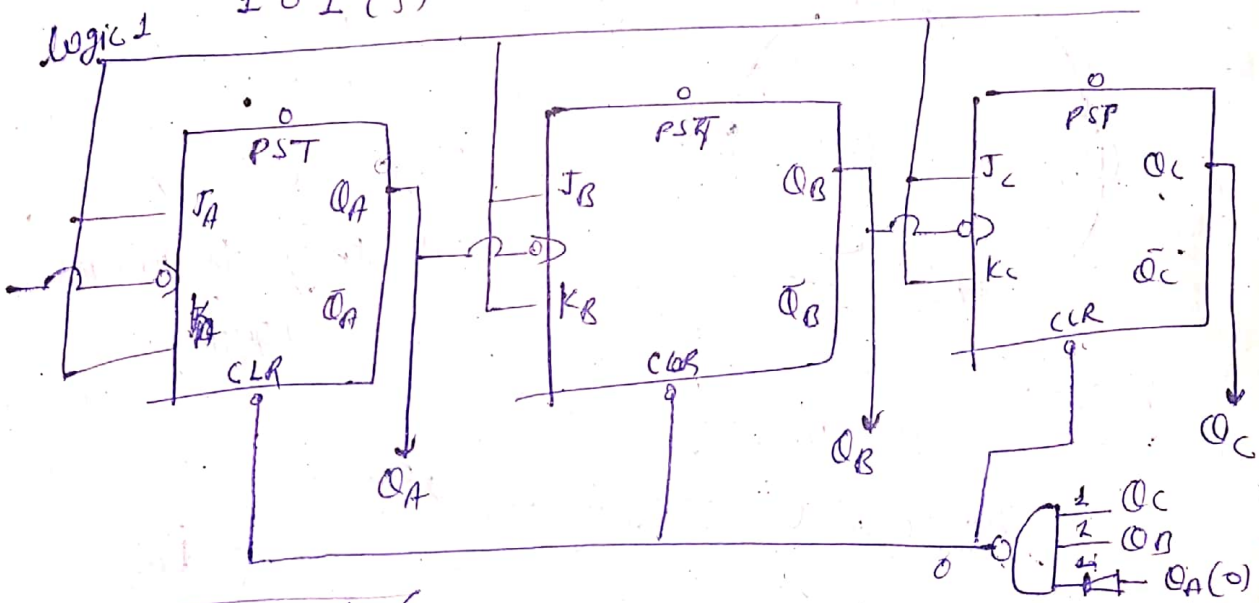
001 (1)

010 (2)

011 (3)

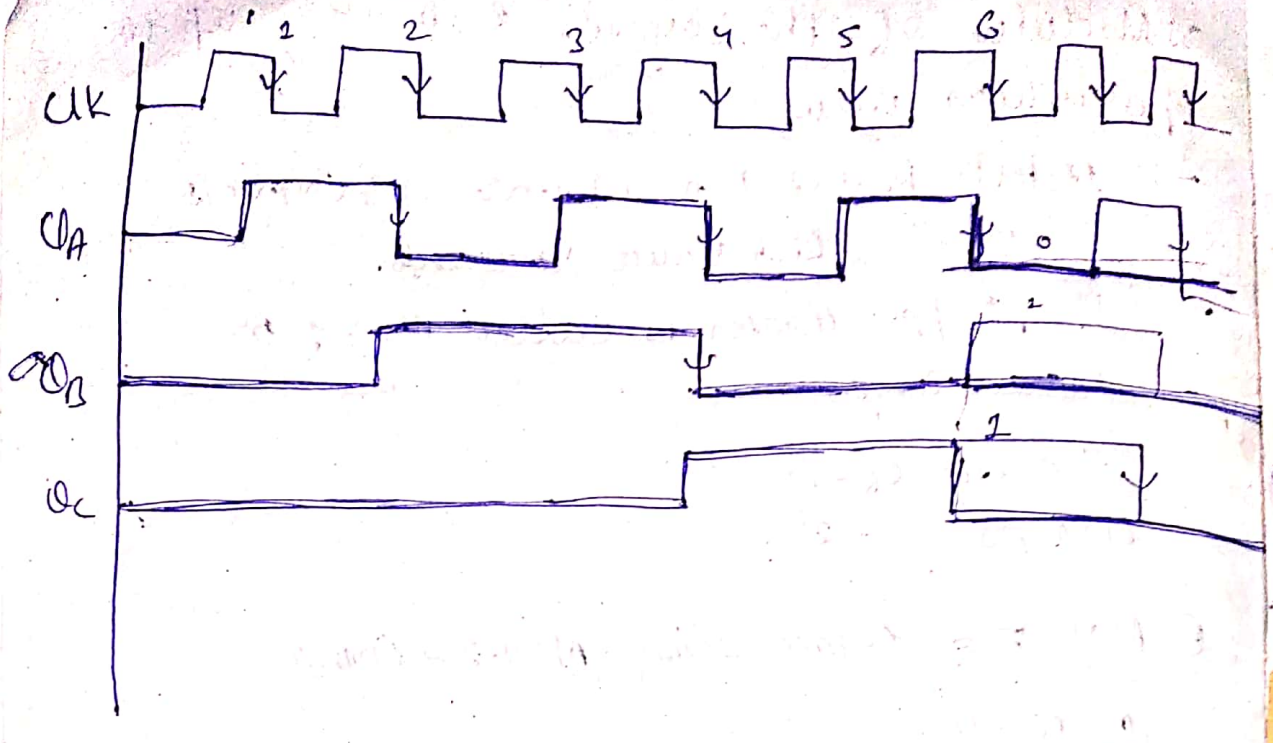
100 (4)

101 (5)



PST = 0 Q = 1

CLR = 0 Q = 0



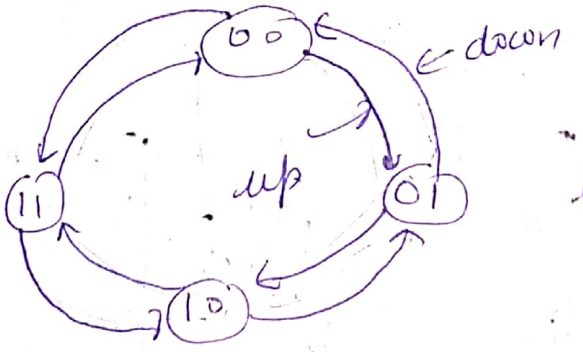
☆ State diagram of a counter

2-bit counter

Max. count → $2^n - 1 = 2^2 - 1$

M-C = 3

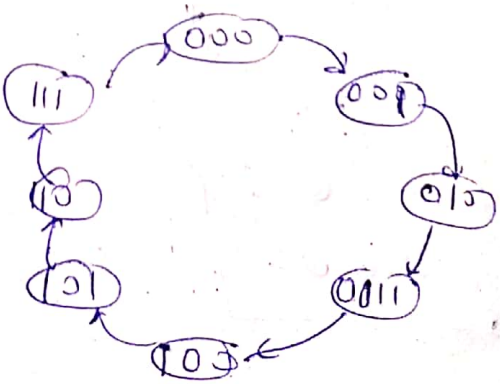
Q_B	Q_A
0	0
0	1
1	0
1	1



in down, state is same only transient is changing

3-bit $M = 2^3 - 1 = 8$

no. of state: $2^3 = 8$



✡ Decade (BCD) Ripple counter

⇒ Negative edge triggered → Q is clock
↓
up counter

⇒ Positive edge triggered → $\bar{Q}$ is clock
up counter

⇒ Negative edge triggered → $\bar{Q}$ is clock →
Down
Counter

⇒ Positive edge triggered → Q is clock
↓
Down Counter

② Mod-M — MOD-N — Cascading.

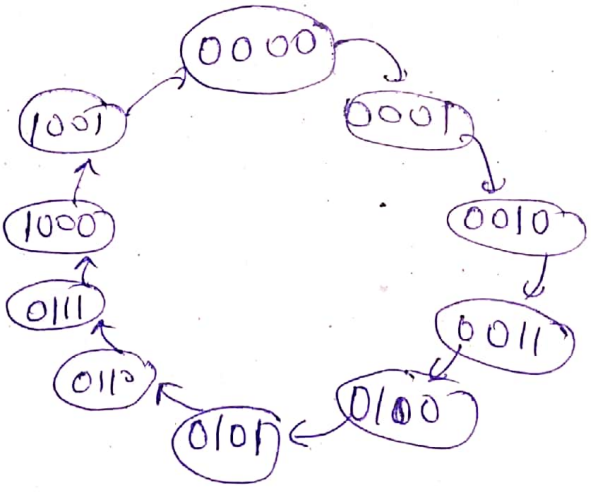
MOD MN

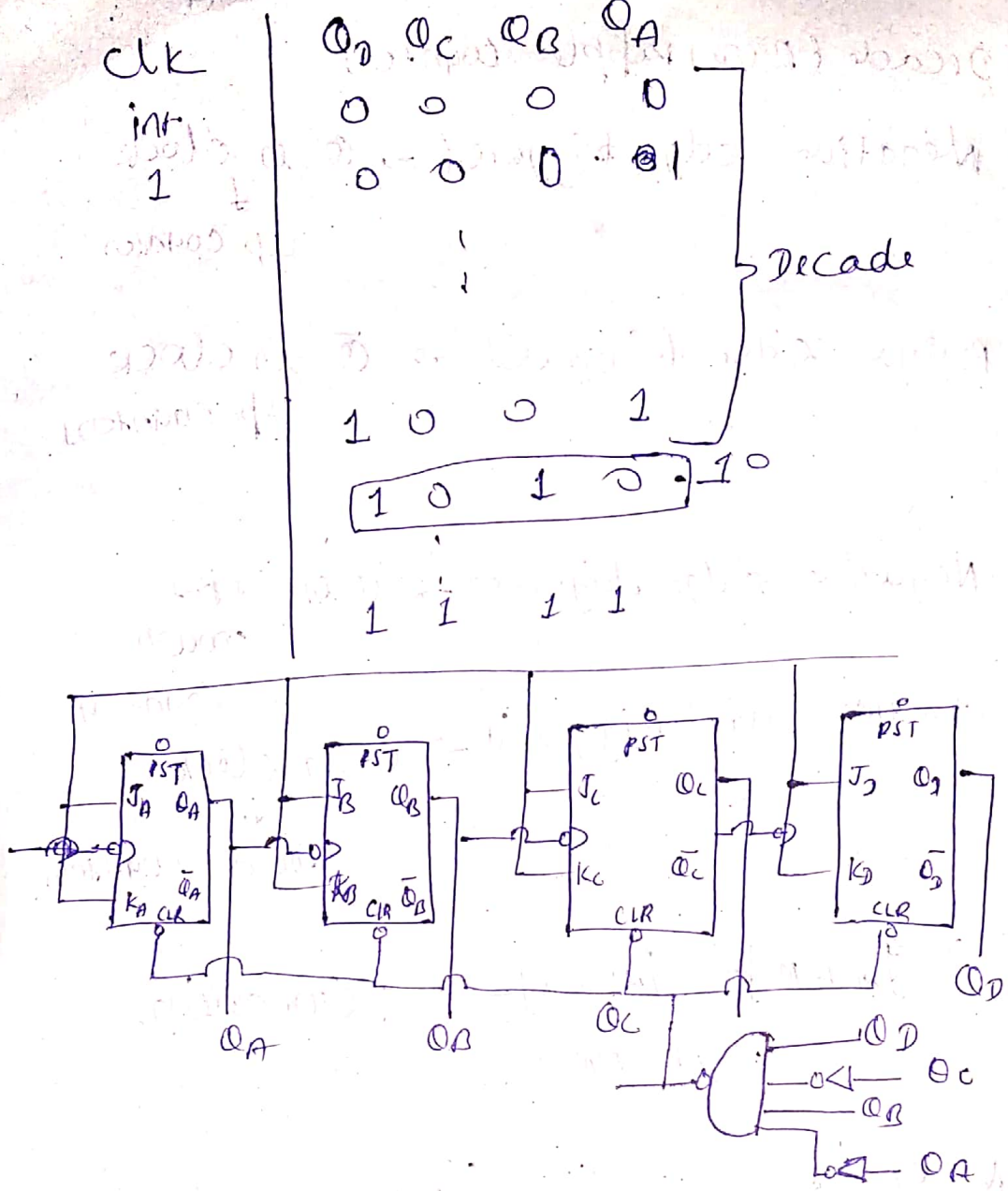
No. of states = 10 (0-9)

∴ 0-9

Max. count : 10-1 = 9

1010





☆ How to design Synchronous counters

Step 1 – Decide the no. of flip flop

Step 2 – Excitation table of FF

Step 3 State diagram and circuit excitation table

Step 4 – Obtained simplified eqn using k-map

Step 5 – Draw the logic diagram

Q Design 2-bit Synchronous up coun

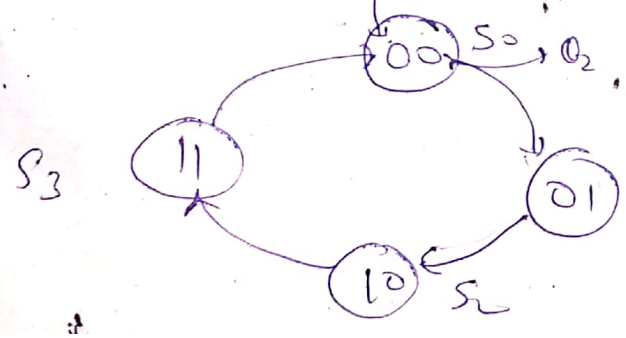
→ 2-bit → 2 flip flop

→ JK flip-flop

Excitation table of J-K flip flop

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

State diagram Q_1 2^n = 4 state max coun 4-1=3



ckt. excitation table

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

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

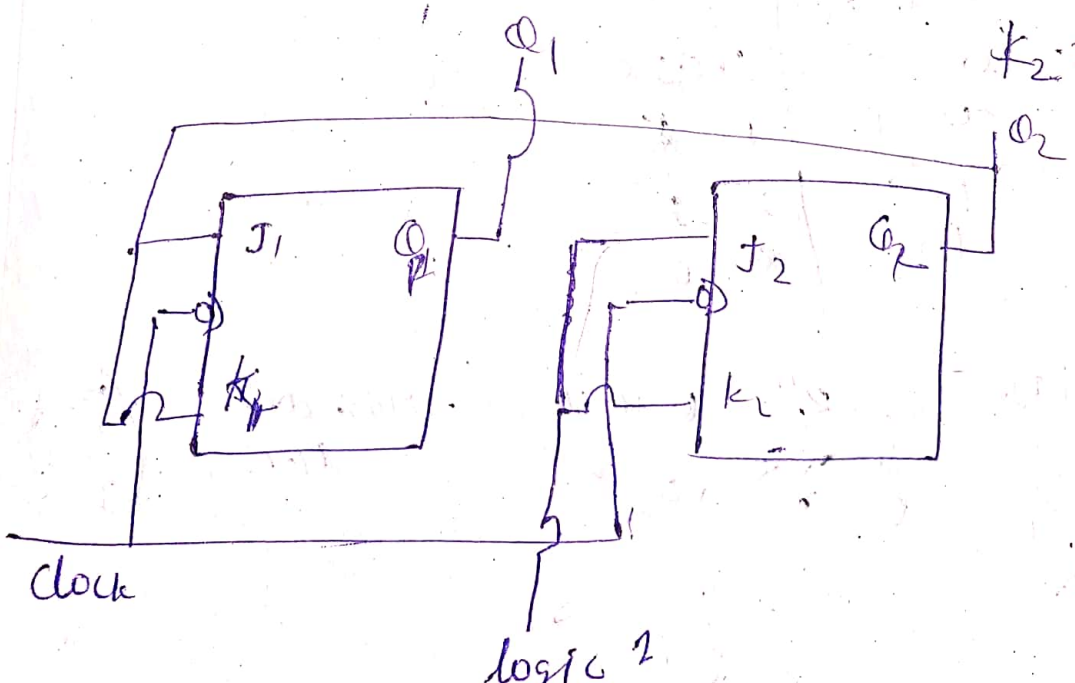
$K_1 = Q_2$

Q_1 \ Q_2	0	1
0	1	x
1	1	x

$J_2 = 1$

Q_1 \ Q_2		
	x	1
	x	1

$K_2 = 1$



★ 3-bit

→ 3 flip

step-2

Ste

☆ 3-bit Synchronous Up Counter

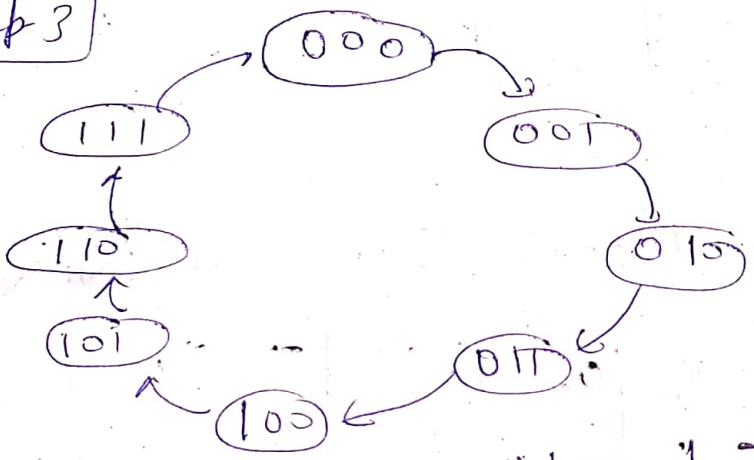
→ 3 flip flop - T flip flop.

step-2

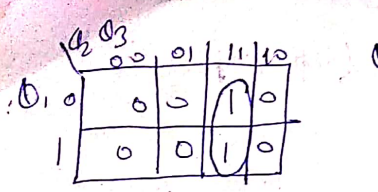
Q_n	Q_{n+1}

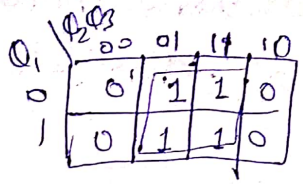
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Step 3



Q_1	Q_2	Q_3	Q_1^*	Q_2^*	Q_3^*	T_1	T_2	T_3
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	0	0	0	1	1	1
1	1	1	0	0	1	1	0	1

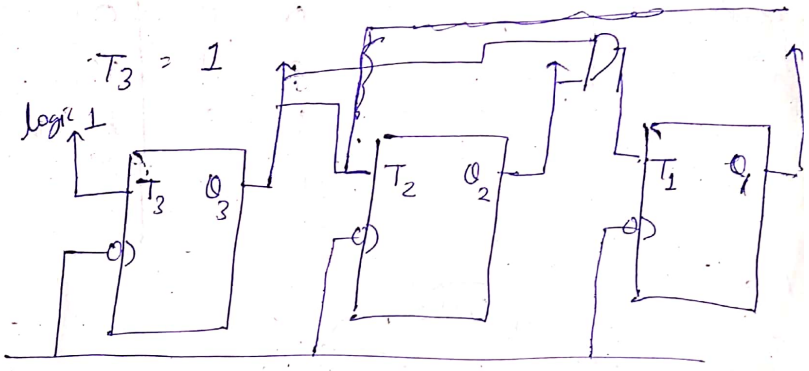




$T_1 = Q_2Q_3$

$T_2 = Q_3$

$T_3 = 1$



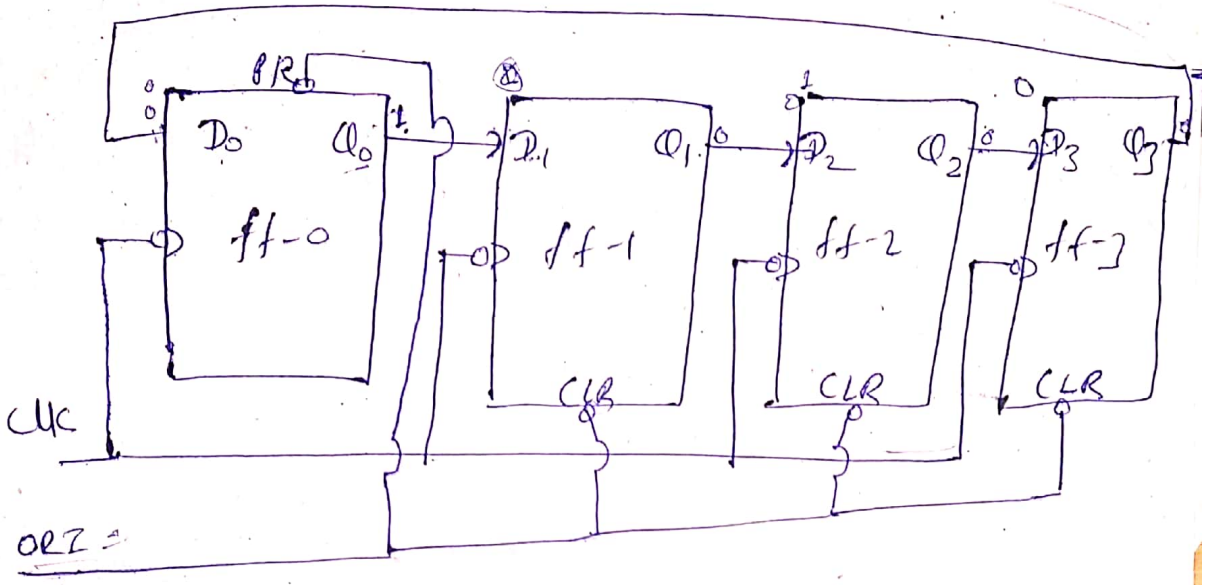
★ 3-bit up/down synchronous counter

M	Q_C	Q_B	Q_A	Q_C^*	Q_B^*	Q_A^*	T_C	T_B	T_A
0	0	0	0	0	0	1	0	0	1
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			
1	0	0	1	0	0	0			
1									

☆ Ring Counter

no. of States = no. of flip flop used

⇒ The only change is the output of last ff is connected to the input of first ff.



PR → Q = 1
CLR → Q = 0

ORI	Clk	Q_0	Q_1	Q_2	Q_3
⊔	×	1	0	0	0
1	↓	0	1	0	0
1	↓	0	0	1	0
1	↓	0	0	0	1
1	↓	1	0	0	0