DIGITAL ELECTRONICS

Boolean Logical Ideas :- These are classified into three types :

1) Producing the constants (0,1) (Null, Identity)
2) Unary operations (Compliment, Transfer)
3) Binary operations (AND, OR, NAND, NOR, EX-OR, Ex-NOR, Inhibition, Implication)

NOTE :- For n- input variables, we get 3^n combinations, 2^{2^n} possible functions

x	y	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
							Buffer	XOR	OR	NOR		NOT					Identity
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		Null	AND	$x\cdot\bar{y}$	x	$\bar{x}\cdot y$	y	Ex-OR		Ex-NOR						NAND	
					Buffer												

$f_0 = 0 \Rightarrow$ Null

$f_1 = x \cdot y$ AND

$x \wedge y$

$f_2 = x \cdot \bar{y}$ Inhibition

x/y [x but not y]

$f_3 = x$ transfer (Buffer)

Implication

$f_4 = \bar{x} \cdot y$ Inhibition

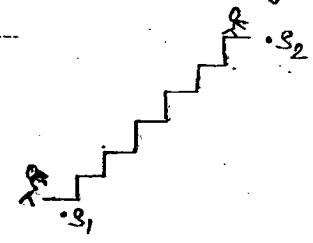
y/x (y but not x)

$f_5 = y$ Transfer

$f_6 = x \oplus y$ Ex-OR

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

NOTE:- The staircase logic gate is Ex-OR gate.



$f = S_1 \oplus S_2$

S_2	S_1	F Bulb
0	0	0
0	1	1
1	0	1
1	1	0

$f_7 = x + y$ OR

$x \vee y$

$f_8 = \overline{x+y}$ NOR

$x \downarrow y$

$f_9 = x \odot y$ Ex-NOR

$= x'y' + xy$

NOTE:- Ex-NOR is also known as coincidence logic gate or Equivalence logic gate.

$f_{10} = \bar{y}$

$f_{11} = x + \bar{y}$ Implication

$x \subset y$ [if y then x]

$f_{12} = \bar{x}$ NOT (complimentary)

$f_{13} = \bar{x} + y$ Implication

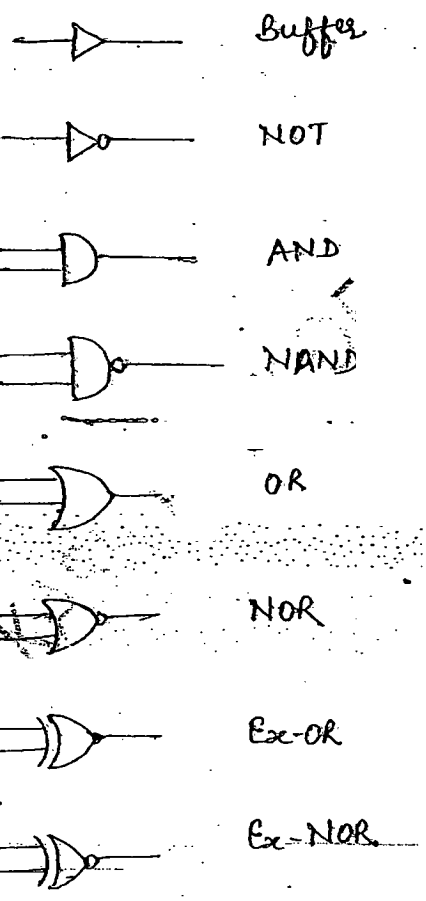
$x \supset y$ [if x then y]

$f_{14} = \overline{x \cdot y}$ NAND

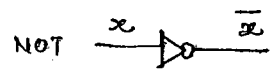
$x \uparrow y$

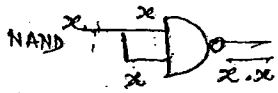
$f_{15} = 1$ Identity

Symbols for the Logic Gate



NAND, NOR - Universal Logic Gates





$x \cdot x = \bar{x} + \bar{x}$

$= \bar{x}$

	NAND (4) NOR	
NOT	1	1
AND	2	3
OR	3	2
Ex-OR	4	5
Ex-NOR	5	4

Duality

Step 1:- Interchange the operator.

Step 2:- Interchange the identity

Interchange

1) (·, +)

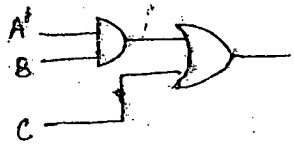
2) (0, 1)

AND	OR
$x \cdot 0 = 0$	$x + 0 = x$
$x \cdot 1 = x$	$x + 1 = 1$
$x \cdot \bar{x} = 0$	$x + \bar{x} = 1$
$x \cdot x = x$	$x + x = x$
$x \cdot 0 = 0$	
$x + 1 = 1$	
$x + x' = 1$	
$x \cdot x' = 0$	

$f = A' \cdot B + C$

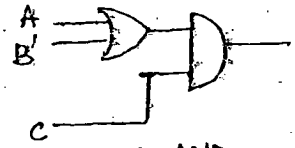
$f^D = ?$

$f^D = (A' + B) \cdot C$



AND-OR

$f = A' \cdot B + C$



OR-AND

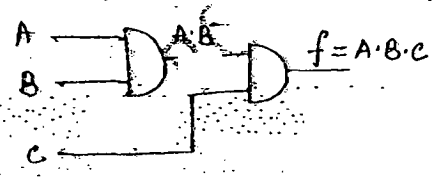
$f^D = (A' + B) \cdot C$

* AND-OR dual form is OR-AND.

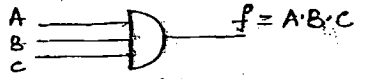
Degenerative forms

When a two level logic gate system output is expressed with a single logic gate then the two level logic gate system is known as degenerated form for the single logic gate.

Ex: AND - AND is the degenerated form for the AND gate.

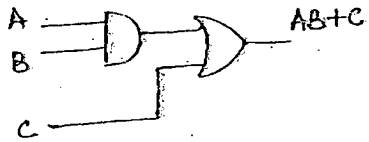


AND-AND

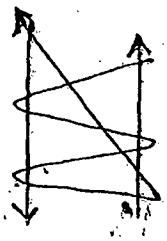


AND

Non-Degenerative forms

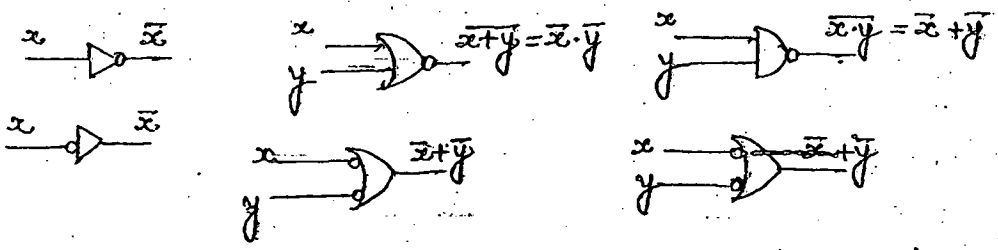


AND-OR



AND – OR	OR – AND
NAND – NAND	NOR – NOR
NOR – OR	NAND – ~~NOR~~ AND
OR – NAND	AND – NOR

Alternative Logic Gates



NAND $\xrightarrow{\text{bubbled}}$ OR

NOR $\xrightarrow{\text{bubbled}}$ AND

AND $\xrightarrow{\text{bubbled}}$ NOR

OR $\xrightarrow{\text{bubbled}}$ NAND

(+ve) and (–ve) Logics

(+ve) Logic	(–ve) Logic
High → 1	High → 0
Low → 0	Low → 1

(+ve) Logic

–2V → 1

–7V → 0

(+ve) AND Logic

A	B	f
0	0	0
0	1	0
1	0	0
1	1	1

(-ve) AND Logic = (+ve) OR Logic

A	B	f
1	1	1
1	0	1
0	1	1
0	0	0

NOTE:- (-ve) AND Logic equal to (+ve) OR Logic, vice versa.

(-ve) NAND Logic equal to (+ve) NOR Logic & vice versa.

Ex-OR, Ex-NOR Shortcuts

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

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

$\overline{x \oplus y} = x \odot y$ [for even]

$x \oplus y \oplus z = x \odot y \odot z$ [for odd]

$\overline{w \oplus x \oplus y \oplus z} = w \odot x \odot y \odot z$

$\bar{x} \oplus y = (\bar{x})y + (\bar{x})y'$
$= xy + x'y'$
$= x \odot y$

$\left.\begin{matrix} \overline{x \oplus y} \\ \bar{x} \oplus y \\ x \oplus \bar{y} \end{matrix}\right\} = x \odot y$

Complementing the Boolean Expression

Step 1: Dual form

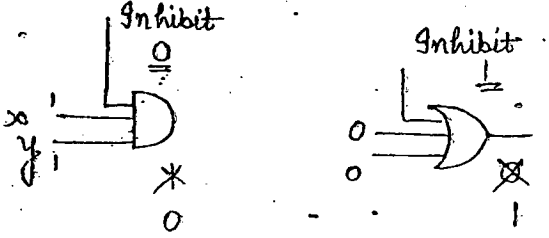
Step-2: Complementing Individual variable

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

$f' = ?$

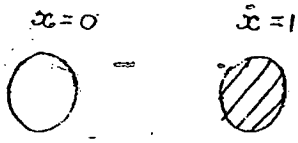
1. $(\bar{A}+B)\cdot C$
2. $(A+\bar{B})\cdot\bar{C} = f'$

Inhibiting the logic gate

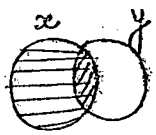


	Inhibit Disable	Enable
NAND	L	H
NOR	H	L
AND	L	H
OR	H	L

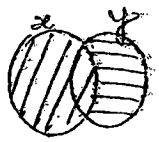
Venn Diagrams

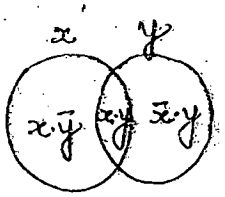


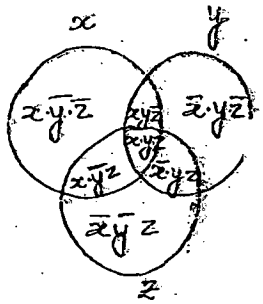
$x + x \cdot y = x$



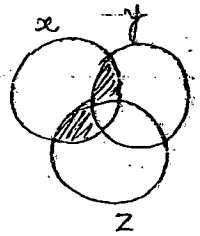
$x + \bar{x}y = x + y$







Q



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

Logic Minimization Techniques

1) Boolean Algebra

NOT

If $x = 0$

$\bar{x} = 1$

$(\bar{x})' = x$

AND

$x \cdot x = x$

$x \cdot 0 = 0$

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

$x \cdot 1 = x$

OR

$x + x = x$

$x + 0 = x$

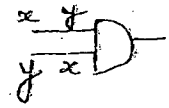
$x + \bar{x} = 1$

$x + 1 = 1$

Commutative law

$x \cdot y = y \cdot x$

$x + y = y + x$



~~Associative La~~

Associative Law

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

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

DeMorgan's Laws

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

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

Distributive Law =

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

Consensus Theorem

If variable is associated with some variable and its complement is associated with some other variable and the next term is formed by the left over variable then the term becomes redundant.

NOTE :- Consensus theorem can be extended to any number of variables.

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

$\Downarrow$ Redundant

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

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

$\Downarrow$ Redundant

Transposition Theorem

Associations and operators can be interchanged.

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

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

Absorption Law

$$x + xy = x$$

$$x \cdot (x+y) = x$$

ORing a variable with ANDing of that variable by another variable results in the same variable.

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

Redundant Literal Rule

$$x + x'y = x + y$$

$$x \cdot (x'+y) = x \cdot y$$

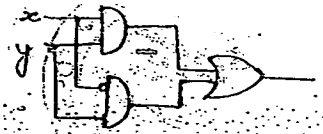
ORing a variable with ANDing of complement with another variable results in the ORing of those two variables.

SOP and POS Forms

SOP

x	y	min term	f
0	0	$\bar{x} \cdot \bar{y}$	0
0	1	$\bar{x} \cdot y$	1
1	0	$x \cdot \bar{y}$	0
1	1	$x \cdot y$	1

$$f = x \cdot y + \bar{x} \cdot y$$



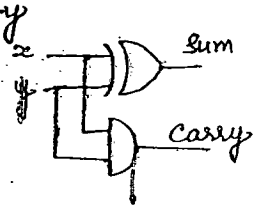
Half Adder

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

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

$$= x \oplus y$$

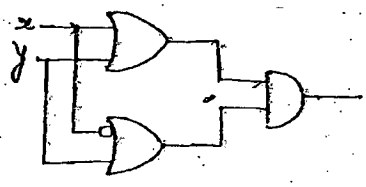
$$carry = xy$$



POS

x	y	Maxterm	f
0	0	$x+y$	0
0	1	$x+\bar{y}$	1
1	0	$\bar{x}+y$	0
1	1	$\bar{x}+\bar{y}$	1

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



$\sum m \Rightarrow SOP$

$\pi M \Rightarrow POS$

$\sum m(0,2) = \bar{x}\cdot\bar{y} + x\cdot\bar{y}$

00 10

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

$\pi M(0,2) = (x+y)\cdot(\bar{x}+y)$

00 10

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

$\sum m(0,2) = \pi M(1,3)$

	x	y	f
1.	0	0	1
2.	0	1	0
3.	1	0	1
4.	1	1	0

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

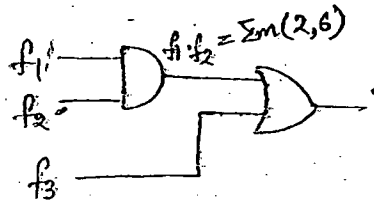
$f = \pi M(1,3)$

$\pi M(1,5) = \sum m(0,2,3,4,6,7)$

$f_1 = \Sigma m(0,1,2,4,6)$

$f_2 = \Sigma m(2,3,5,6,7)$

$f_3 = \Sigma m(1,2,4,7)$



$f = f_1 \cdot f_2 + f_3$

$= \Sigma m(2,6) + \Sigma m(1,2,4,7)$

$= \Sigma m(1,2,4,$

$A = \Sigma m(0,1,2,4,6,7)$

$B = \Sigma m(2,3,4,5,6)$

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

$A'B = \Sigma m(3,5)$

(B but not A)

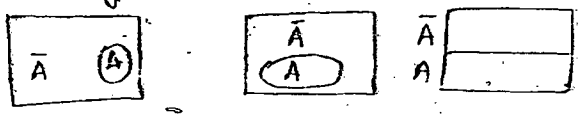
$AB' = \Sigma m(0,1,7)$

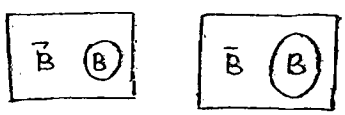
$A'B + AB'$

$A \oplus B = \Sigma m(0,1,3,5,7)$

K-Map

It is a group of adjacent cells. Each cell is represented by a minterm (group of literals).





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

① –

	$\bar{B}$	B
$\bar{A}$	$\bar{A}\bar{B}$ 00 0	$\bar{A}B$ 01 1
A	$A\bar{B}$ 10 2	AB 11 3

② –

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	$\bar{A}\bar{B}\bar{C}$ 000 0	$\bar{A}\bar{B}C$ 001 1	$\bar{A}BC$ 011 3	$\bar{A}B\bar{C}$ 010 2
A	$A\bar{B}\bar{C}$ 100 4	$A\bar{B}C$ 101 5	ABC 111 7	$AB\bar{C}$ 110 6

$f(ABC)$ — MSB LSB

③ –

$f(ABC)$ — MSB LSB

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	$\bar{A}\bar{B}\bar{C}$ 000 0	$\bar{A}B\bar{C}$ 010 2	$AB\bar{C}$ 110 6	$A\bar{B}\bar{C}$ 100 4
C	$\bar{A}\bar{B}C$ 001 1	$\bar{A}BC$ 011 3	ABC 111 7	$A\bar{B}C$ 101 5

④ –

$f(CAB)$ — MSB LSB

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	0	1	3	2
C	4	5	7	6

⑤ – $f = \sum m(0, 1, 2, 6)$ → 000, 001, 010, 110

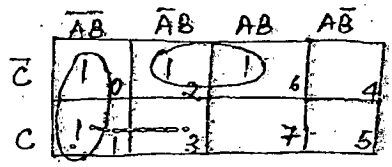
$$f = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C}$$

⑤.1 –

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1 (0)	1 (1)	3	1 (2)
A	4	5	7	1 (6)

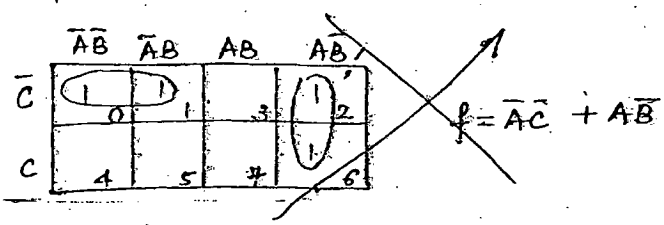
$$f = \bar{A}\bar{B} + B\bar{C}$$

(5.2)



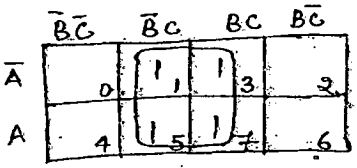
$f = \bar{A}\bar{B} + B\bar{C}$

(5.3)



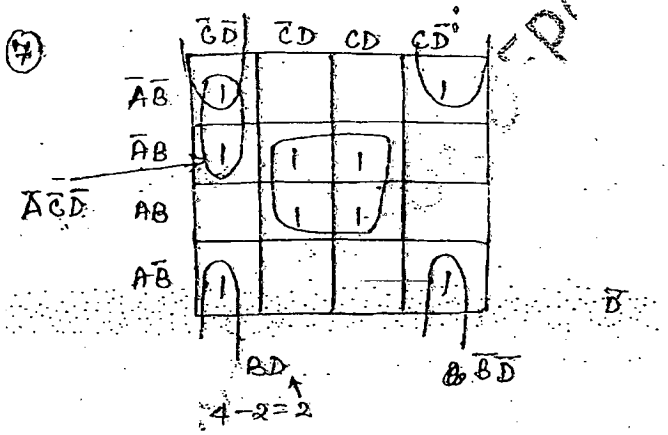
(6) $f = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + ABC$

(6.1)



$f = C$

(7)



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

(8)

pair	1
Quad	2
Octet	3

(9)

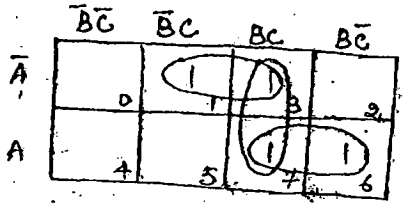
K = 1 all cells '1'

No need CKT

Redundant Group

It is that part of the circuit with or without it there is no effect on the circuit result.

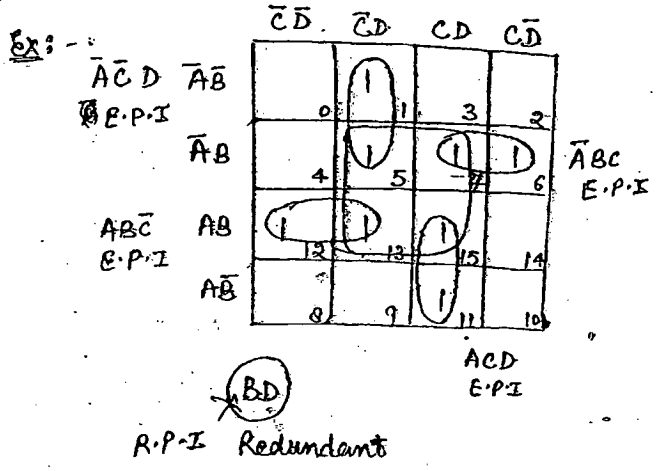
Ex:-



$f = \Sigma m(1,3,5,7)$

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

$\bar{A}C$ E·P·I, AB E·P·I, BC R·P·I Redundant



Implicant

It is the min term corresponding to that cell which is having "1" in the K-map.

Prime Implicant: All possibility groupings in the K-map are known as prime implicants.

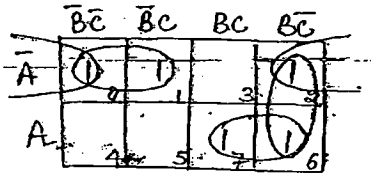
Essential Prime Implicant: It is the prime implicant which is having atleast a single one which is only one time group known as essential prime implicant.

Redundant Prime Implicant: It is the prime implicant with or without it, there is no effect on the circuit result.

Selective Prime Implicant: This is a prime implicant which is under the process of selection.

NOTE: If the grouping was done by zeros then we get false implicant, false prime implicant, false E.P.I etc.

Ex 1)



$f = \Sigma m(0,1,2,6,7)$

$\therefore$ $\bar{A}\bar{B}$ (0,1) E.P.I ✓

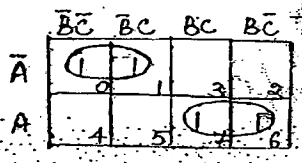
$B\bar{C}$ (2,6) ~~E.P.I~~ P.I
$\bar{A}\bar{C}$ (0,2) ~~E.P.I~~ P.I
} S.P.I

AB (6,7) E.P.I

Imp → 5 P.I → 4

Non P.I → 0 E.P.I → 2

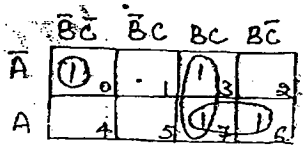
2)



$\bar{A}\bar{B}$ E.P.I P.I

AB E.P.I P.I

3)



$\bar{A}\bar{B}\bar{C}$ - BC E.P.I P.I

AB E.P.I P.I

Imp → 4

Non - P.I → 1

P.I → 2

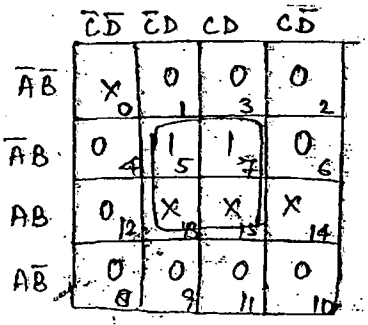
E.P.I → 2

Don't Care Condition

The unoccured inputs corresponding output which is unspecified can be treated as don't care.

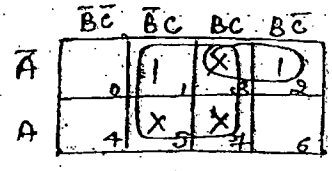
Q Design a circuit which gives the alarm at 5'o clock and 7'o clock.

A	B	C	D		
0	0	0	0	0	X
				1	0
				2	0
				3	0
				4	0
0	1	0	1	5	1
				6	0
0	1	1	1	7	1
				8	0
				9	0
				10	0
				11	0
				12	0
1	1	0	1	13	X
1	1	1	0	14	X
1	1	1	1	15	X



$f = BD$

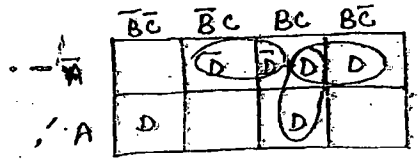
Ex: $f = \Sigma m(1,2) + d(3,5,7)$



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

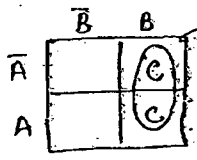
Variable Entered Mapping

Ex:1) $f = \bar{A}\bar{B}C\bar{D} + \bar{A}BCD + \bar{A}BC\bar{D} + \bar{A}B\bar{C}D + ABCD + A\bar{B}\bar{C}D$

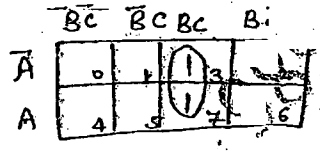


$f = \bar{A}C\bar{D} + \bar{A}BD + BCD + AB\bar{C}D$

2)



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



$f = BC$

Step 1 : Keep all the variables as zeros and get the expression.

NOTE :- Check whether the given 1 is fully covered. If it is fully covered it becomes redundant.

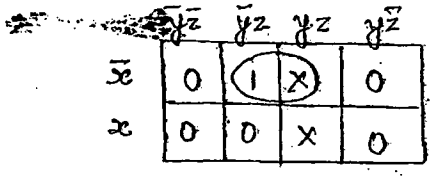
Step 2 : Select any variable and keep 1 in it, other variable should be kept as zero, given 1 should be replaced by don't care.

Step 3 :- Repeat the above process by selecting each variable.

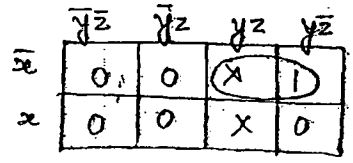
	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{x}$	0	A	$A+\bar{A}$	$\bar{A}$
x	B	0	X	C

	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{x}$	0	0	1	0
x	0	0	X	0

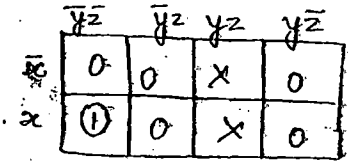
$(\bar{y}z)$



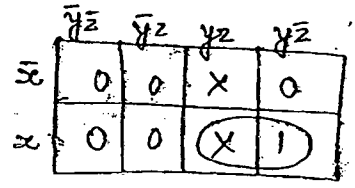
$A\bar{x}z$



$\bar{A}\bar{x}y$



$Bx\bar{y}\bar{z}$



Cxy

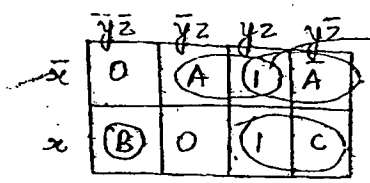
$f = yz + A\bar{x}z + \bar{A}xy + Bx\bar{y}\bar{z} + Cxy$

Redundant

AB ✗ $A\bar{A}$ ✗

(11) (1X) (AA) ($A\bar{A}$) (AX) ($\bar{A}x$) ($\bar{A}1$) ($\bar{A}1$)

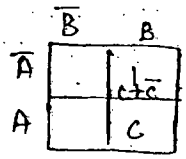
Ex:



→ fully-covered

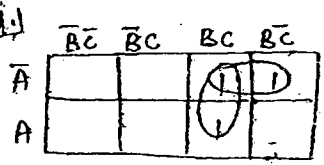
$f = A\bar{x}z + \bar{A}xy + Cxy + Bx\bar{y}\bar{z}$

Ex



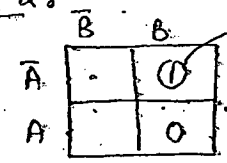
$f = \bar{A}B + ABC$

Method:



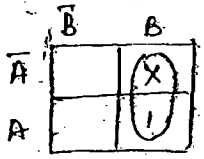
$f = \bar{A}B + BC$

Method-2:



→ NOT fully covered

$\bar{A}B$

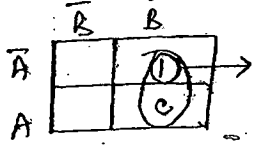


CB

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

Method-3: (Shortcut)

Make the pair & check whether 1 fully covered or not. Keeping all variables zero - make pair using 1's. If not then by



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

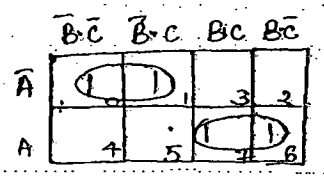
POS forms in K. Maps

Step 1: Interchange the operator

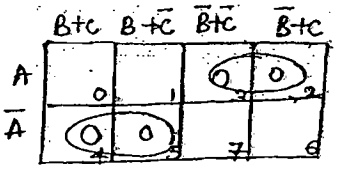
Step 2: Compliment the individual variable

Ex: $f = \pi M(2,3,4,5)$

$f = \Sigma m(0,1,6,7)$



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

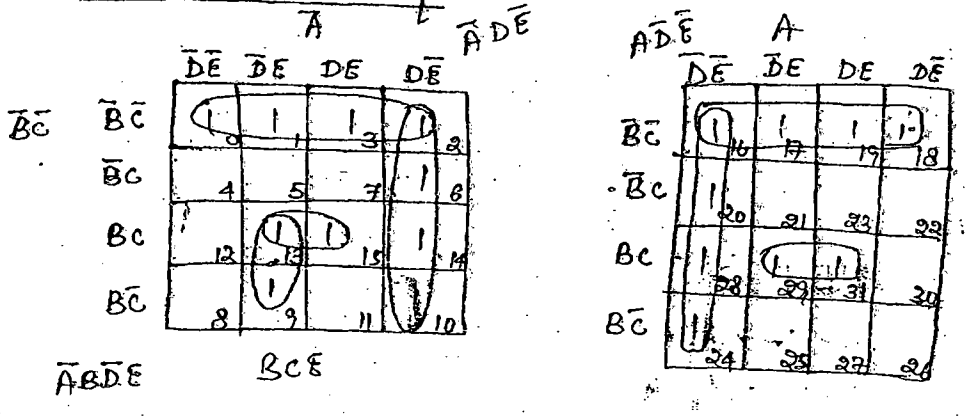


$f = (A+\bar{B}).(\bar{A}+B)$

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

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

5 Variable K-Map



* Grouping possible only if there is one variable change $\bar{A}D\bar{E}$ and $A\bar{D}\bar{E}$ (two variable change)

Implementation of Boolean Expressions

Case 1: By NAND gates

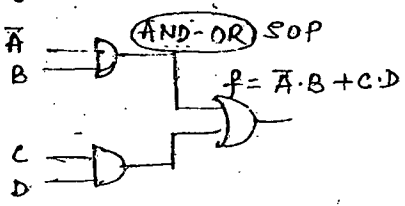
All the terms should be product terms. To get this, do the double complementation at every + operator.
(जहाँ जहाँ '+' है, वहाँ वहाँ बार बार करो)

Case 2: By NOR gates

All the terms should be sum terms. To get this, do the double complementation at every '.' operator.
(जहाँ जहाँ '.' है, वहाँ वहाँ बार बार करो)

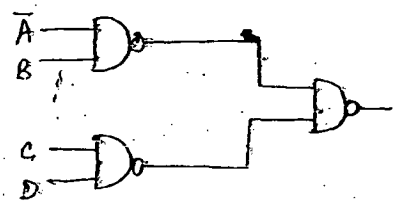
NAND

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



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

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

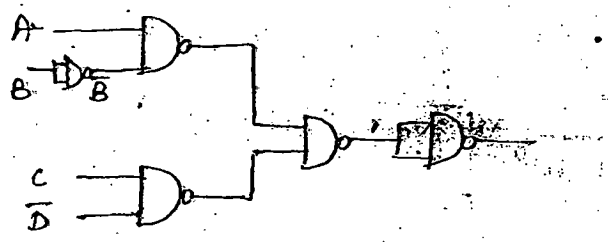


$f = (\bar{A} + B) \cdot (\bar{C} + D)$ POS

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

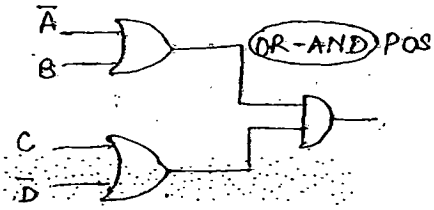
$$= \overline{\overline{\bar{A} \cdot B} \cdot \overline{\bar{C} \cdot D}}$$

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



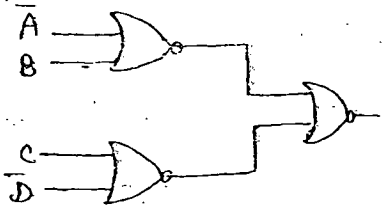
NOR

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



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

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

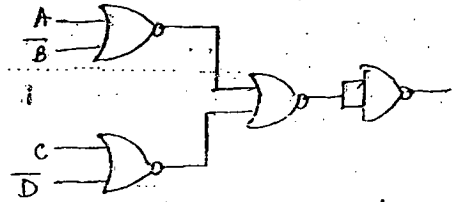


$f = \bar{A} \cdot B + \bar{C} \cdot D$

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

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

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



* When NOT gate not given use NAND gate for NOT operation.

Mc- Clusky

Step 1 : Consider the circuit in the form of SOP.

Step 2 : Form different groups depending on no. of 1's in the K-map.

Step 3 : Compare the successive groups for one literal change.

Step 4 : Repeat the above process until we get all different groups (states)

Ex $f = \Sigma m(0, 1, 3, 7, 8, 9, 11, 14, 15)$

Implicants

Group	Minterm	Variable ABCD
0	0	0 0 0 0
1	1	0 0 0 1
	8	1 0 0 0
2	3	0 0 1 1
	9	1 0 0 1
3	7	0 1 1 1
	11	1 0 1 1
	14	1 1 1 0
4	15	1 1 1 1

Pairs

Group	Minterm	Variable ABCD
0	0, 1	0 0 0 –
	0, 8	– 0 0 0
1	1, 3	0 0 – 1
	1, 9	– 0 0 1
	8, 9	1 0 0 –
2	3, 7	0 – 1 1
	3, 11	– 0 1 1
	9, 11	1 0 – 1
3	7, 15	– 1 1 1
	11, 15	1 – 1 1
	14, 15	1 1 1 –

↓ ABC

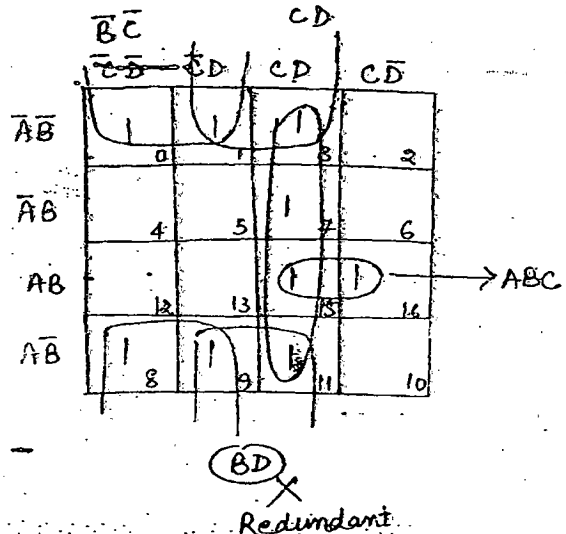
Quads

Group	Min term	Variable ABCD	
0	0, 1, 8, 9	– 0 0 –	$\bar{B}\bar{C}$
	0, 8, 1, 9	– 0 0 –	
1	1, 3, 9, 11	– 0 – 1	$\bar{B}D$
	1, 9, 3, 11	– 0 – 1	
2	3, 7, 11, 15	– – 1 1	CD
	3, 11, 7, 15	– – 1 1	

$$f = \bar{B}\bar{C} + \bar{B}D + CD + ABC$$

Redundant

	0	1	3	7	8	9	11	14	15
E.P.I $\bar{B}\bar{C}$	⊗	×			⊗	×			
R.P.I $\bar{B}D$		×	×			×	×		
E.P.I CD			×	⊗			×		×
E.P.I ABC								⊗	×



SOP and SSOP

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

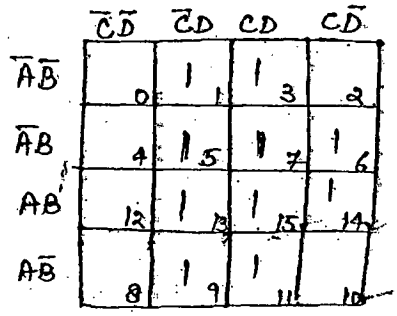
$$f = \bar{A}BC + \bar{A}\bar{B}(C+\bar{C}) + (A+\bar{A})BC$$

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

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

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1 (0)	1 (1)	1 (3)	(2)
A	(4)	(5)	1 (7)	(6)

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



Pg-40 Problems.

Section-1: Boolean Operations and Expressions

1) $f = A + B + C + D$

2) $f = ABCDE$

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

4) (a) $0+0+1 = 1$ (b) $1+1+1 = 1$ (c) $1 \cdot 0 \cdot 0 = 0$

(d) $1 \cdot 1 \cdot 1 = 1$ (e) $1 \cdot 0 \cdot 1 = 0$ (f) $1 \cdot 1 + 0 \cdot 1 \cdot 1 = 1 + 0 = 1$

5) product term = 1, sum term = 0

(a) $AB = 1$
for $A = 1, B = 1$

(b) $A\bar{B}C = 1$
for $A = 1, B = 0, C = 1$

(c) $A + B = 0$
for $A = 0, B = 0$

(d) $\bar{A} + B + \bar{C} = 0$
for $A = 1, B = 0, C = 1$

(e) $\bar{A} + \bar{B} + C = 0$
for $A = 1, B = 1, C = 0$

(f) $\bar{A} + B = 0$
for $A = 1, B = 0$

g) $A\bar{B}\bar{C} = 1$
for $A = 1, B = 0, C = 0$

6) (a) $X = (A+B)C + B$

A	B	C	(A+B)	(A+B)C	X = (A+B)C+B
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	1	1

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

A	B	C	$\overline{(A+B)}$	$X = (\overline{A+B})C$
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

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

A	B	C	$A\bar{B}C$	AB	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	1	1	1

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

A	B	(A+B)	$(\bar{A}+B)$	X
0	0	0	1	0
0	1	1	1	1
1	0	1	0	0
1	1	1	1	1

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

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

Section-2 Laws and Rules of Boolean Algebra

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

Commutative Law

(b) $AB\bar{C}D + \overline{ABC} = D\bar{C}BA + \overline{C}\overline{BA}$

Commutative Law

(c) $AB(CD + E\bar{F} + GH) = ABCD + ABE\bar{F} + ABGH$

Distributive Law

8) a) $\overline{\overline{AB+CD}} + \overline{EF} = AB + CD + \overline{EF}$

By rule $\bar{\bar{A}} = A$

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

$\because A\bar{A} = 0 \quad \therefore \quad 0 + 0 + AB\bar{C} = AB\bar{C}$

c) $A(BC + BC) + AC = A(BC) + AC$

$\because A + A = A$

$\therefore A(BC) + AC$

d) $AB(C+\bar{C}) + AC = AB + AC$

$\because C + \bar{C} = 1$

$\therefore AB + Ac$

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

$\because A + AB = A$

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

f) $ABC + \bar{A}B + \overline{ABCD} = ABC + \bar{A}B + D$

$\because A + \bar{A}B = A + B$ (1st & 3rd term)

Section-3 De Morgan's Theorems

9)

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

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

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

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

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

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

(g) $\overline{AB+CD} = \overline{(AB)}\cdot\overline{(CD)} = (\bar{A}+\bar{B})\cdot(\bar{C}+\bar{D})$

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

10)

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

(b) $\overline{\bar{A}B(CD+\bar{E}F)} = \bar{A}B+\overline{(CD+\bar{E}F)} = \bar{A}B+\bar{C}\bar{D}\cdot\bar{E}F = \bar{A}+\bar{B}+(\bar{C}\bar{D})(\bar{E}F)$

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

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

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

(e) $\overline{AB(CD+\bar{E}F)(\bar{A}B+\bar{C}D)} = \overline{(\bar{A}B)}+\overline{(CD+\bar{E}F)}+\overline{(\bar{A}B+\bar{C}D)}$

$= AB+(\bar{C}D\cdot\bar{\bar{E}}\bar{F})+(AB\cdot CD)$

$= AB+\{CD\cdot(E+\bar{F})\}+(ABCD)$

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

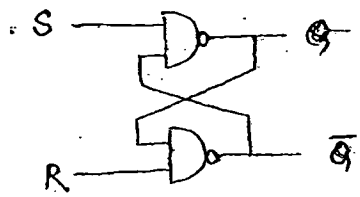
SEQUENTIAL CIRCUITS

* Logic gates with feedback connections known as Sequential circuits.
* In this, the present output not only depends on the present input, it also depends on past output.

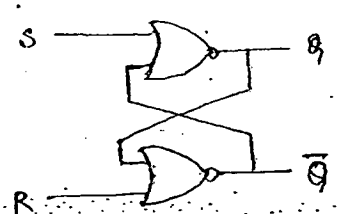
Ex: Latches

* Latches are two types:-

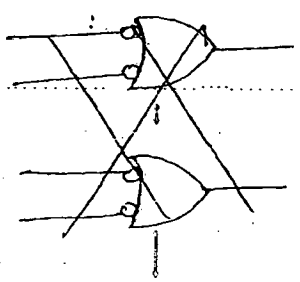
1) NAND Gate Latch (~~Active Low~~)

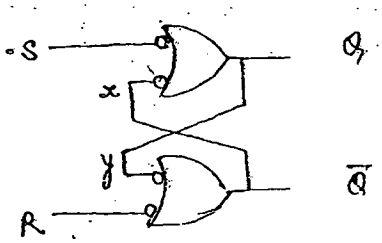


2) NOR Gate Latch (Active High)



Operation of S-R Flip Flop (Active Low)





Case 1: $S=1, R=1$

If Initially, $Q=0$, then the feedback inputs will be $x=1$ and $y=0$, then the next state will be $Q^+=0$.

If initially $Q=1$, then the feedback inputs will be $x=0, y=1$ then the next state will be $Q^+=1$.

So, by the observation we can say that when $S=1, R=1$ we get Q^+ = No change (NC).

Case 2: $S=1, R=0$

Initially if $Q=0$, then the feedback inputs will be $x=1, y=0$ then the next state will be $Q^+=0$.

Initially if $Q=1$, then the feedback inputs will be $x=0$ and $y=1$, then we get $Q=\bar{Q}=1$ only which is an intermediate state, it is not the stable state. Feedback inputs should be operated until we get the stable state.

By the observation, we can say that when $S=1, R=0$ we get $Q^+=0$, the reset state.

Case 3: $S=0, R=1$

In the same above manner, the procedure should be conducted for initially $Q=0$ and initially $Q=1$.

By the observation, we get when $S=0, R=1$, $Q^+=1$ the set condition.

Case 4: $S=0, R=0$

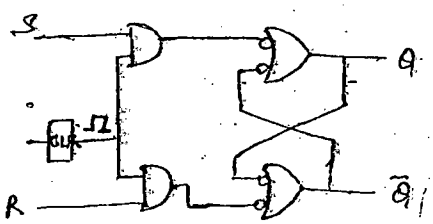
In this case we get $Q^+=\bar{Q}^+=1$ only which is an unused state also known as don't care condition.

NOTE: * The S-R latch can be controlled by using a switch known as Gated S-R latch.

* The controlling can also be done by using clock generator known as clocked S-R flip flop.

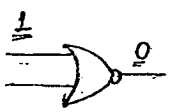
Active Low

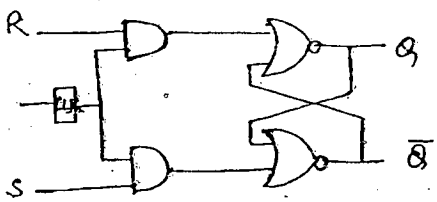
CLK	S	R	Q^+
⊓	1	1	NC
⊓	1	0	0 Reset
⊓	0	1	1 Set
⊓	0	0	x Don't Care



Active High

CLK	S	R	Q^+
⊓	0	0	NC
⊓	1	0	1 Set
⊓	0	1	0 Reset
⊓	1	1	x Don't Care





J-K Flip Flop

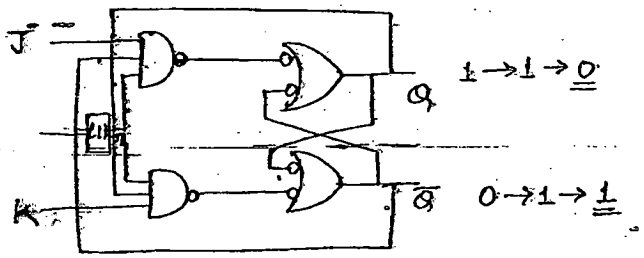
The modification of S-R flip flop with external feedback connections is known as J-K flip flop.

When J = 1, K = 1 ..... and clock pulse is applied, by the observations we can say that $Q_n^+ = \bar{Q}$, toggle condition

NOTE: * Internal feedback should be operated until we get stable output.

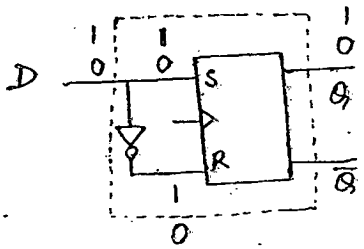
* External feedbacks should be operated only one time for a single clock pulse.

CLK	J	K	Q_n^T
⊓	0	0	NC
⊓	1	0	1 Set
⊓	0	1	0 Reset
⊓	1	1	$\overline{Q}$ Toggle

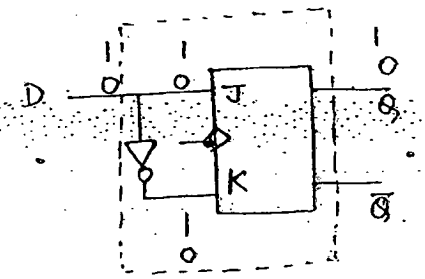


$Q \rightarrow Q^+$
$0 \rightarrow 1$
$1 \rightarrow 0$

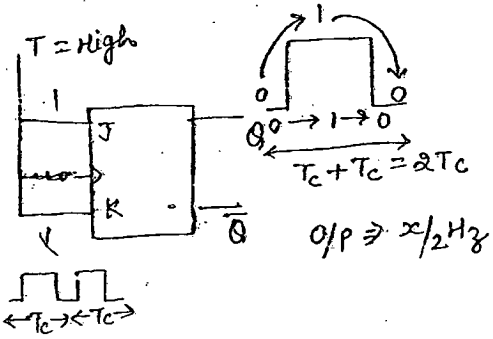
D- Flip Flop



CLK	D	Q^+
⊓	0	1
⊓	0	1



T- Flip Flop

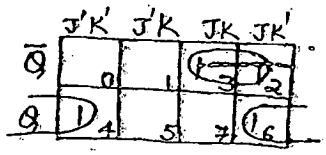


CLK	T	Q^+
⊓	0	Q NC
⊓	1	$\overline{Q}$ Toggle

Characteristic Equations of Flip Flop

J-K flip flop

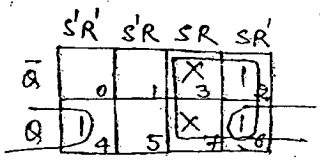
Q	J	K	Q^+	
0	0	0	0	NC
0	0	1	0	Reset
0	1	0	1	set
0	1	1	1	$\bar{Q}$ Toggle
1	0	0	1	
1	0	1	0	
1	1	0	1	
1	1	1	0	



$$Q^+ = JQ' + K'Q$$

S-R flip flop

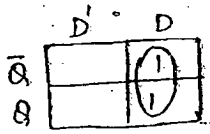
Q	S	R	Q^+	
0	0	0	0	NC
0	0	1	0	Reset
0	1	0	1	Set
0	1	1	x	Don't care
1	0	0	1	
1	0	1	0	
1	1	0	1	
1	1	1	x	



$$Q^+ = S + R'Q$$

D flip flop

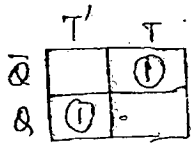
Q	D	Q^+
0	0	0
0	1	1
1	0	0
1	1	1



$$Q^+ = D$$

T-flip flop

Q	T	Q^+
0	0	0
0	1	1
1	0	1
1	1	0



$$Q^+ = T'Q + TQ'$$

$$Q^+ = T \oplus Q$$

Excitation Tables

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

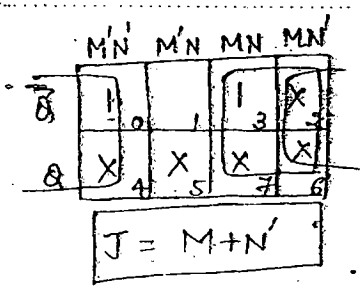
Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

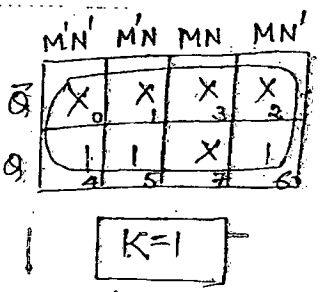
Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

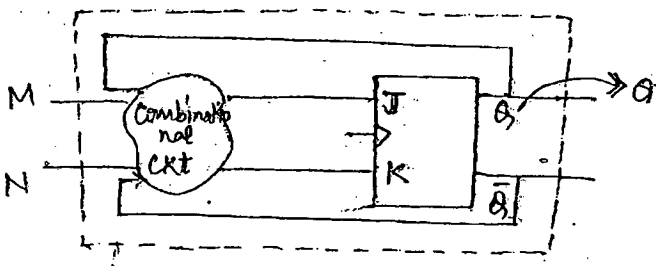
Designing of flip flops

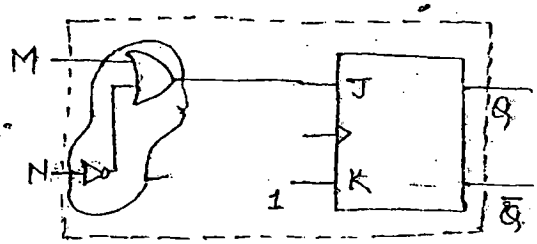
Q	M	N	Q^+	J	K
0	0	0	1	1	X
0	0	1	0	0	X
0	1	0	X	X	X
0	1	1	1	1	X
1	0	0	0	X	1
1	0	1	0	X	1
1	1	0	0	X	1
1	1	1	X	X	X

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0



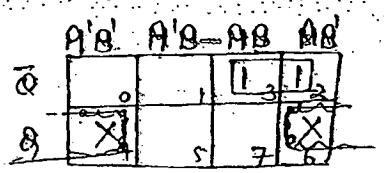


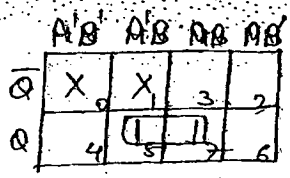




Q	J (A)	K (B)	Q^+	S	R
0	0	0	0	0	X
0	0	1	0	0	X
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	1	X	0
1	0	1	0	0	1
1	1	0	1	X	0
1	1	1	0	0	1

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0





~~$\bar{Q}A$~~

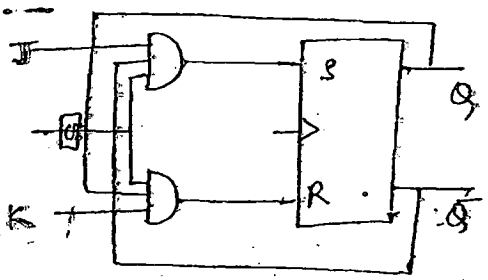
$S = A\bar{Q}$

$J\bar{Q}$

$R = BQ$

KQ

A	B	Q^+
0	0	NC
0	1	Reset
1	0	Set
1	1	$\bar{Q}$ Toggle



Using (SR) → JK (truth table) (JK) → SR $T \rightarrow JK$ $D \rightarrow SR$

	T	T	SR	JK
	D	D	D	T

Race around Problem

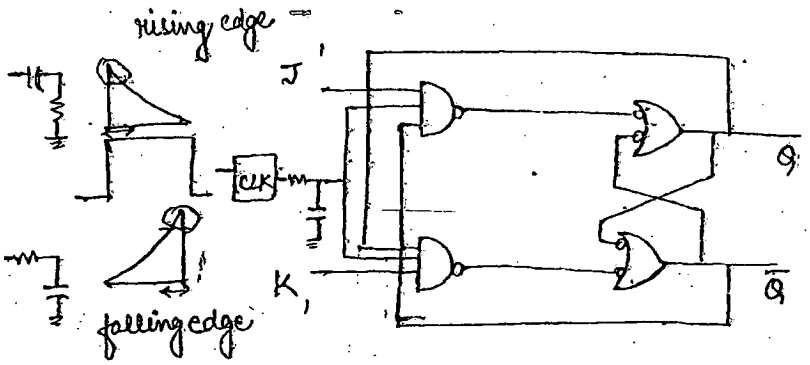
When J=1, K=1 and clock pulse is applied, we get toggle. but if the output occurence time input clock pulse in the ON state then there is a repetition of toggle for a single clock pulse at the input knows as Race around Problem.

Methode to avoid Race around problem

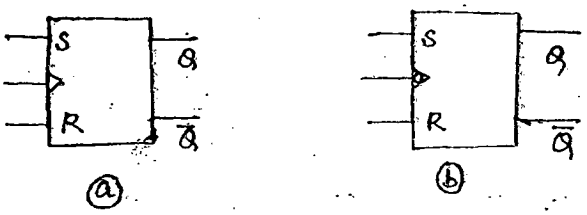
1) Reducing the pulse width
2) By using the edge triggering.
3) By using Master-Slave J-K flip flop.

NOTE:- Edge triggering is of two types:

1) Positive edge triggering (Leading edge triggering, Rising edge triggering)
2) Negative edge triggering (Trailing edge triggering, Failing edge triggering)

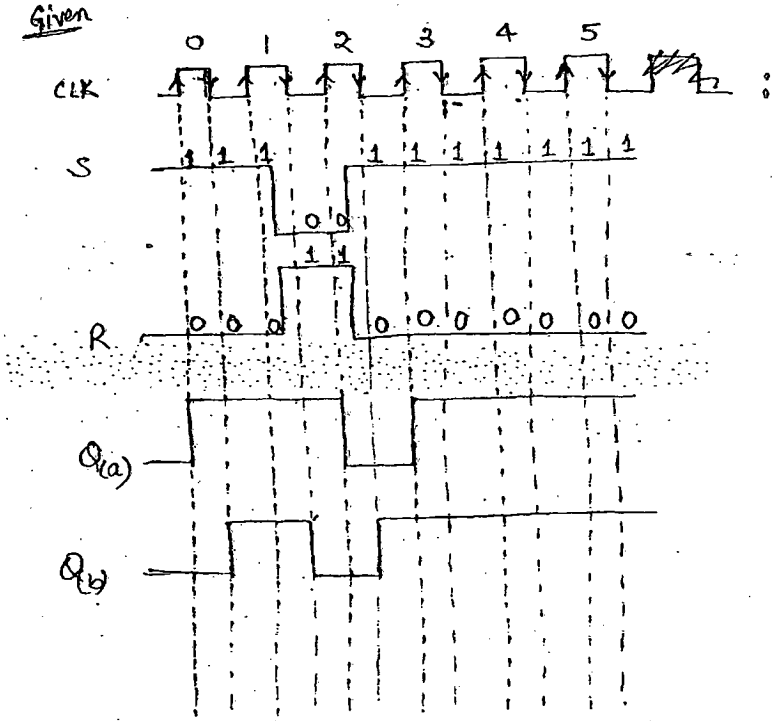


Edge Triggering

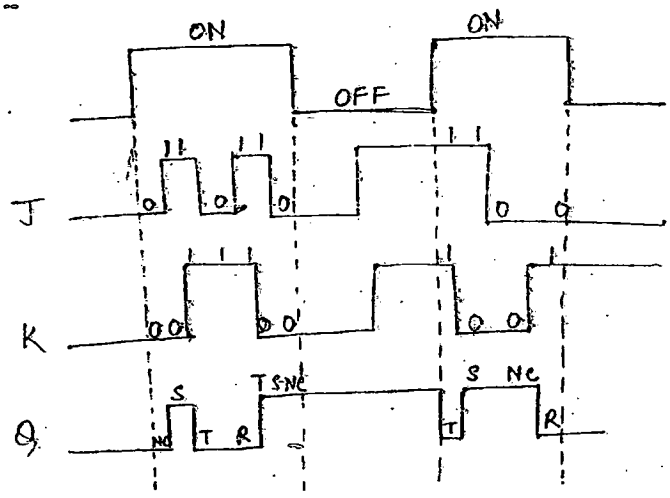


ⓐ ⓑ

Given



Pulse Triggering

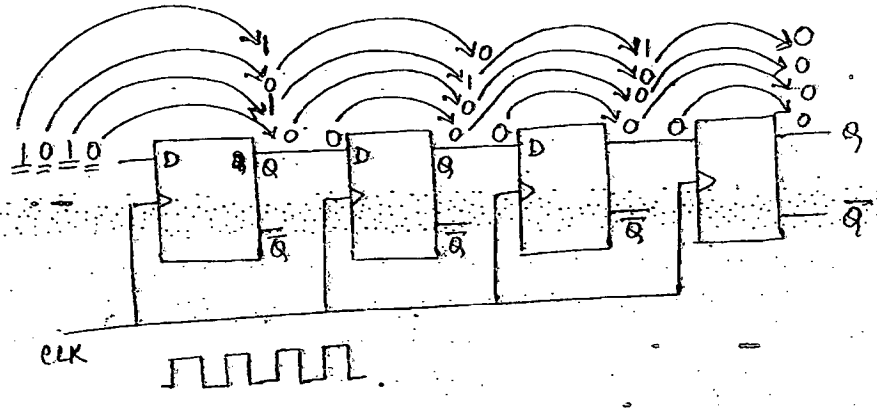


Binary Shift Register

Flip flops connected in the cascaded manner.

A → 1 0 1 0

1) Keep I/P.
2) Apply ⊓
3) Check O/P



Types of Shift Register

SISO (Serial Input Serial Output) ⇒ n and $(n-1)$

SIPO (Serial Input Parallel Output) ⇒ n and 0

PISO (Parallel Input Serial Output) ⇒ 1 and $(n-1)$

PIPO (Parallel Input Parallel Output) ⇒ 1 and 0

Application of Shift Register

1) Used for Temporary data storage.

$$\text{Time Delay}(\Delta t) = N \cdot T_c = N \cdot \frac{1}{f_c}$$

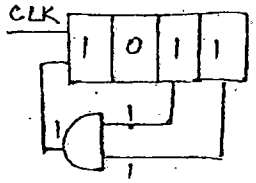
$T_c \rightarrow$ clock period ; $N \rightarrow$ No. of bit

2) These are used for data conversion.
SIPO converts serial form of data into parallel form.
PISO converts parallel form of data into serial form

3) It performs Arithmetic operations
i. Left Shift Register is the multiplication by 2 circuit.
Right Shift Register is the division by 2 circuit (Error is 0.5 for odd numbers).

* Shift Registers are used to design Ring counter.

Q After 3 pulses o/p = ?



CLK	1	0	1	1
1	1	1	0	1
2	0	1	1	0
3	0	0	1	1

Counters

It counts the number of clock pulses applied to it.

Counters are of two types:

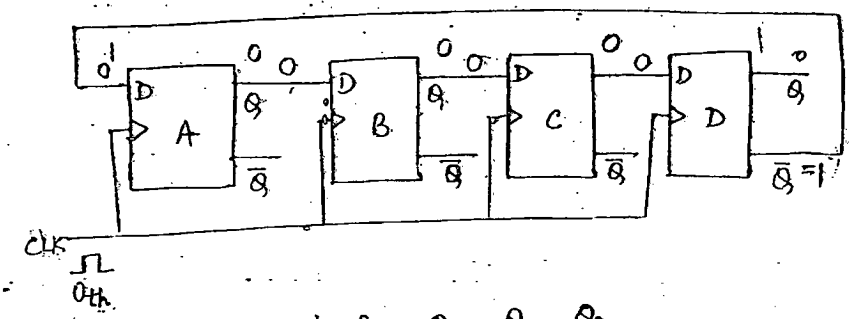
1) Synchronous counters (Parallel counters)
2) Asynchronous counters (Ripple counters)

NOTE:- Ring counters are special category of synchronous counters.

Ring counters are of two types:

1) Johnson's Ring counter (or) Twisted Ring Counter:

Feedback is given with the complementary output of last flip flop.

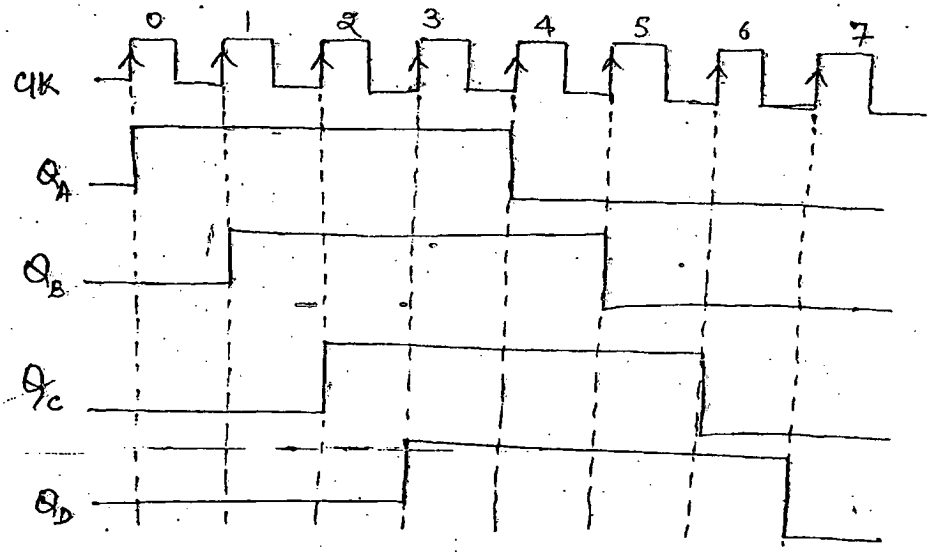


CLK	Q_A	Q_B	Q_C	Q_D	
0	1	0	0	0	→ 8
1	1	1	0	0	→ 12
2	1	1	1	0	→ 14
3	1	1	1	1	→ 15
4	0	1	1	1	→ 7
5	0	0	1	1	→ 3
6	0	0	0	1	→ 1
7	0	0	0	0	→ 0
8	1	0	0	0	X

mod 8

$n \rightarrow (2n)$ → possible states

→ clk pulses



Q What are the used state and unused states of the Johnson's Ring Counter? What will be the next state for the unused states? Check whether it is a self corrected counter or not?

Sol^n:- used states :-

$\rightarrow 8 \rightarrow 12 \rightarrow 14 \rightarrow 15 \rightarrow 7 \rightarrow 3 \rightarrow 1 \rightarrow 0$

un used states :-

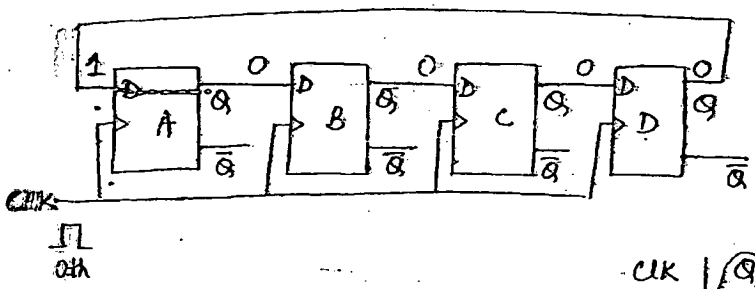
2, 4, 5, 6, 9, 10, 11, 13

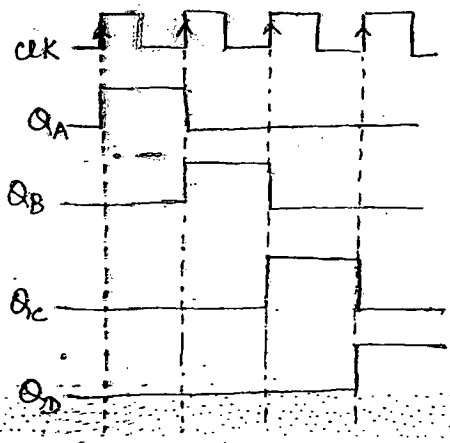
unused		next state
2	$\longrightarrow$	9
4	$\longrightarrow$	10
5	$\longrightarrow$	2
6	$\longrightarrow$	11
9	$\longrightarrow$	4
10	$\longrightarrow$	13
11	$\longrightarrow$	5
13	$\longrightarrow$	6

NOTE:- By observing the next state of unused states, we can say that it is not a self corrected counter, it is a lock out condition.

2) Ordinary Ring Counter

Feedback is given from the ex complementary output of last flip flop.

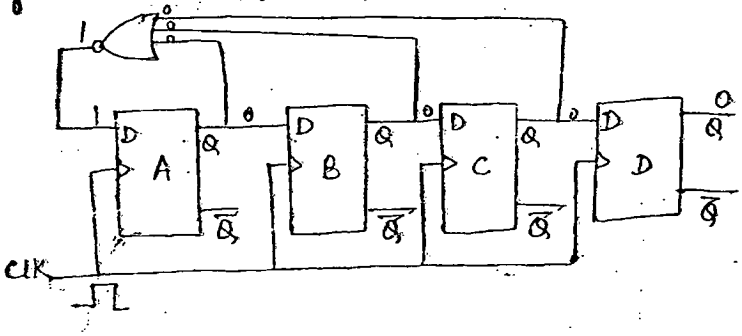




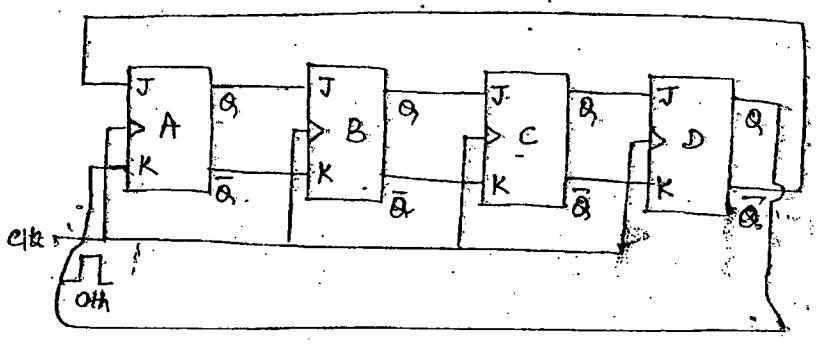
CLK	Q_A	Q_B	Q_C	Q_D	
0	1	0	0	0	mod 4
1	0	1	0	0	
2	0	0	1	0	
3	0	0	0	1	
4	1	0	0	0	X

n → (n) → possible states
(n) → clk pulses

Self Started Ring Counter



CLK	Q_A	Q_B	Q_C	Q_D
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

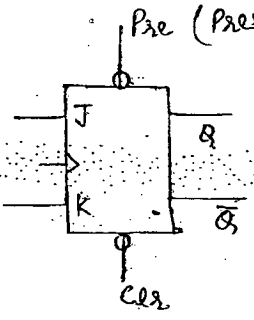


Johnson's Counter also known as

1) Creeping Counter
2) Walking Counter
3) Mobius Counter
4) Switch Tail Counter

Unused States
- Johnson's Counter $\Rightarrow 2^n - 2n$
- Ordinary Counter $\Rightarrow 2^n - n$

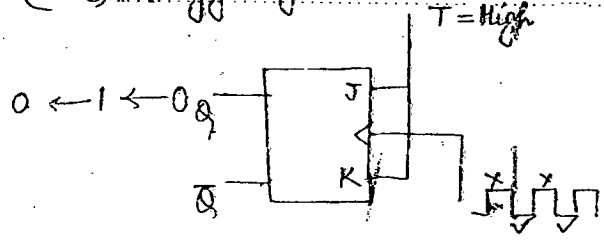
Asynchronous Inputs



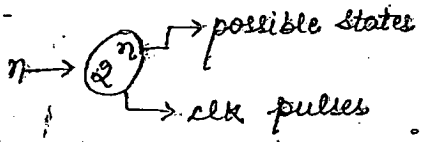
Pre	clr	Q^+
0	0	f f
0	1	0
1	0	1
1	1	X don't care

$\overline{Pre}$	$\overline{Clr}$	Q^+
1	1	f f
1	0	0
0	1	1
0	0	X don't care

(-ve) Triggering

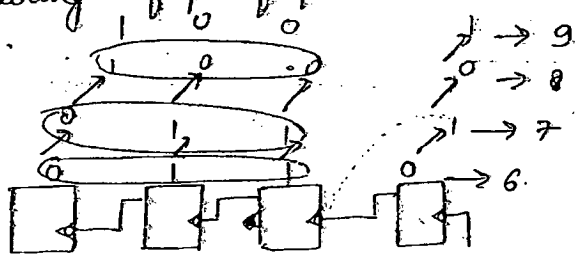


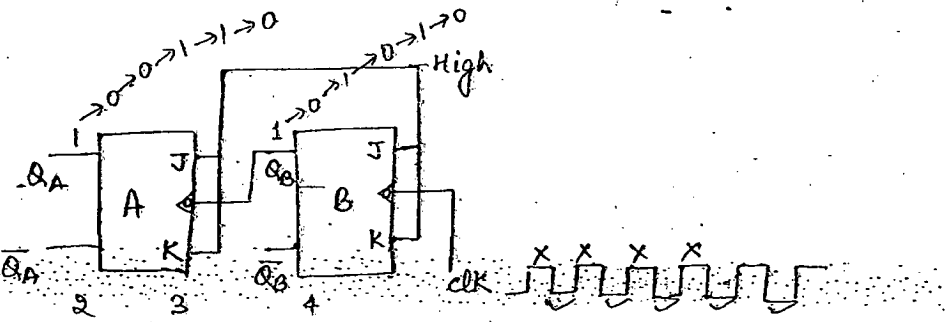
Asynchronous Counters (Ripple)

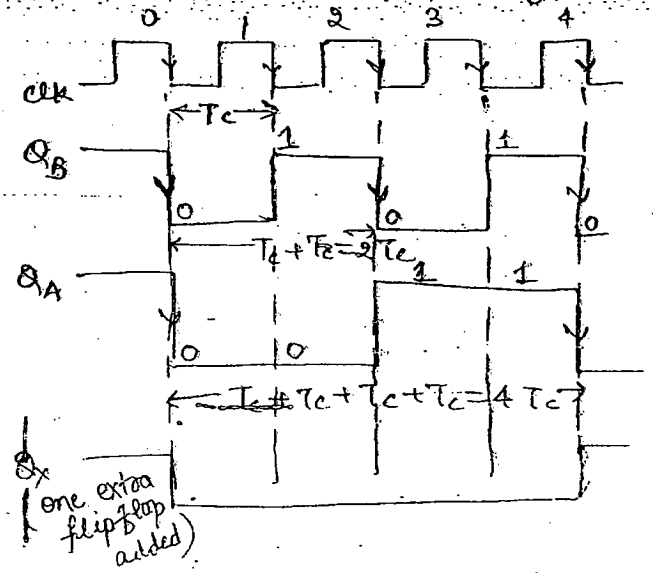


Flip flops are not triggered simultaneously.

NOTE: In the case of asynchronous counter, the triggering should be continued upto its possible extend. If the triggering was stopped at any flip flop then the remaining flip flop states should be written as it is.



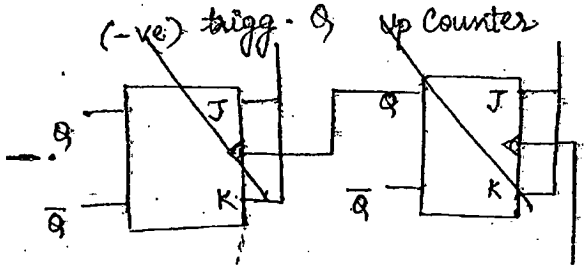




clk	Q_A	Q_B	
0	0	0	mod 4
1	0	1	
2	1	0	
3	1	1	
4	0	0	×

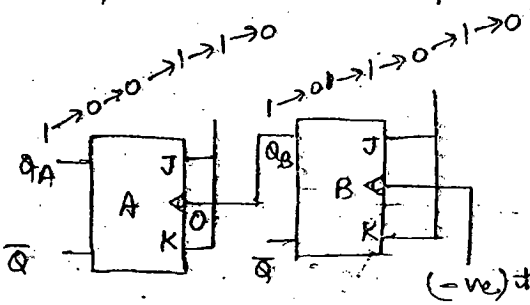
mod n ⇒ $\frac{\text{frequency}}{n}$

(-ve) trigg. Q Up Counter



-ve trigg
up up
down down

+ve trigg
up down
down up



(-ve) trigg & up connected so Up Counter

clk	Q_A	Q_B	
0	0	0	
1	0	1	
2	1	0	mod 4
3	1	1	
4	0	0	X

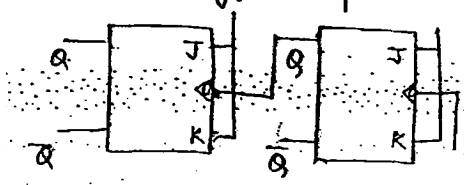
Up

0	000
1	
2	
3	
4	
5	
6	
7	111

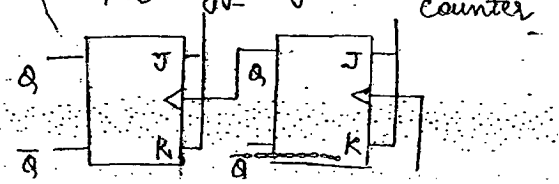
down

7	111
6	
5	
4	
3	
2	000
1	

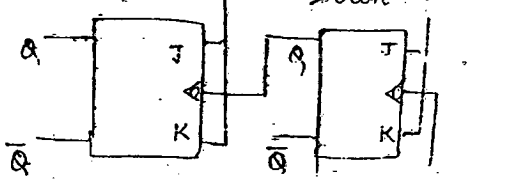
(-ve) trigg. Q Up counter



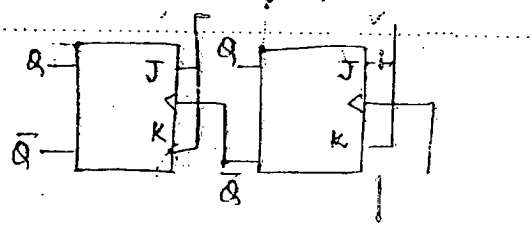
+ve triggering Q down Counter



(→ve trigg $\bar{Q}$ Down Counter



+ve trigg. $\bar{Q}$ Up Counter

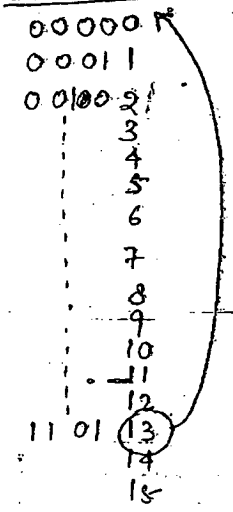


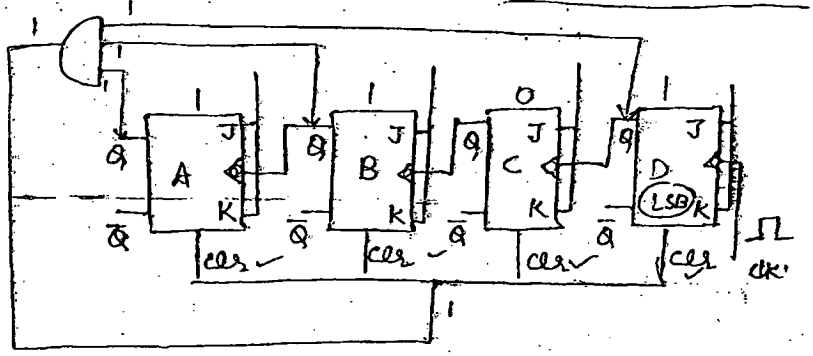
Designing of Asynchronous Mod Counter

Right trigger – right code
left trigger – reverse code

Mod 13 (using right trigger)

1101 ⇒ Mod 13

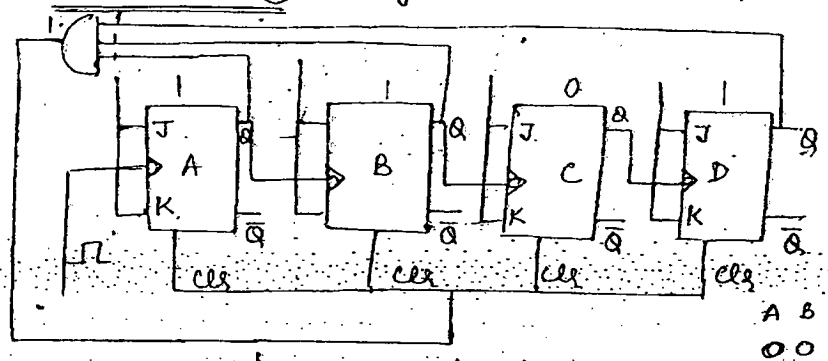


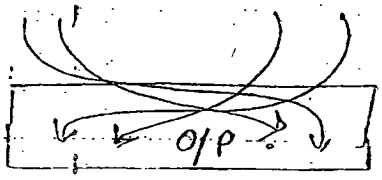


1101 ⇒ ~~Mod 13~~

↓ (reverse)

1011 ⇒ Mod (11) using left trigger

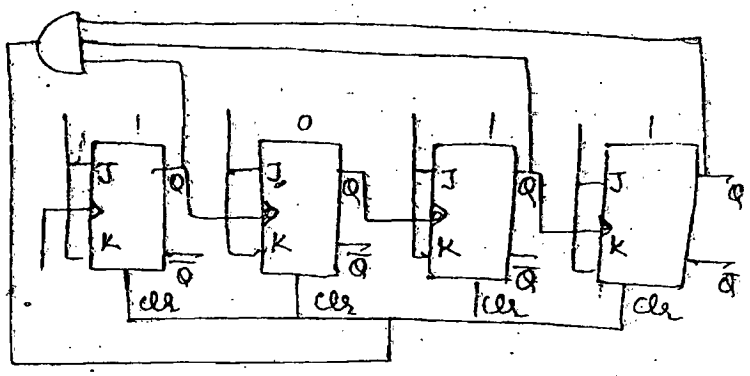




A	B	C	D	A	B	C	D
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				

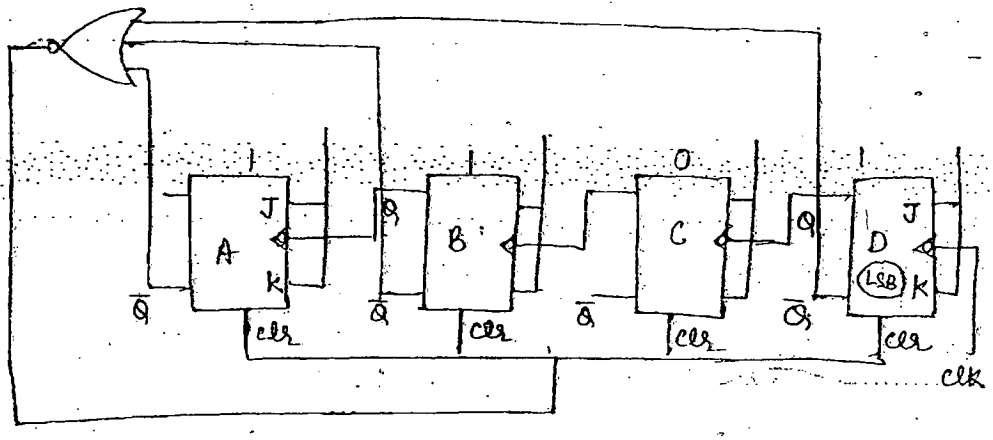
1101 ⇒ Mod 13 using left trigger

↳ 1011



Shortcut

NAND	Q	$\overline{clr}$
NOR	$\overline{Q}$	clr
AND	Q	clr
OR	$\overline{Q}$	$\overline{clr}$



Procedure to find Mod Value for Asynchronous Counter

Case 1: For Up Counter:-

Step 1:- Keep 1's at those flip flops which are having the connections to the logic gate. (other flip flop should be kept zero).

Step 2:- Check whether right triggered or left triggered.

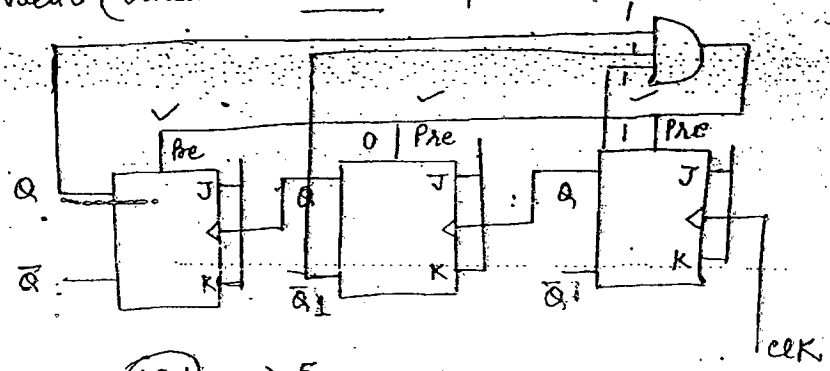
NOTE:- Right ~~triggering~~ → Right Code (↱)
Left Triggering → Reverse Code (for Mod Value) (↰)

Case 2:- For Down Counter

Step 1:- Keep the 1's at those position from which logic gate is having connection (other position should be kept as zero)

Step 2:- Check whether it is right triggered or left triggered.

NOTE:- Right triggered - right code value
Left triggered - left code value should be subtracted from the maximum state for the mod value (otherwise 1's compliment).



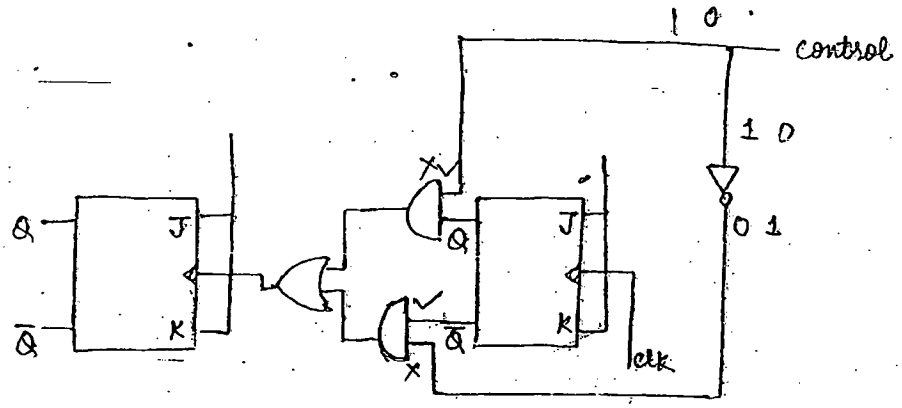
(101) → 5

1's compliment Mod 5

0 10 → 7 - 5 = (2)

Mod 2 → Mod (2)

Asynchronous Up/Down Counter



Control

1 → Up Counter

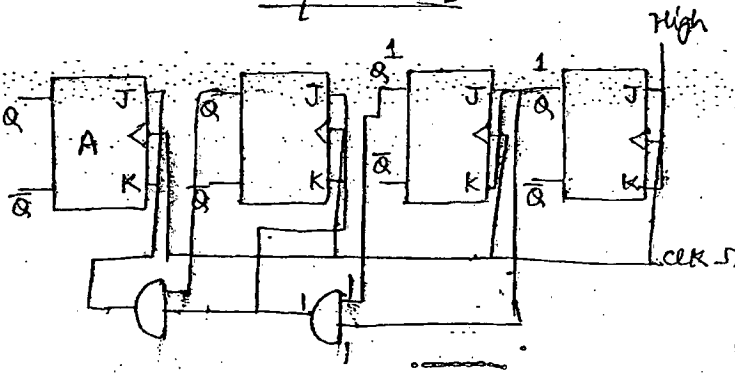
0 → Down Counter

Synchronous Counters

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

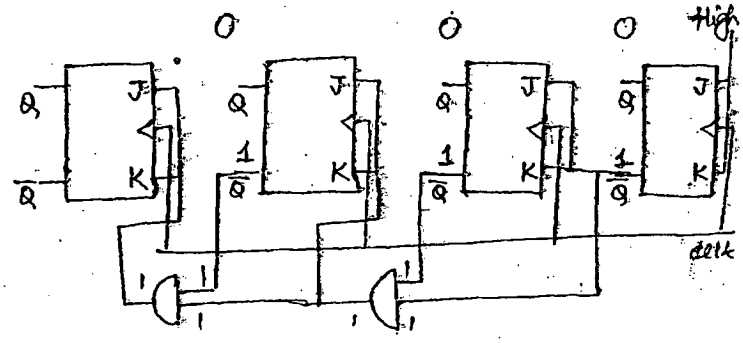
$n \rightarrow 2^n$ → possible states, → clk pulses

Synchronous Series Carry Up Counter



Synchronous Series Down Counter

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



Time Delay

First two flip flops never require logic gate.

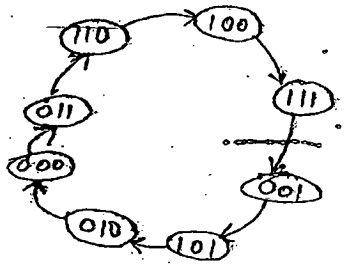
$$\text{Synch (series carry)} \Rightarrow t_p + (N-2)t_g$$

$t_g \rightarrow$ time delay of logic gate

Synchronous Counter Designs
(Sequence Generator)

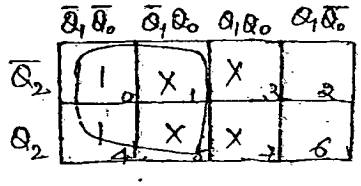
State Diagram	4
Next Table	7
Excitation	1
K-Map	5
Boolean Exp.	2
Design	0
	3
	6

State Diagram

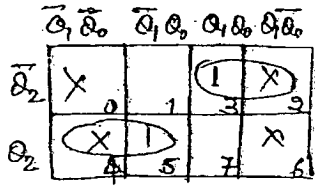


Present state $Q_2\ Q_1\ Q_0$	next state $Q_2^+\ Q_1^+\ Q_0^+$	J_0	K_0	J_1	K_1	J_2	K_2
1 0 0	1 1 1	1	X	1	X		
1 1 1	0 0 1	X	0	X	1		
0 0 1	1 0 1	X	0	0	X		
1 0 1	0 1 0	X	1	1	X		
0 1 0	0 0 0	0	X	X	1		
0 0 0	0 1 1	1	X	1	X		
0 1 1	1 1 0	X	1	X	0		
1 1 0	1 0 0	0	X	X	1		

Q Q^+ J K
0 0 0 X
0 1 1 X
1 0 X 1
1 1 X 0

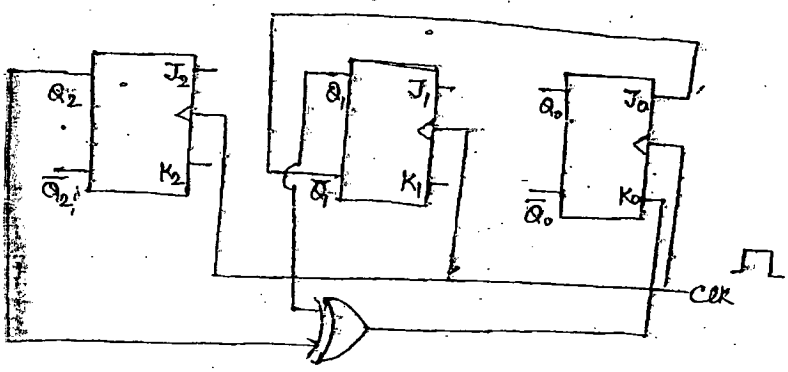


$J_0 = \bar{Q}_1$



$K_0 = \bar{Q}_2Q_1 + Q_2\bar{Q}_1$

$K_0 = Q_1 \oplus Q_2$



	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1 Q_0$	$Q_1 Q_0$	$Q_1\bar{Q}_0$
$\bar{Q}_2$	0	1	X_3	X_2
Q_2	4	5	X_7	X_6

$J_1 = \bar{Q}_0 + Q_2$

Pg 56

Q.13.

	present state Q_2 Q_1	next state Q_2^+ Q_1^+	T_1	T_2
0	0 0	1 0	0	1
2	1 0	1 1	1	0
3	1 1	0 1	0	1
1	0 1	0 0	1	0

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

	$\bar{Q}_1$	Q_1
$\bar{Q}_2$	0	1
Q_2	2	3

$T_1 = Q_1\bar{Q}_2 + Q_2\bar{Q}_1$

$T_1 = Q_1 \oplus Q_2$

	$\bar{Q}_1$	Q_1
$\bar{Q}_2$	0	1
Q_2	2	3

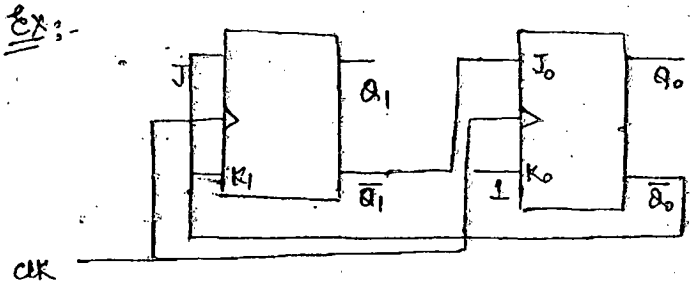
$T_2 = Q_1 Q_2 + \bar{Q}_1\bar{Q}_2$

$T_2 = Q_1 \odot Q_2$

Procedure to find Mod Value for Synchronous Counter

Step 1 : Keep the inputs and outputs in the form of table.
Step 2 : Write the corresponding input connections.
Step 3 : Keep some default state and make a horizontal line.

Ex:-



	$\bar{Q}_0$ J_1	$\bar{Q}_0$ K_1	$\bar{Q}_1$ J_0	1 K_0	Q_1	Q_0	
CLK							
0					0	0	mod (3)
1	1	1	1	1	1	1	
2	0	0	0	1	1	0	
3	1	1	0	1	0	0 ×	

$Q_1 Q_0 \Rightarrow 00, 11, 10, 00 \ldots$

(or)

$11, 10, 00, 11 \ldots$

$Q_0 Q_1 \Rightarrow 00, 11, 01, 00 \ldots$

$4Q_0 + 2Q_1 = ?$

$4(0) + 2(0) = 0$

$4(1) + 2(1) = 6$

$4(0) + 2(1) = 2$

$\Rightarrow 0, 6, 2 \ldots$

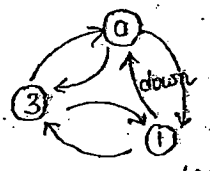
Q56

		1	1	Q_C	1	1	$\bar{Q}_A$			
Q	CLK	J_A	K_A	J_B	K_B	J_C	K_C	Q_A	Q_B	Q_C
	0							1	1	0
	1	1	1	0	1	1	0	0	0	1
	2	1	1	1	1	1	1	1	1	0

mod 2

X

Synchronous Up/Down Counter



2 → missing

2 10 → 2

2 10 → 6

up	down
0	3
1	1
3	0

Q	Q^+	JK
0	0	0 X
0	1	1 X
1	0	X 1
1	1	X 0

Up (Y=0)

Y	Present state Q_1 Q_0	Next state Q_1^+ Q_0^+	J_0 K_0	J_1 K_1
0	0 0	0 1	1 X	0 X
0	0 1	1 1	X 0	1 X
0	1 1	0 0	X 1	X 1

Down (Y=1)

Y	Present state Q_1 Q_0	Next state Q_1^+ Q_0^+	J_0 K_0	J_1 K_1
1	1 1	0 1	X 0	X 1
1	0 1	0 0	X 1	0 X
1	0 0	1 1	1 X	1 X

$\bar{Q}_1\bar{Q}_0$ $\bar{Q}_1Q_0$ Q_1Q_0 $Q_1\bar{Q}_0$

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

$J_0 = 1$

$\bar{Q}_1\bar{Q}_0$ $\bar{Q}_1Q_0$ Q_1Q_0 $Q_1\bar{Q}_0$

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

$K_0 = \bar{Y}Q_1 + Y\bar{Q}_1$

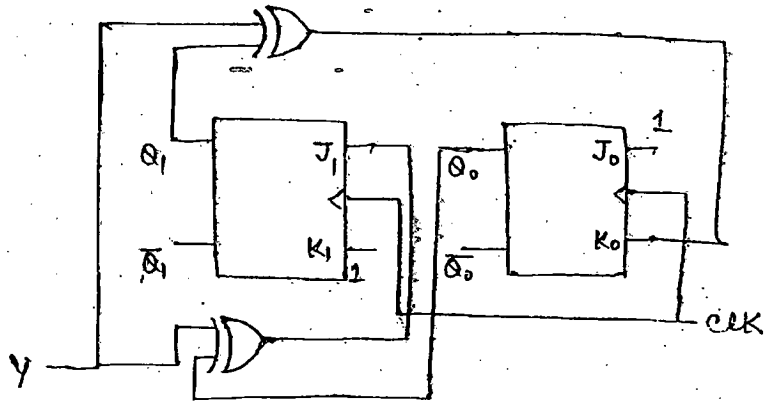
$K_0 = Y \oplus Q_1$

$\bar{Q}_1\bar{Q}_0$ $\bar{Q}_1Q_0$ Q_1Q_0 $Q_1\bar{Q}_0$

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

$\bar{Q}_1\bar{Q}_0$ $\bar{Q}_1Q_0$ Q_1Q_0 $Q_1\bar{Q}_0$

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



Lock Out Condition

* When counter went to unwanted state by voltage fluctuation and if it is able to come back to the wanted sequence after one or more clock pulses then it is known as self corrected counter.
* If it is not able to come back to the wanted state after any clock pulses it is known as not self corrected counter treated as Lock Out Condition.

Method to avoid Lock Out Condition

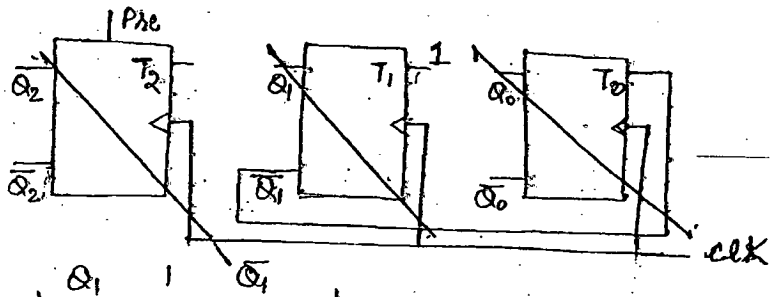
Design additional combinational circuit whose output is connected to proper preset and clear circuit to get the desired wanted state.

Wanted	Unwanted
0	1
3	2
5	4
6	7

$T_0 = \bar{Q}_1$

$T_1 = 1$

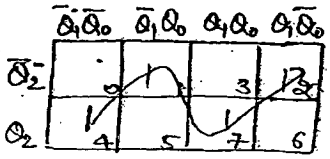
$T_2 = Q_1$



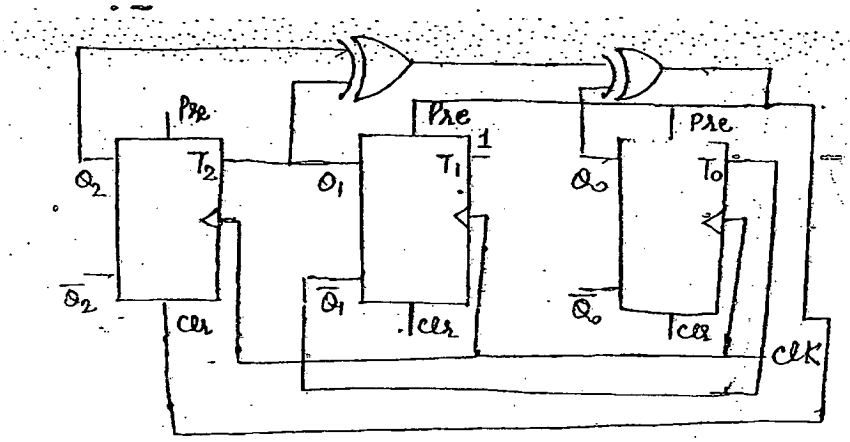
Q_1 | $\overline{Q_1}$

clk	T_2	T_1	T_0	Q_2	Q_1	Q_0	
0				1	0	0	→ 4
1	0	1	1	1	1	1	→ 7
2	1	1	0	0	0	1	→ 1
3	0	1	1	0	1	0	→ 2
4	1	1	0	1	0	0	→ 4

Not Self corrected
Lock Out Condition



$$f = Q_2 \oplus Q_1 \oplus Q_0$$



State Reduction Method

Step 1 : Draw the state diagram for the given circuit.

~~Step 2~~

State Diagram :- It is a graphical representation of a circuit having the parameters, present state, input, next state, output.

x/y denotes x is input, y is output

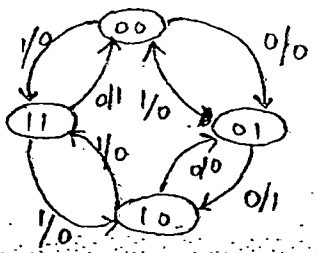
Step 2 : Draw the state table for the state diagram.

Step 3 : Identify the equivalent states.

Equivalent states : Two states are said to be equivalent if their next states and output are equal.

Step 4 : When two states are equivalent, one of the states should be eliminated.

Step 5 : Repeat the above process until we get all different states.



x/y
x → I/P
y → O/P

Present State		next state x=0	next state x=1	output y x=0	output y x=1
0	0	0 1	1 1	0	0
0	1	1 0	0 0	1	0
1	0	0 1	1 1	0	0
1	1	0 0	1 0	1	0

Pg 50

4)

Present state	next state x=0	next state x=1	output y x=0	output y x=1
a	c	b	0	0
b	d	c	0	0
c	~~f~~ e	d	1	1
d	e	~~f~~ d	1	0
e	~~f~~ d	a	0	1
~~f~~	~~g~~ e	f	1	0
~~g~~	f	a	0	1

5 states

d = f
e = g

33)

Present state	next state x=0	next state x=1	output y x=0	output y x=1
a	c	b	0	0
b	d	c	0	0
c	g	d	1	1
d	e	f	1	0
e	f	a	0	0
f	g	f	1	1
g	f	a	0	1

~~7 states~~

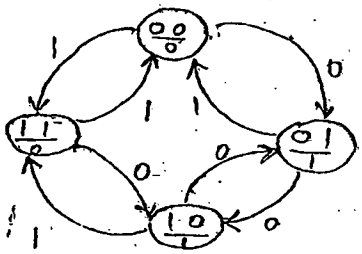
State Reduction methods are of two types:

~~Mealy~~ Mel

1) Mealy Method :- The present output depends on present state and present input.

The above discussed examples are Mealy Method only.

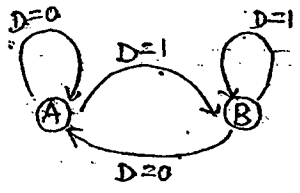
2) Moore Method :- In this method, the output depends only on present state.

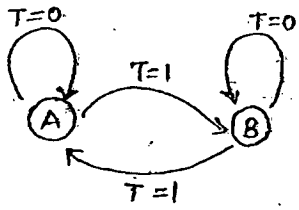


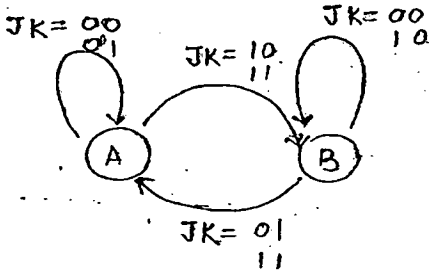
State Diagrame of flip flop

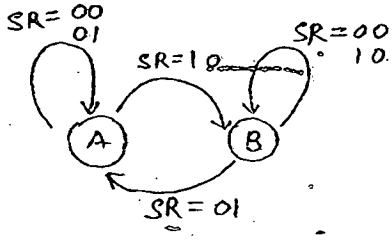
0 → A

1 → B

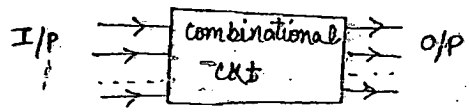








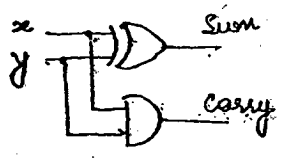
COMBINATIONAL CIRCUITS



Half Adder

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

$Sum = \bar{x}y + x\bar{y}$

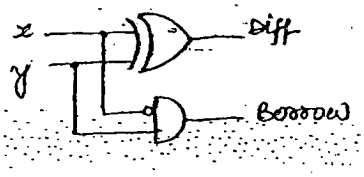


Half Subtractor

x	y	Borrow	Diff
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$Diff = x \oplus y$

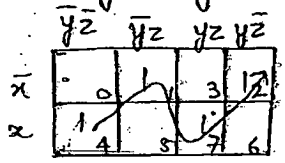
$Borrow = \bar{x}y$



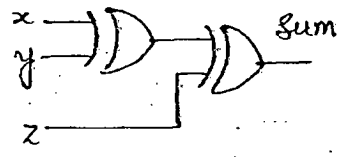
Full Adder

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

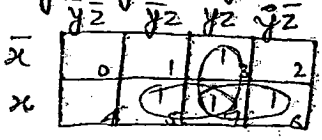
$Sum = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$



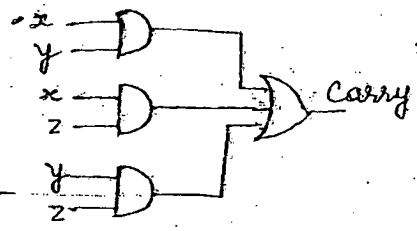
$Sum = x \oplus y \oplus z$



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



$Carry = xy + xz + yz$

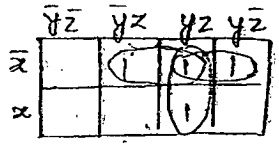


Full Subtractor

x	y	z	Borrow	Diff
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

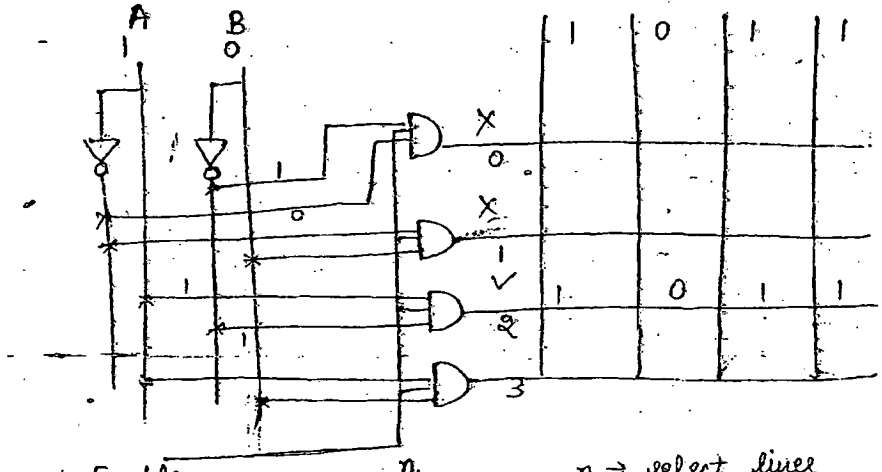
$Diff = x \oplus y \oplus z$

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



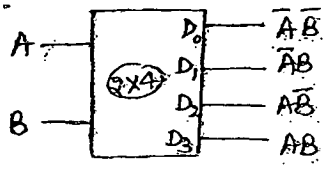
$Borrow = \bar{x}y + \bar{x}z + yz$

Decoder



$n \rightarrow$ select lines

$2^n \rightarrow$ address lines

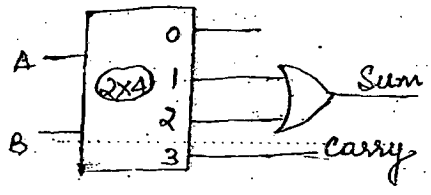


Half Adder

	A	B	Carry	Sum
0	0	0	0	0
1	0	1	0	1
2	1	0	0	1
3	1	1	1	0

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

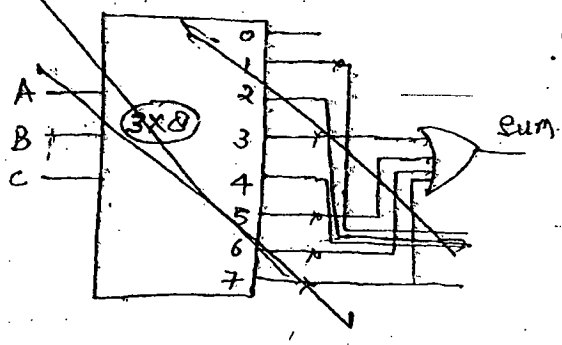
$Carry = \Sigma m(3)$



Full Adder

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

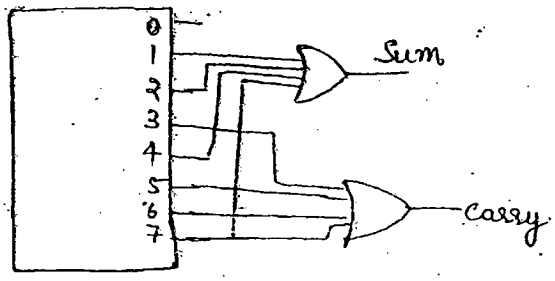
$Carry = \Sigma m(3,5,6,7)$



Full Adder

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

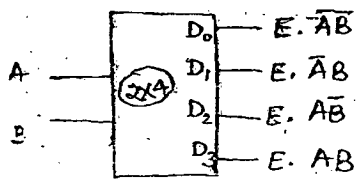
$Carry = \Sigma m(3,5,6,7)$



DeMux (one to Many)

Decoder with enable input is Demux.

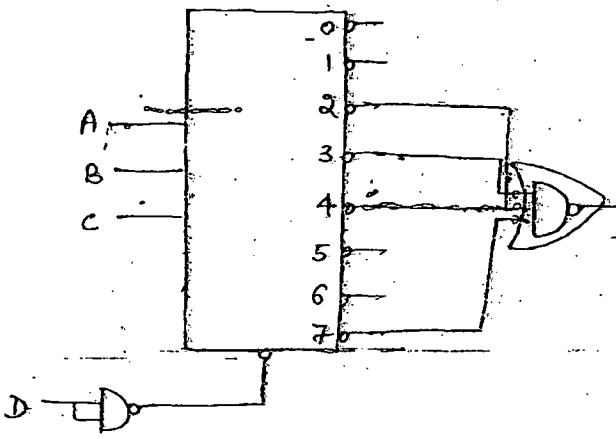
Ex: Telephone exchange with so many channels.



Full Adder

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

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	1	3	2
A	4	5	7	6



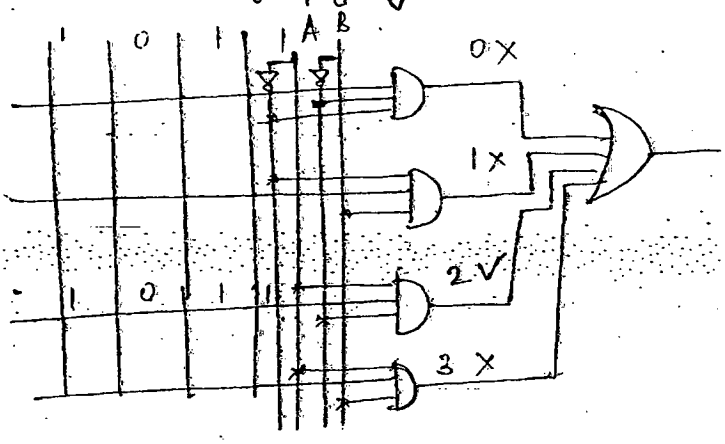
$f = Eg$

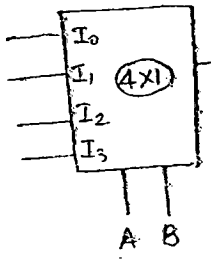
$= D[\bar{A}B + BC]$

NAND Gate
eqv:- Bubbled OR Gate

Multiplexer

- It is many to one circuit.
- It is a data selector.
- It is a parallel-serial converter.
- It is a waveform generator

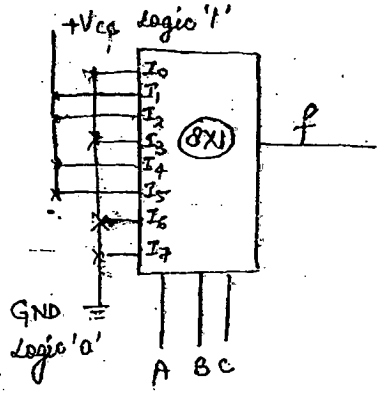


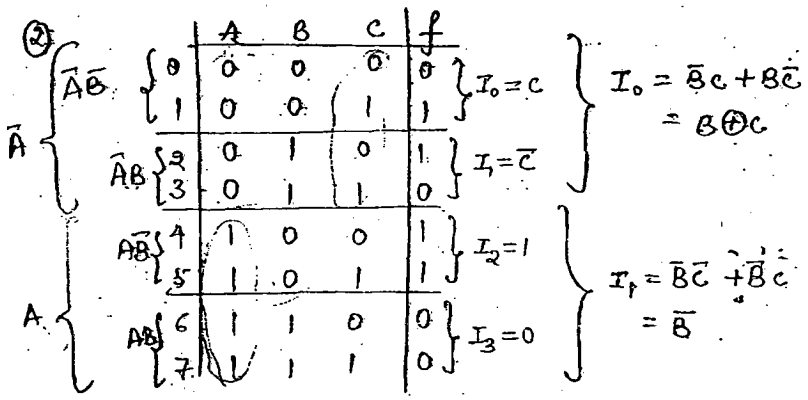


$f = \bar{A}\bar{B}(I_0) + \bar{A}B(I_1) + A\bar{B}(I_2) + AB(I_3)$

Designing by using Mux's

① $f = \Sigma m(1,2,4,5)$

①.① 

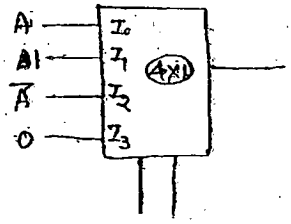
② 

		A	B	C	f	
$\bar{A}\bar{B}$	0	0	0	0	0	$I_0 = C$
	1	0	0	1	1	
$\bar{A}B$	2	0	1	0	1	$I_1 = \bar{C}$
	3	0	1	1	0	
$A\bar{B}$	4	1	0	0	1	$I_2 = 1$
	5	1	0	1	1	
AB	6	1	1	0	0	$I_3 = 0$
	7	1	1	1	0	

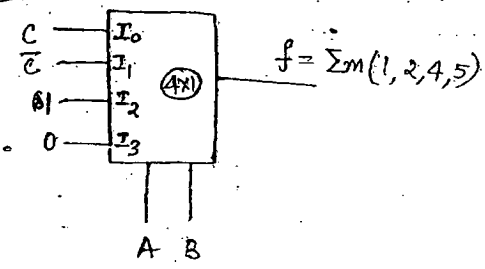
$I_0 = \bar{B}C + B\bar{C}$
$= B \oplus C$

$I_1 = \bar{B}\bar{C} + \bar{B}C$
$= \bar{B}$

②.① MSB i/p



②.② LSB i/p



③

	A	B	C
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

③.① If ~~Select MSB~~ (Horizontal La-La-La-La)

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

③.② i/p LSB ~~Select LSB~~ (Vertical La-La)

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

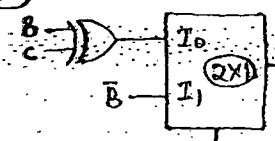
Designing by using (2X1) MUX

④

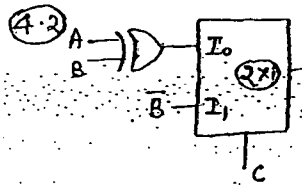
	C	A	B	f
$\bar{C}$	0	0	0	0
	0	0	1	1
	0	1	0	1
	0	1	1	0
C	1	0	0	1
	1	0	1	0
	1	1	0	1
	1	1	1	0

$I_0 = A \oplus B$

$I_1 = \bar{B}$

④.① 

$f = \Sigma m(1,2,4,5)$

④.② 

⑤ Select MSB (vertical La-La)

	I_0	I_1
$\bar{B}\bar{C}$	0	4
$\bar{B}C$	1	5
$B\bar{C}$	2	6
BC	3	7
	$\bar{B}C+B\bar{C}$	$\bar{B}\bar{C}+\bar{B}C$
	$= B \oplus C$	$= \bar{B}$

MSB select line

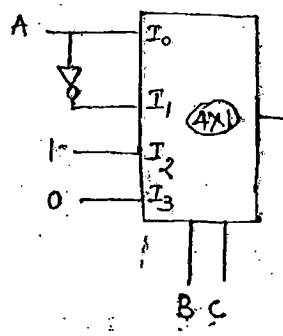
⑥ Select LSB (Horizontal La-La-La-La)

	I_0	I_1
$\bar{A}\bar{B}$	0	1
$\bar{A}B$	2	3
$A\bar{B}$	4	5
AB	6	7
	$A \oplus B$	$\bar{B}$

LSB Select line

(7)

Q



$f = \Sigma m\,(1,2,4,6)$

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

Given A $\bar{A}$ 1 0

Pg 49

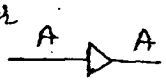
12

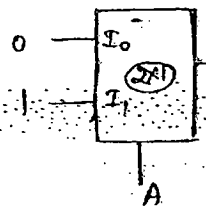
	I_0	I_1	I_2	I_3
$\bar{Z}$	⓪	2	4	6
Z	1	③	⑤	⑦
	$\bar{Z}$	Z	Z	Z

LSB Method

Logic gates by Mux's

1) Buffer



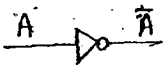


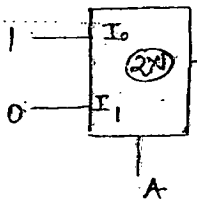
$f = \bar{A}(I_0) + A(I_1)$

$= \bar{A}(0) + A(1)$

$= A$

2) NOT





$f = \bar{A}(I_0) + A(I_1)$

$= \bar{A}(1) + A(0)$

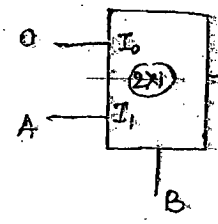
$= \bar{A}$

③ AND

	A	B	$f = A \cdot B$
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

$f = \Sigma m(3)$

	I_0	I_1
$\bar{A}$	0	1
A	2	3
	0	A



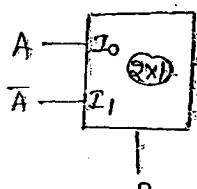
$f = \bar{B}(I_0) + B(I_1)$
$= \bar{B}(0) + B \cdot A$
$= A \cdot B$

④ Ex-OR

	A	B	$f = A \oplus B$
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

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

	I_0	I_1
$\bar{A}$	0	1
A	2	3
	A	$\bar{A}$



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

Shortcut :- No. of (2×1) Mux required to design logic gate.

	(2×1) Mux
NOT	1
AND	1
OR	1
Ex-OR	2
Ex-NOR	2
NAND	2
NOR	2

\# Q. (Half Adder) → No. of Mux's
Sum → Ex-OR → 2
Carry → AND → 1
3 Mux Required.

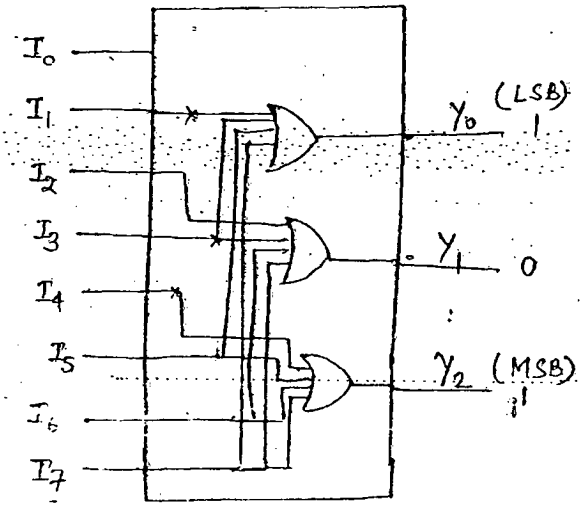
Encoder

I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$Y_0 = I_1 + I_3 + I_5 + I_7$

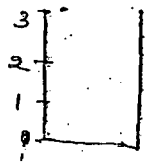
$Y_1 = I_2 + I_3 + I_6 + I_7$

$Y_2 = I_4 + I_5 + I_6 + I_7$



Octal to binary Code Encoder

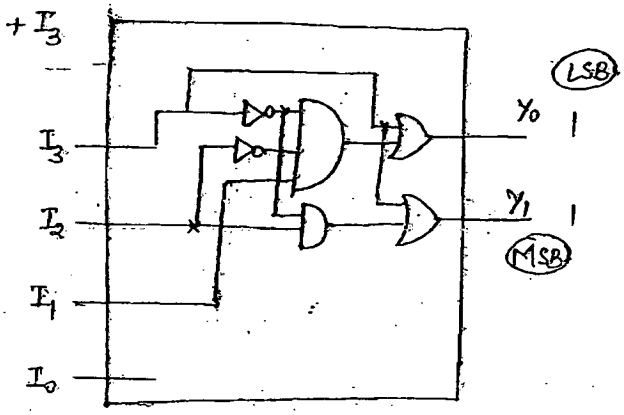
Priority Encoder



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

$Y_0 = I_1 \cdot I_2' \cdot I_3' + I_3$

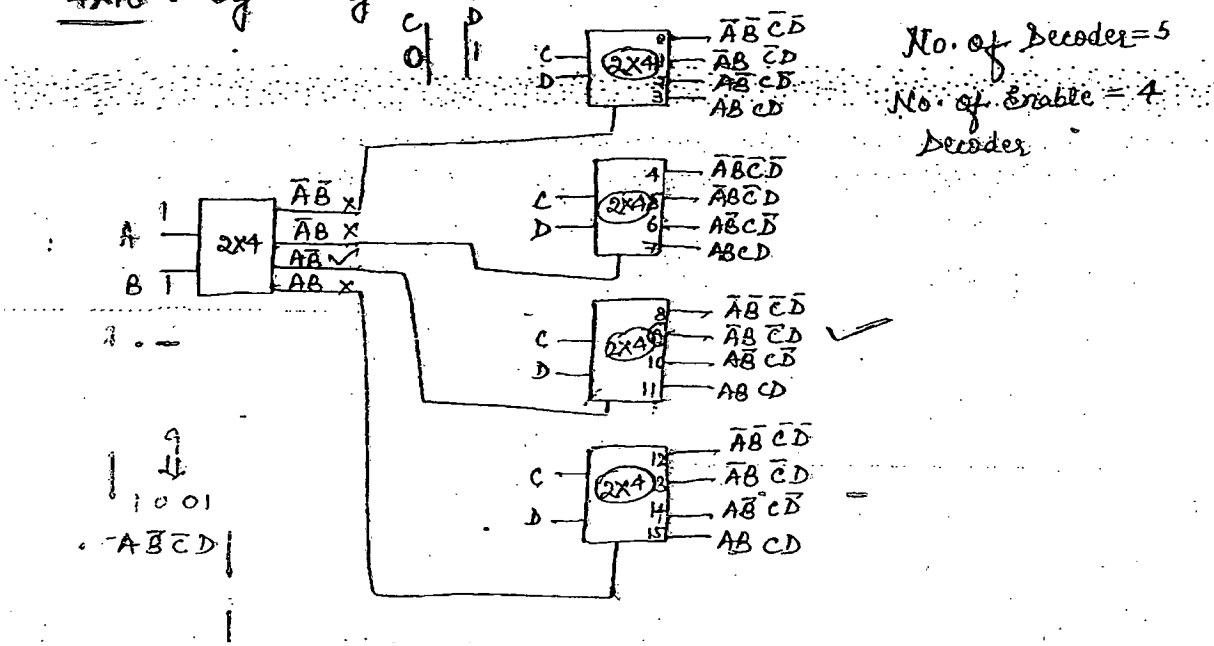
$Y_1 = I_2 \cdot I_3' + I_3$



4X2

Higher Order Circuits designing by Lower Order Circuits

4X16 : by using 2X4



No. of Decoder = 5

No. of Enable = 4 Decoder

Shortcut: No. of 2×4 required for 4×16

$$\frac{16}{4} = 4$$

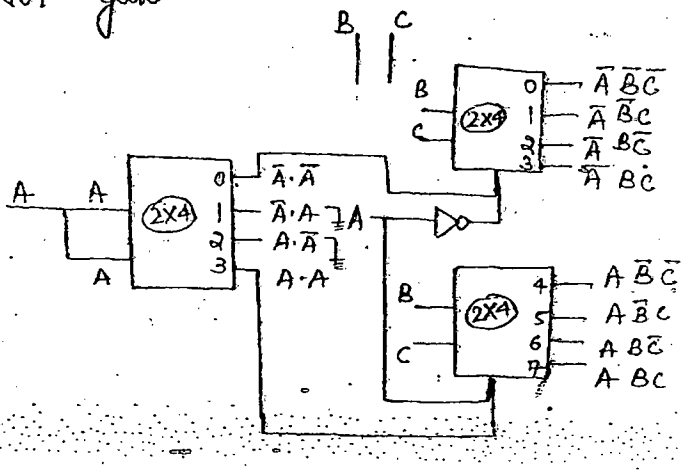
$$\frac{4}{4} = 1$$

∴ Total 2×4 Required = 4+1 = 5

3×8 : by using 2×4

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

* Whenever full division not possible then one extra circuit will be required.
* If odd number present then the circuit required will be NOT gate.



16×256 by using 4×16

$$\frac{256}{16} = 16$$

$$\frac{16}{16} = 1$$

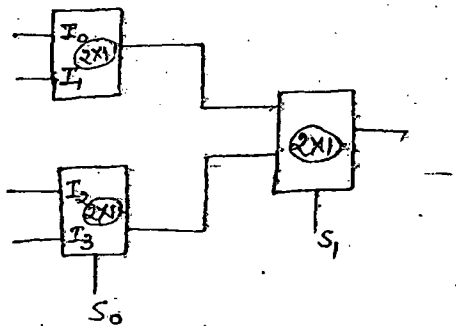
17

4x1 Mux by using 2x1

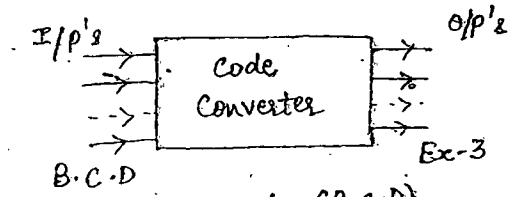
$\frac{4}{2} = 2$

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

Total No of Mux = 3



Codes and Code Converter



Binary Decimal Code (B·C·D)

8 4 2 1

7 5 9 3 6 1

0111 0101 1001 0011 0110 0001. → B·C·D

1/0

0001 0000 → B·C·D

1/6

0001 0110

10 → 1010
11 → 1011
12 → 1100
13 → 1101
14 → 1110
15 → 1111

} Invalid B·C·D.

Self Complimentary Code

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

* By adding 3 (11) in BCD Number, it becomes excess-3.

0 → 0000 BCD
11
0011 → Ex-3

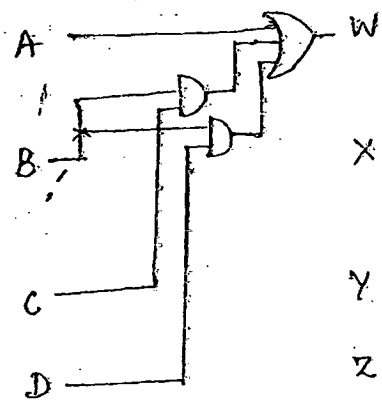
1 → 0001 BCD
11
0100 → Ex-3

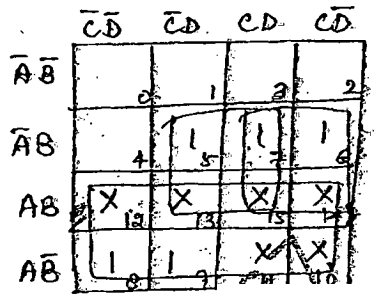
B.C.D to Ex-3 (Excess-3) Code Converter

	I/P B.C.D				O/P Ex-3			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	0	0	0	0
6	0	1	1	0	0	0	0	1
7	0	1	1	1	0	0	1	0
8	1	0	0	0	0	0	1	1
9	1	0	0	1	0	1	0	0

I/P B.C.D

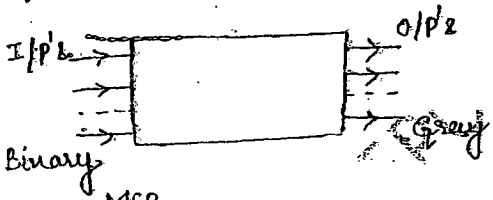
O/P Ex-3

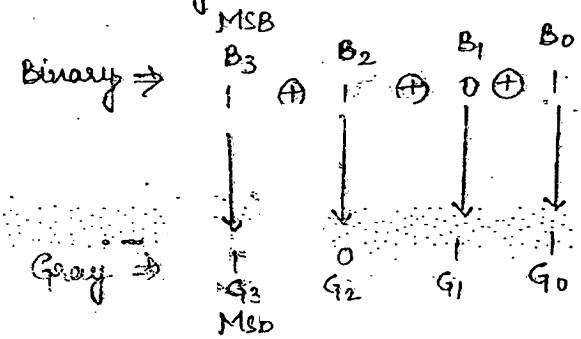




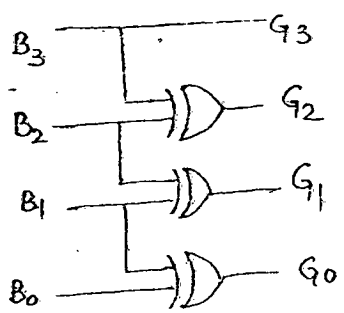
$W = A + BC + BD$

Binary to Gray Codes





I/P Binary — O/P Gray



$G_3 = B_3$

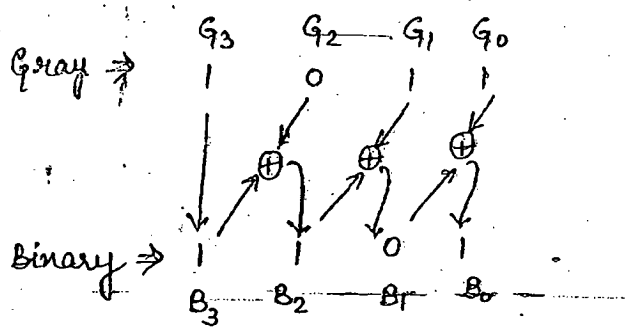
$G_2 = B_3 \oplus B_2$

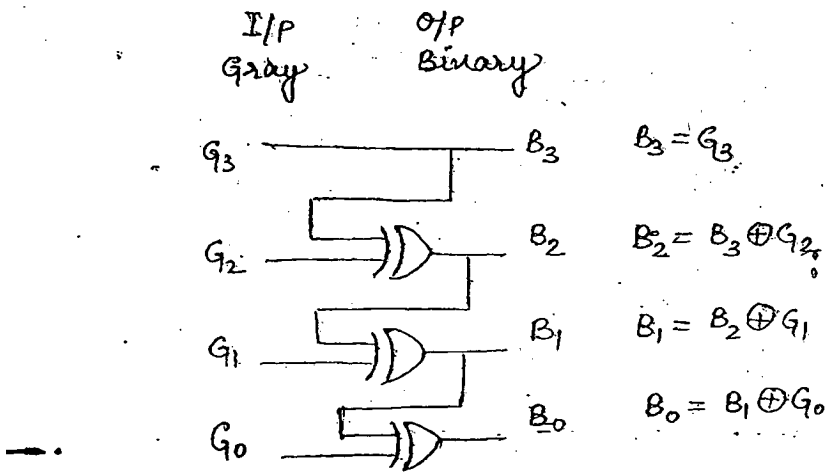
$G_1 = B_2 \oplus B_1$

$G_0 = B_1 \oplus B_0$

~~Binary to Gray Code~~

Gray to Binary Codes





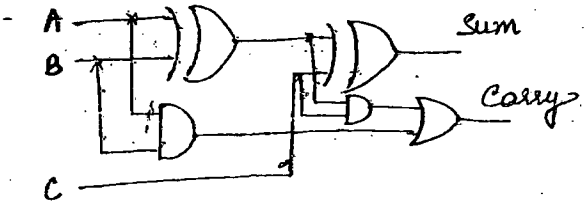
\#

Binary		Gray	
0 → 00	1	00	1
1 → 01		01	
2 → 10	2	11	1
3 → 11	1	10	

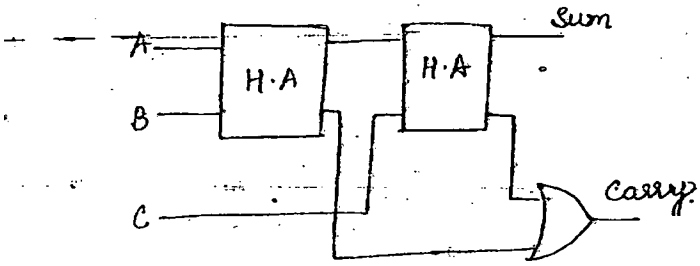
* Difference in Gray code is 1.
* It is known as Hamming Distance = 1
  (Unit distance code)
  → used in K-Maps

Full Adder

A, B, C → Sum → Sum



using Half Adder

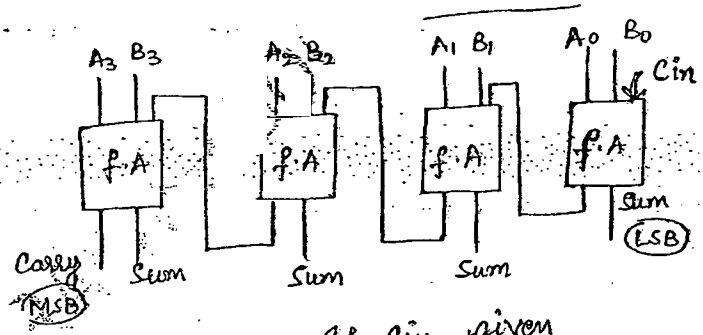


Ripple Carry Adder (or) Parallel Carry Adder

$A \rightarrow A_3\ A_2\ A_1\ A_0$

$B \rightarrow B_3\ B_2\ B_1\ B_0$

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



(N) f·A ⇒ If Cin given

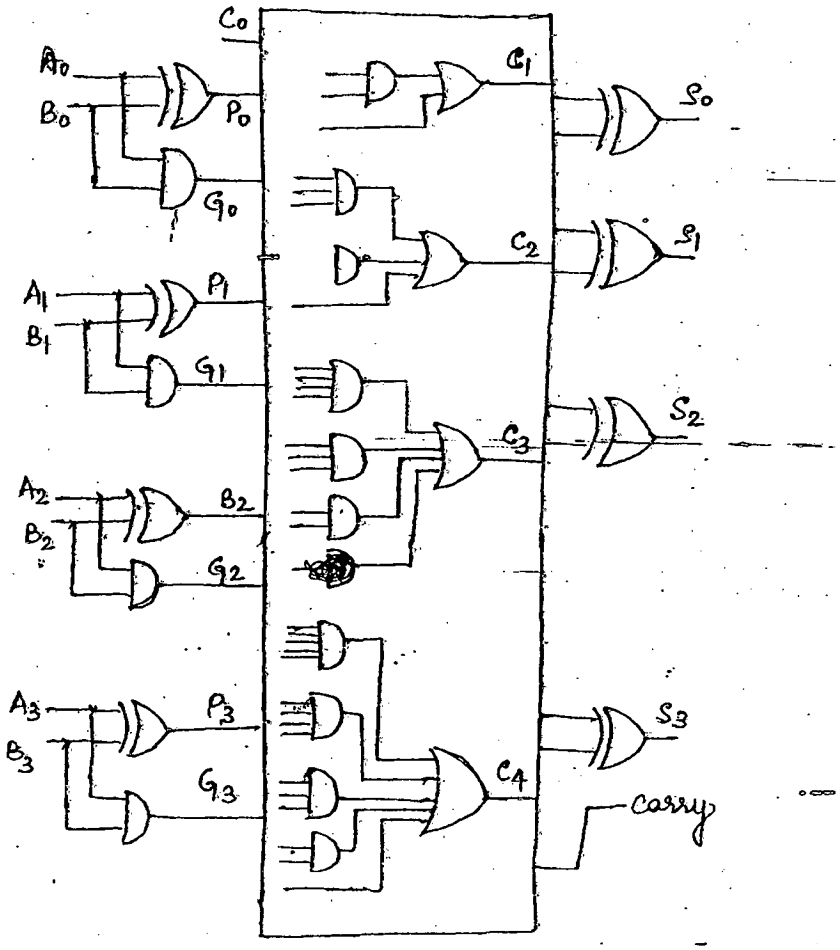
(N-1) f·A + 1 H·A

(N-1) [2 H·A + 1 OR gate] + 1 H·A

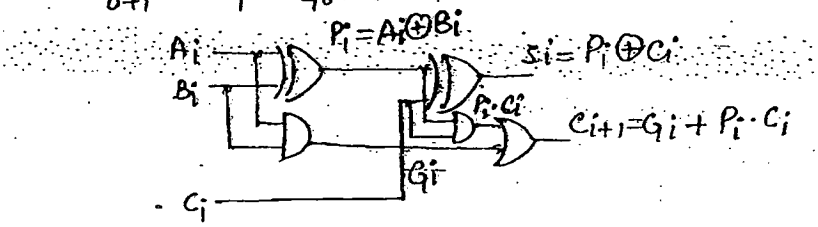
(2N-1) H·A + (N-1) OR gates

Ex: N = 4 $(2 \times 4 - 1) = 7$ H·A N-1

4-1 = 3 OR gates



$C_{0+1} = C_1 = G_0 + P_0 C_0$



$C_{0+1} = C_1 = G_0 + P_0 C_0$

$C_{1+1} = C_2 = G_1 + P_1 C_1 = G_1 + P_1 [G_0 + P_0 C_0] = G_1 + P_1 G_0 + P_1 P_0 C_0$

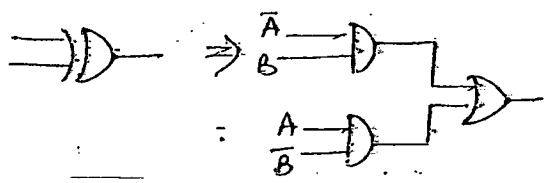
$C_{2+1} = C_3 = G_2 + P_2 C_2 = G_2 + P_2 [G_1 + P_1 G_0 + P_1 P_0 C_0]$

$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

$C_{3+1} = C_4 = G_3 + P_3 C_3 = G_3 + P_3 [G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0]$

$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$

Q



Carry Generator -

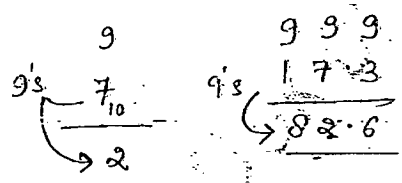
Eg:

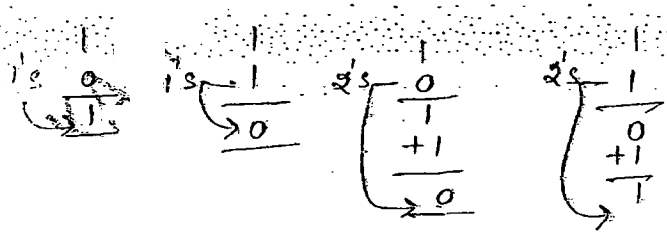
No. of AND gates $\Rightarrow \frac{n(n+1)}{2}$ $= \frac{4(4+1)}{2} = 10$

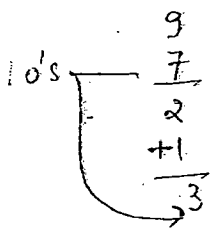
No. of ~~OR~~ Gates $\Rightarrow n$ $\rightarrow 4$

Compliment Methods

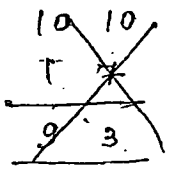
	Dec	Bin	Oct	Hexa
r's $\Rightarrow$	10's	2's	8's	16's
(r-1)'s $\Rightarrow$	9's	1's	7's	15's







10's



10's

999
17.3
82.6
+1
82.7

2's Complement Shortcut

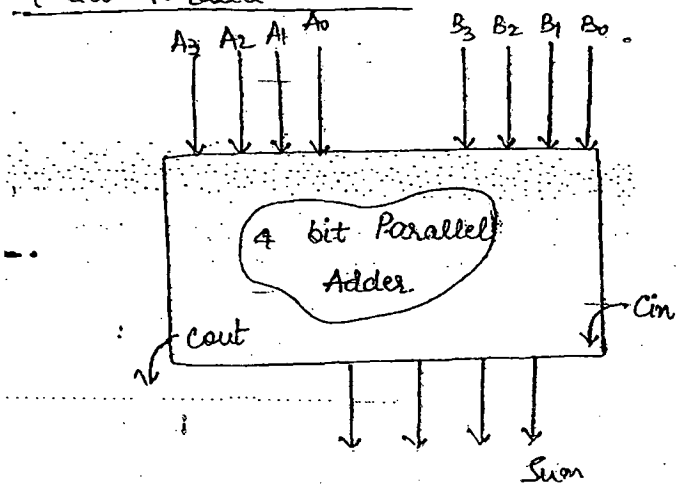
Write as it is till where the 1 is present. (from LSB side), take the complement of each (1 to 0 and 0 to 1) of the remaining.

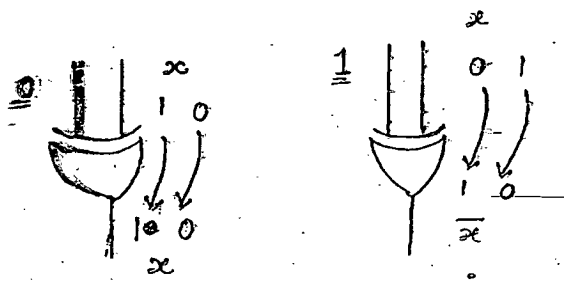
10100) 2's
01100

10101) 2's
01101

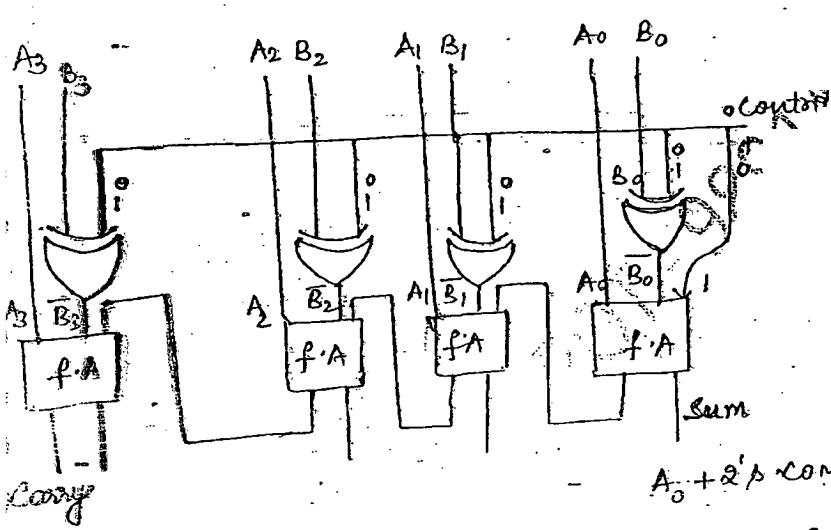
10000) 2's
10000

4 bit Parallel Adder





2's Complement Adder / Subtractor



$A_3 A_2 A_1 A_0$

$B_3 B_2 B_1 B_0$

A_0 + 2's comp. B_0

Control ⇒ 0 ⇒ A + B

1 ⇒ A − B

x → minuend

y → substrahend

A → 1 0 1 1

B → 1 0 1 0 ($B_3 B_2 B_1 B_0$)

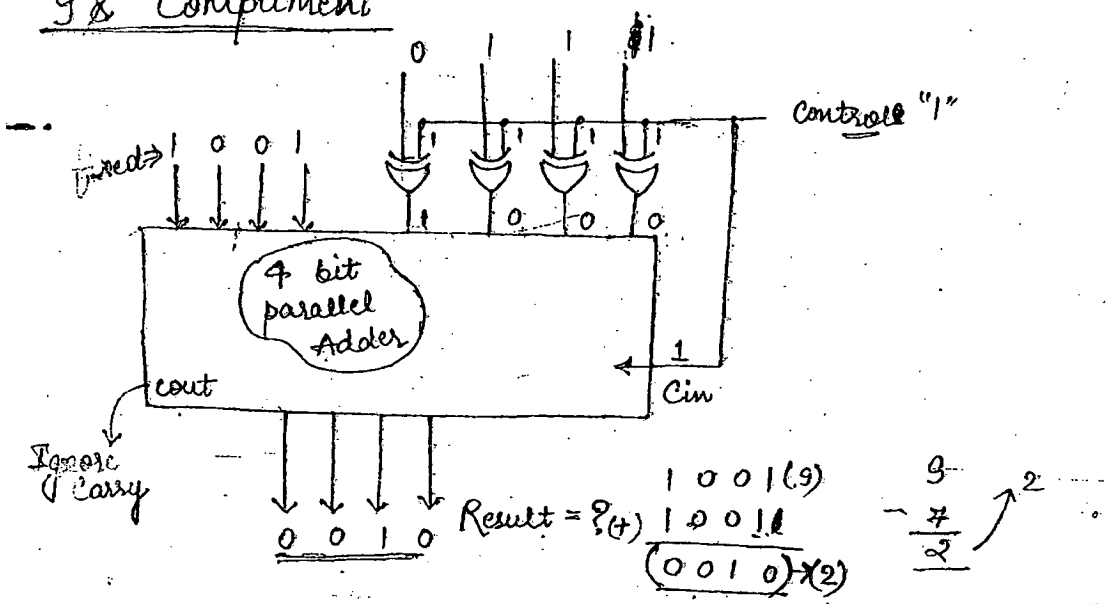
0 1 0 1

+1

2's comp

0 1 1 0

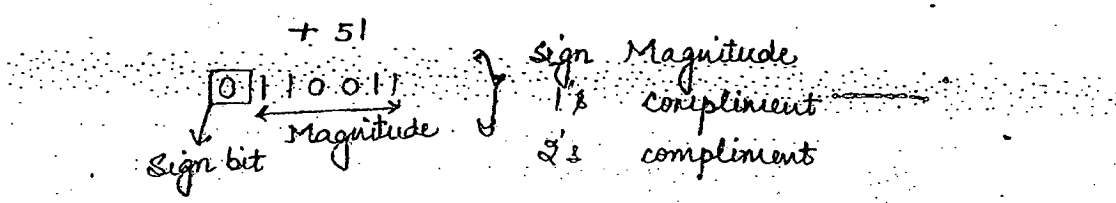
9's Compliment



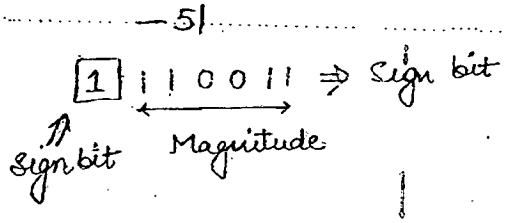
NUMBER REPRESENTATION

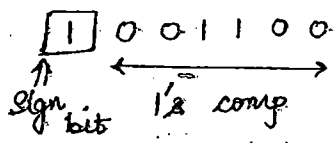
1) Sign Magnitude
2) 1st 1's compliment
3) 2's compliment

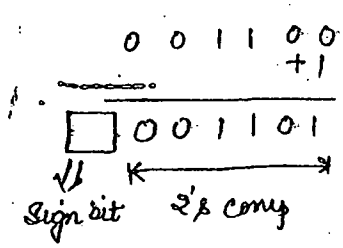
Case (i) : (+)ve No's



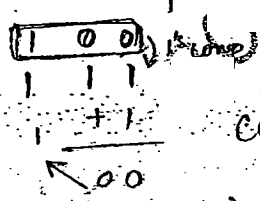
Case (ii) : (-ve) No's







Sign Mag.			1's Comp			2's comp		
x	y	z	x	y	z	x	y	z
0	0	0 +0	0	0	0 +0	0	0	0 +0
0	0	1 +1	0	0	1 +1	0	0	1 +1
0	1	0 +2	0	1	0 +2	0	1	0 +2
0	1	1 +3	0	1	1 +3	0	1	1 +3
1	0	0 -0	1	1	1 -0	1	0	0 Special case (-4)
1	0	1 -1	1	1	0 -1	1	1	1 -1
1	1	0 -2	1	0	1 -2	1	1	0 -2
1	1	1 -3	1	0	0 -3	1	0	1 -3



For Special case:

$$\boxed{(-2^{n-1})}$$

$1\,0\,0 \Rightarrow -4$

$1\,0\,0\,0 \Rightarrow -8$

$1\,0\,0\,0\,0 \Rightarrow -16$

} Magnitude of number with negative sign

Range :-

$+[2^{n-1}-1]$ to $-[2^{n-1}-1] \Rightarrow$ Sign Mag
$\Rightarrow$ 1's comp. comp

$n=3$ $+(2^{3-1}-1)$ to $-(2^{3-1}-1)$

$+3$ to -3

$+(2^{n-1}-1)$ to $-(2^{n-1}) \Rightarrow$ 2's comp

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

$+3$ to -4

Q

	sign	1's	2's
0101	+5	+5	+5
11101	+13	-2	-3

B.C.D ADDITION

Case (i)

$5 \Rightarrow 0101$

$4 \Rightarrow 0100$

1001

When two valid BCD numbers are added, the result is invalid then 6 should be added to get the final result in the BCD form.

Case (i)

$5 \rightarrow 0101$

$4 \rightarrow 0100$

$9 \quad 1001 \rightarrow$ valid

Case (ii)

$5 \rightarrow 0101$

$7 \rightarrow 0111$

1100 invalid

0110 (Add 6)

$0001 \quad 0010$

12

Case (iii)

$8 \rightarrow 1000$

$9 \rightarrow 1001$

0001 valid

carry $\downarrow$ 0110

$0001 \quad 0111$

17

$10 \rightarrow 1010$

$11 \rightarrow 1011$

$12 \rightarrow 1100$

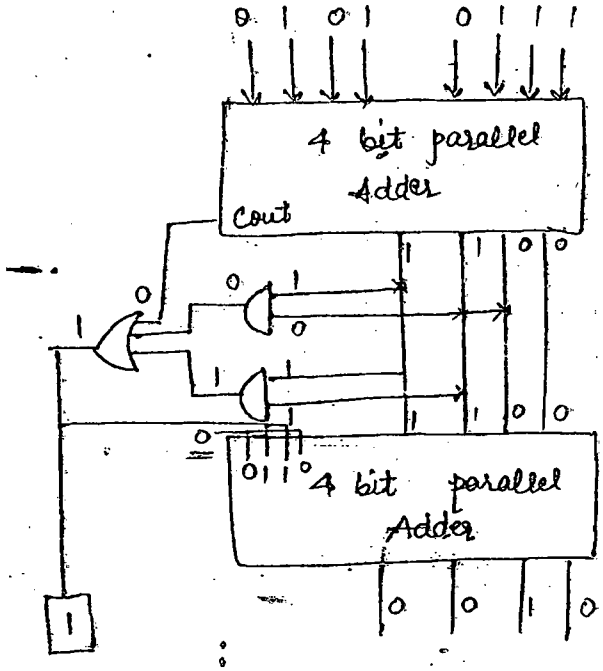
$13 \rightarrow 1101$

$14 \rightarrow 1110$

$15 \rightarrow 1111$

} Six invalid B.C.D

B.C.D Adder



	8	4	2	1
Invalid BCD	1	X	1	X
	1	1	X	X

B.C.D Subtraction

Case (i) : Result (+)ve

9's compliment of subtrahent should be added to the minuend. If the result is invalid 6 should be added along with end around-carry.

$+6 \Rightarrow 0\ 1\ 1\ 0$

$-4 \Rightarrow 0\ 1\ 0\ 1$ [9's comp of 4]

$1\ 0\ 1\ 1$ invalid

$0\ 1\ 1\ 0$

$1\ 0\ 0\ 0\ 1$

$\Rightarrow \quad +1$

$0\ 0\ 1\ 0$ $\quad +2$

end around carry

Case (ii) : Result (−)ve

9's compliment of substrahend should be added to the minuend. If the result is valid and no carry, then to get the final answer, result should be 9's complimented.

$-6 \Rightarrow 0\ 0\ 1\ 1$ [9's comp of 6]

$+4 \Rightarrow 0\ 1\ 0\ 0$

$0\ 1\ 1\ 1$ Valid

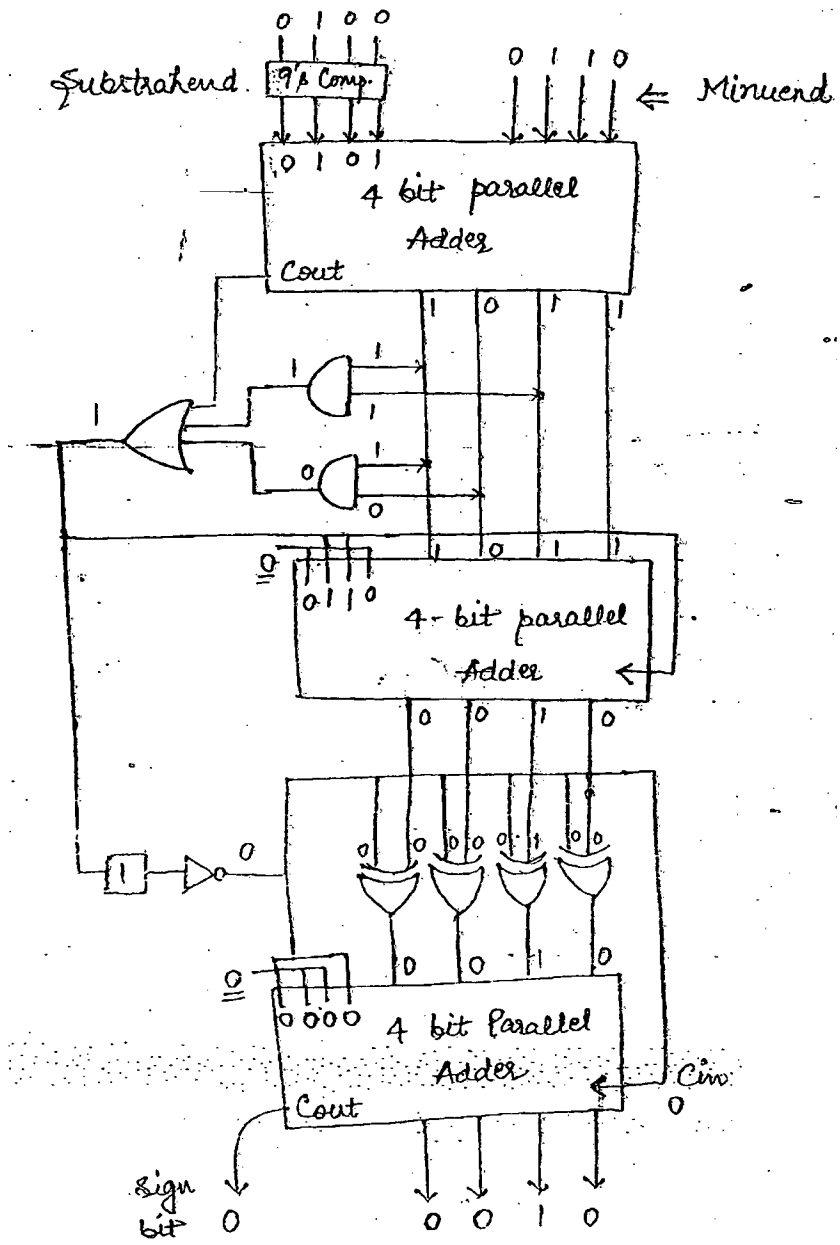
No Carry

If Result is
1) Valid B.C.D No
2) No Carry

Result is (−)ve

To get final Ans ⇒ 9's comp

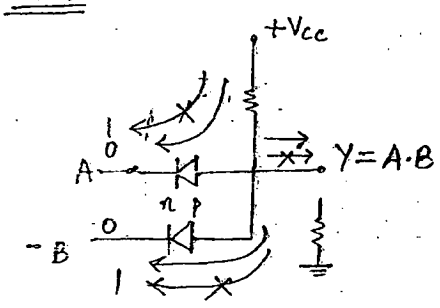
9
−7
= 2


Subtrahend
9's Comp.
Minuend
4 bit parallel Adder
Cout
4- bit parallel Adder
4 bit Parallel Adder
Cin
Cout
Sign bit

(+2)

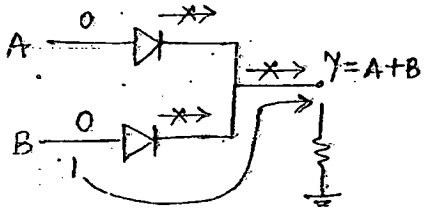
Switching Devices as Logic Gates

AND

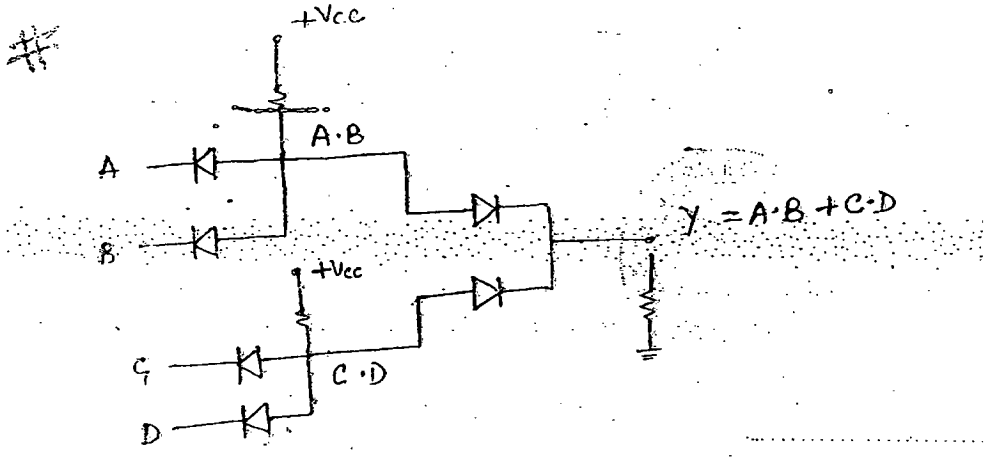


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR

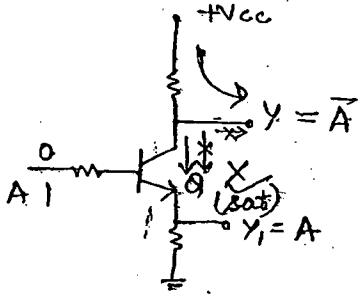


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



Resistor Transistor Logic (RTL)

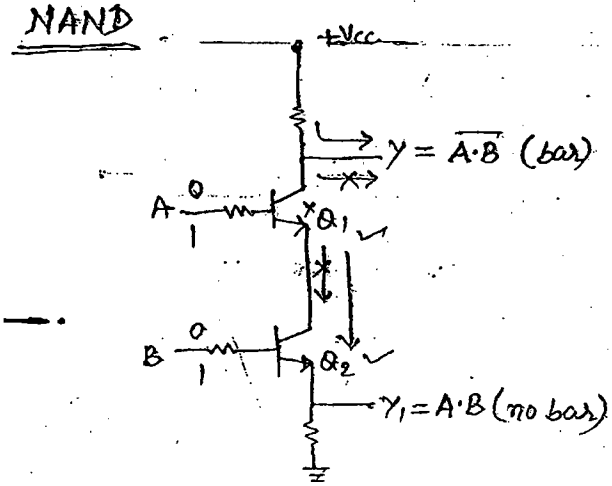
NOT



A	Q	Y
0	X	1
1	✓	0

Shortcut :–
column = AND operation
row = OR operation
(same for N-MOS)

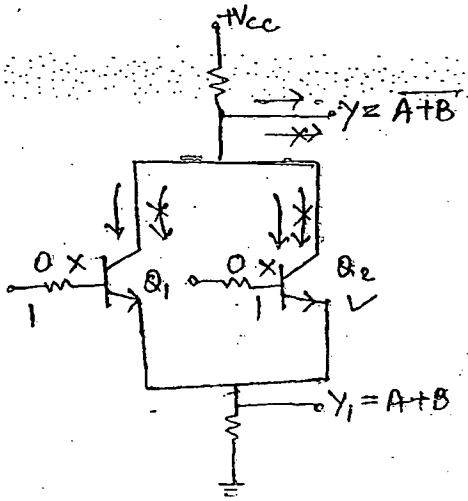
NAND



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

A	B	Q_1	Q_2	Y
0	0	X	X	1
0	1	X	✓	1
1	0	✓	X	1
1	1	✓	✓	0

NOR



A	B	Q_1	Q_2	Y
0	0	X	X	1
0	1	X	✓	0
1	0	✓	X	0
1	1	✓	✓	0

Direct Coupled Transistor Logic (DCTL)

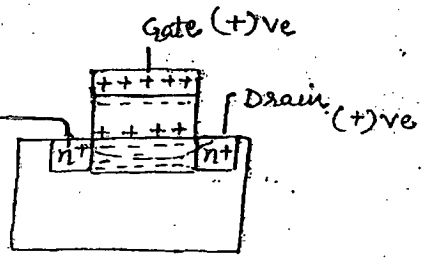
* It is the same as that of RTL except that input resistor are removed.
* They suffer with Current Hogging problem.

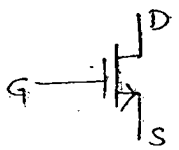
Current Hogging: Because of different saturation levels of loading gates, current will be blocked or hogged in certain loads only and certain loads are in the starvation of current known as current Hogging problem.

MOS

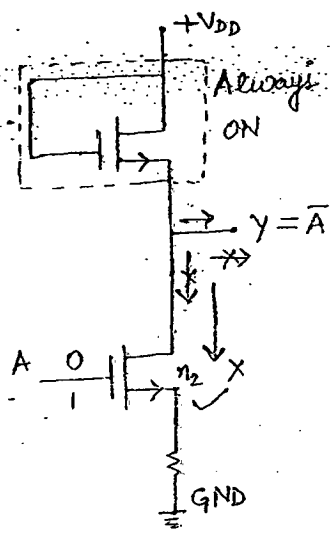
N-MOS

N-MOS ⇒ Gate ⇒ (+)ve ⇒ ON
Gate ⇒ (-)ve ⇒ OFF



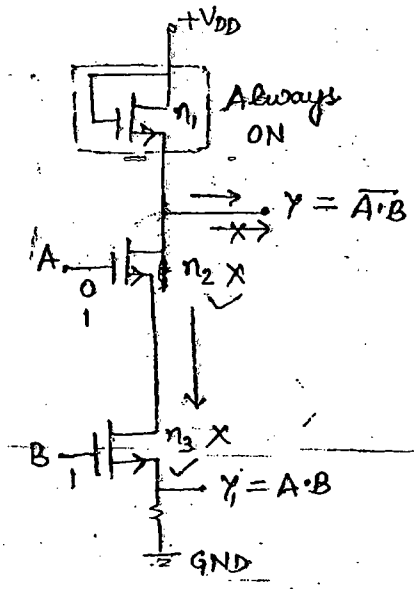


NOT



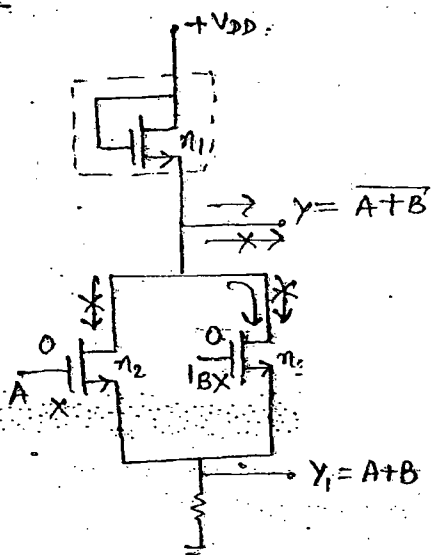
A	n_1	n_2	Y
0	✓	X	1
1	✓	✓	0

NAND



A	B	n_1	n_2	n_3	Y
0	0	✓	✗	✗	1
0	1	✓	✗	✓	1
1	0	✓	✓	✗	1
1	1	✓	✓	✓	0

NOR

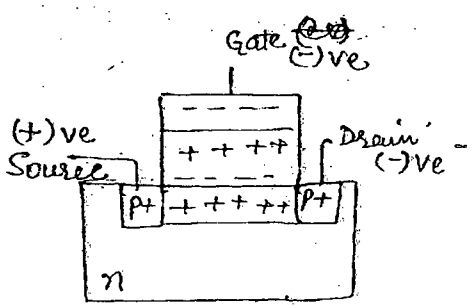


A	B	n_1	n_2	n_3	Y
0	0	✓	✗	✗	1
0	1	✓	✗	✓	0
1	0	✓	✓	✗	0
1	1	✓	✓	✓	0

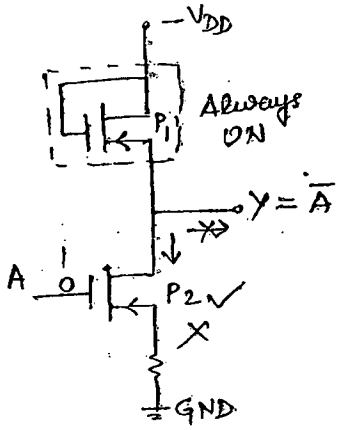
P- Mos

p-Mos ⇒ Gate ⇒ (+)ve ⇒ OFF

Gate ⇒ (−)ve ⇒ ON

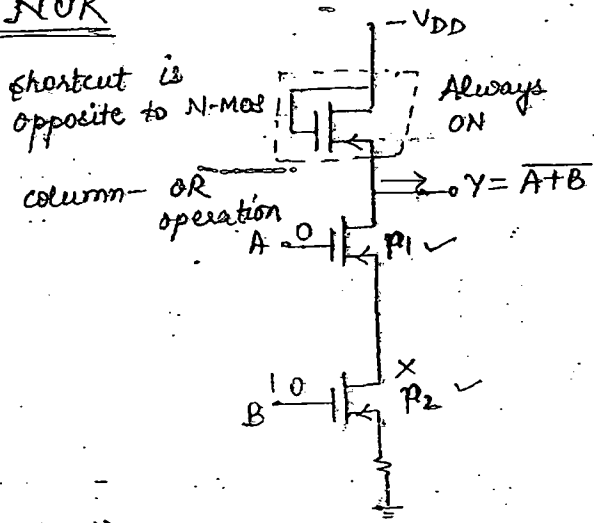


NOT

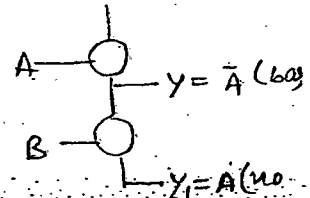


A	B	P_1	P_2	Y	
0		✓	✓	(GND) 1	
1		✓	✗	($-V_{DD}$) 0	

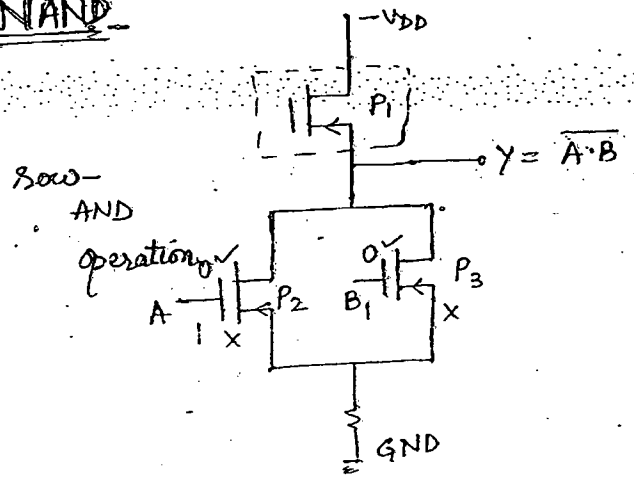
NOR



A	B	P_1	P_2	P_3	Y	
0	0	✓	✓	✓	1	GND
0	1	✓	✓	✗	0	$-V_{DS}$
1	0	✓	✗	✓	0	$-V_{DS}$
1	1	✓	✗	✗	0	$-V_{DD}$



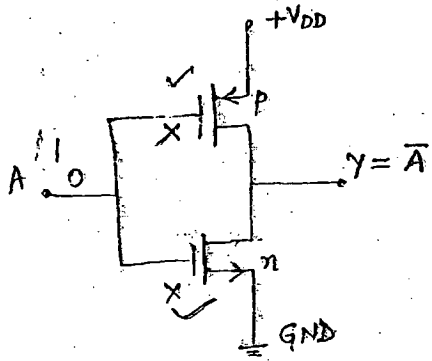
NAND



A	B	P_1	P_2	P_3	Y
0	0	✓	✓	✓	1
0	1	✓	✓	✗	1
1	0	✓	✗	✓	1
1	1	✓	✗	✗	0

C-MOS

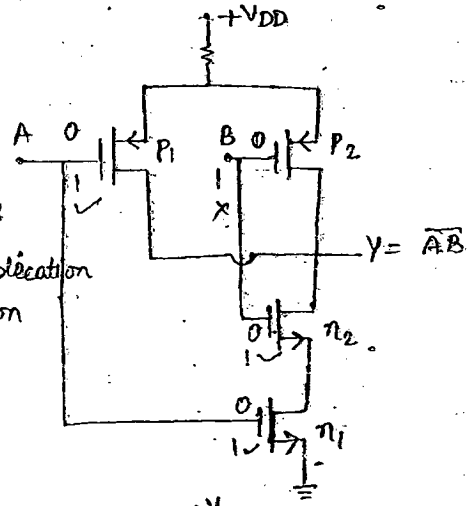
NOT



A	P	n	Y	
0	✓	✗	1	+VDD
1	✗	✓	0	GND

NAND

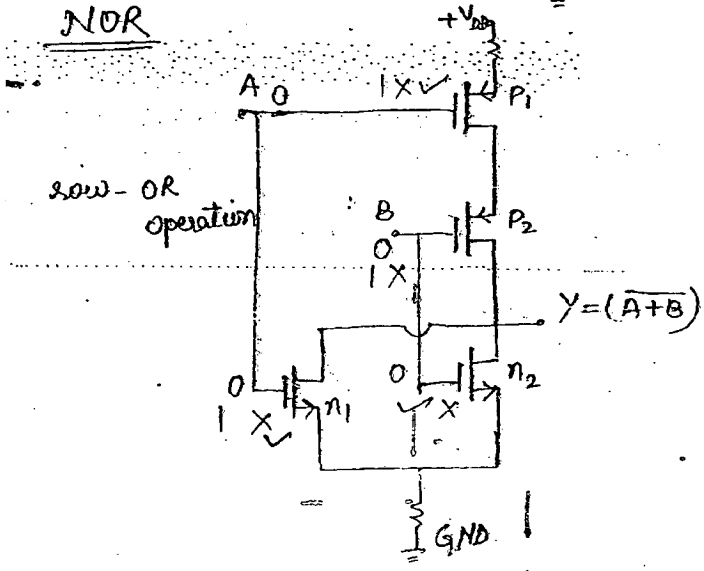
use shortcut seeing N-MOS column - multiplication AND operation



A	B	P1	P2	n1	n2	Y	
0	0	✓	✓	✗	✗	1	+VDD
0	1	✓	✗	✗	✓	1	+VDD
1	0	✗	✓	✓	✗	1	+VDD
1	1	✗	✗	✓	✓	0	GND

NOR

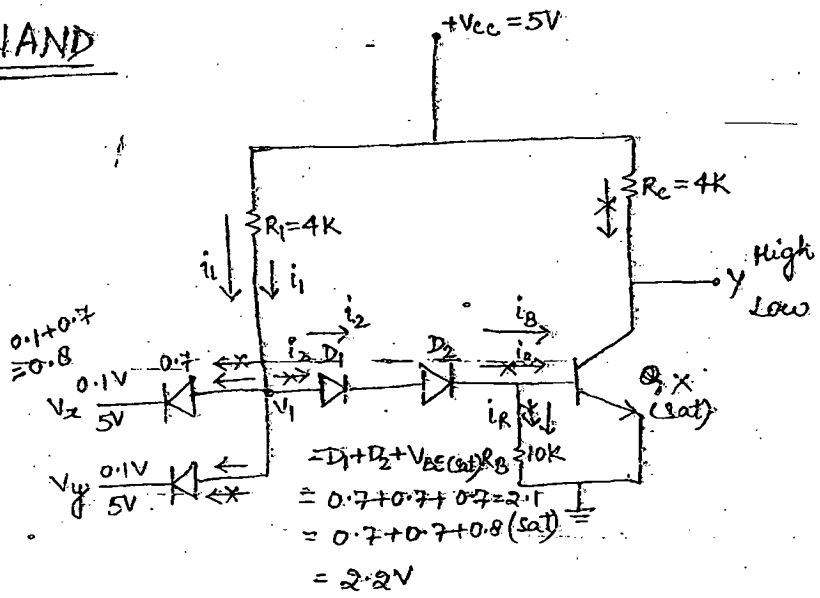
row - OR operation



A	B	P1	P2	n1	n2	Y	
0	0	✓	✓	✗	✗	1	+VDD
0	1	✓	✗	✗	✓	0	GND
1	0	✗	✓	✓	✗	0	GND
1	1	✗	✗	✓	✓	0	GND

DTL

NAND



Given;

$V_\gamma = 0.7V$

$V_{BE(ON)} = 0.7V$

$V_{BE(sat)} = 0.8V \quad , \beta = 25$

$V_{CE(sat)} = 0.1V$

Case (i): $V_x = 0.1V$, $V_y = 0.1V$.

$$i_1 = \frac{V_{cc} - V_1}{R_1} = \frac{5 - 0.8}{4K} = 1.05mA$$

$i_2 = i_B = i_R = 0$

$\Rightarrow$ Q is cut off

$\Rightarrow$ Y = High (5V)

Case (ii): $V_x = 0.1V$, $V_y = 5V$

Same as that of (i)

$\Rightarrow Y_g$ = High (5V)

Case (iii): $V_x = 5V$, $V_y = 0.1V$

Same as that of case (i) and Case (ii)

$\Rightarrow$ Y = High (5V)

Saturation condition

$$\frac{i_C'}{i_B} \leq \beta$$

$$\frac{1.22 + (1.05)}{0.62} \leq 25$$

$$N = 13.7$$

$$\boxed{N \simeq 13}$$

Case (ii) : Output of the driving gate is high

In this case, input diode of the loading gate is reverse biased and there exist a reverse saturation current which passes through R_c of driving gate from $+V_{cc}$. By N number of Loads, we get much voltage drop across R_c and output alters from high value which leads to improper operation so even in this case also there is a limit on Fan out.

NOTE : Among the above two cases, the least value should be taken for the overall fanout. Also, the output low case is preferable.

Procedure to find Noise Margin

Case (i) : Output of the driving gate is low

In this case, the other section of the loading gate requires $D_1 + D_2 + V_{BE(ON)} = 0.7 + 0.7 + 0.7 = 2.1V$ so the difference of voltage $2.1 - 0.8 = 1.3V$ is required the acceptable fluctuating voltage so the noise margin is 1.3V.

Case (ii) : Output of the driving gate is high

In this case also, the noise margin will be calculated and among the above two cases the least value is preferable for overall value of Noise Margin.

Power Dissipation $P.D = V_{cc} \cdot I_{cc}$

$$P.D = \frac{P.D(0) + P.D(1)}{2}$$

~~NOTE~~ :-

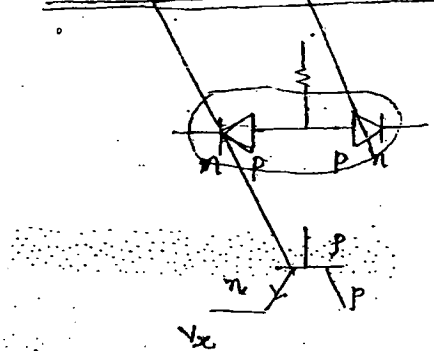
Q. ★ What happens if one of the diodes D_1 or D_2 is removed?

- Fan out increases
- Noise margin reduces

Q. ★ What happens if one more diode D_3 is added?

- Fan out reduces
- Noise Margin increases

TTL (Transistor Transistor Logic)

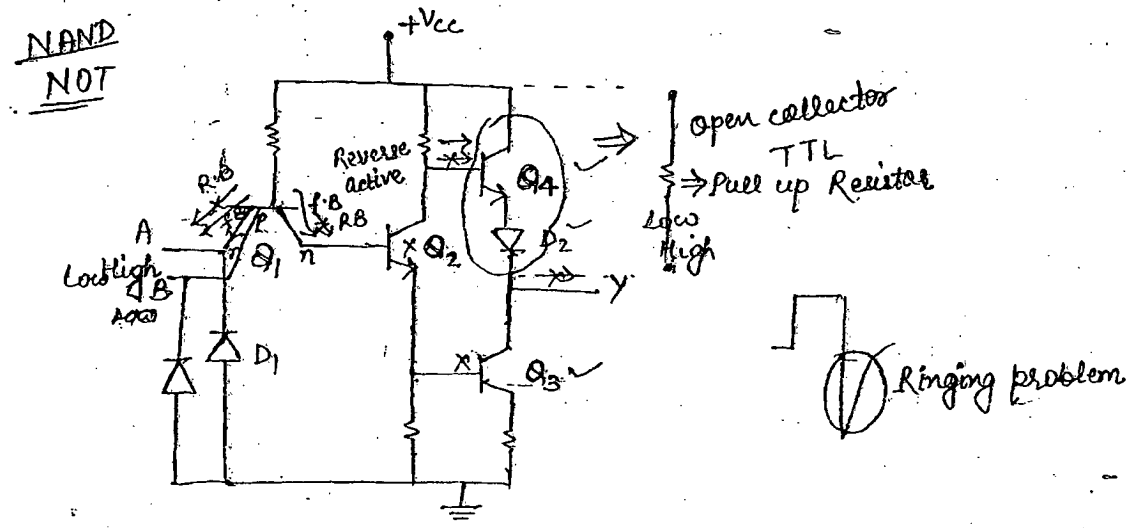


★★ NOTE :- The tabular form related to the values of fan out, propagation delay, noise margin, power dissipation etc is present in theory book.

★★ Without reading this, one should not go for ! exam.

TTL

NAND
NOT



The Purpose of D_1: It avoids the ringing problem. The negative voltage fluctuations can be grounded by D_1 and the input transistor will be safe.

The Purpose of D_2: It is used for the proper switching voltages at the output.

* TTL Technologies are three types.

1) Totem Pole TTL:- In this case, output transistors are not ON simultaneously. The above discussed example is Totem Pole TTL only.

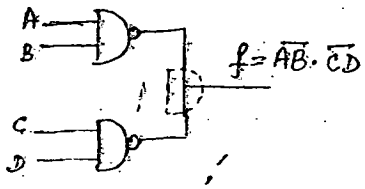
2) Open Collector TTL: In this case, the collector section of output transistor is a pull up resistor is used for proper switching output.

NOTE :- Pull up resistor also avoids voltage fluctuations of power supply.

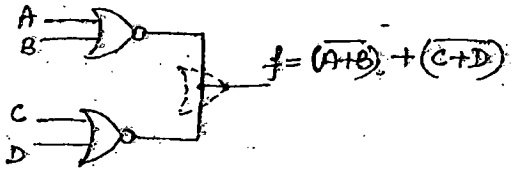
** Open collector TTL allows wired logic.

Wired Logic

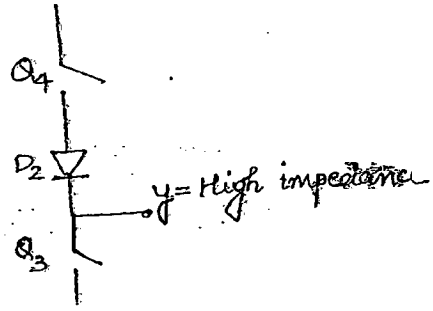
Wired AND Logic



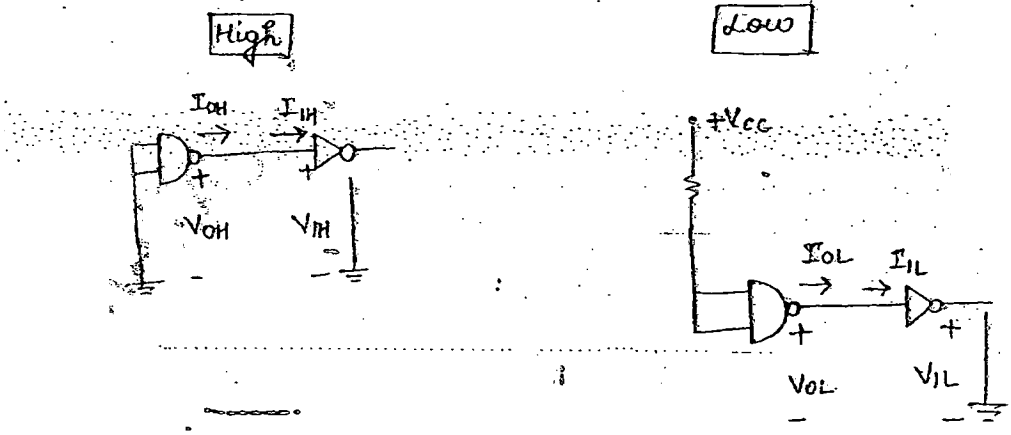
Wired OR Logic

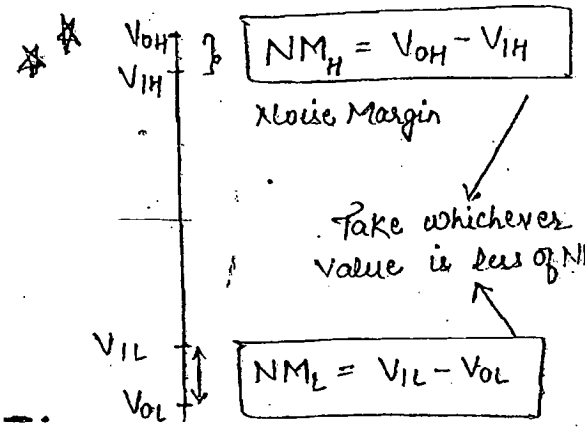


⇒ **Tristate Logic** :- In the case of Tristate logic, there will be 3 states, Logic 0, Logic 1 and High Impedance state.



Voltage and Current Parameters





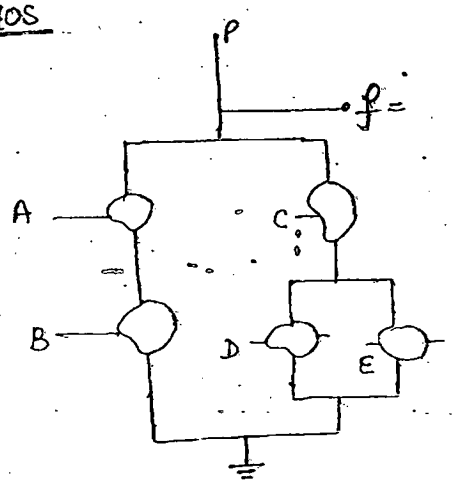
$$\text{fan out} = \frac{I_{OH}}{I_{IH}}$$

$$\text{fan out} = \frac{I_{OL}}{I_{IL}}$$

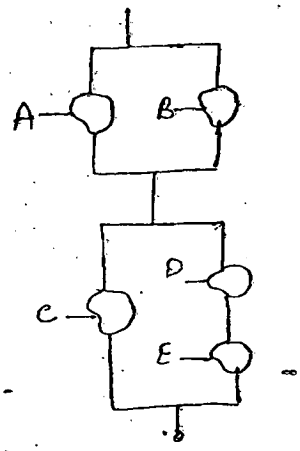
→ Take whichever value is less of fan out.

Q Design $f = \overline{AB + C(D+E)}$

Using NMOS



Using PMOS



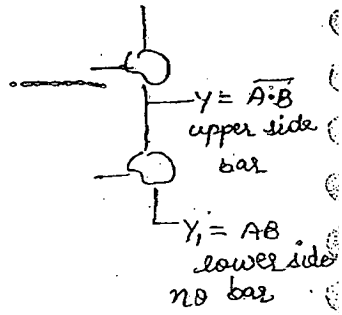
Using CMOS

Combine both NMOS & PMOS Ckt

Shortcut: To know which gate it is

applicable to all except PMOS { column – multiplication AND operation; row – OR operation

PMOS { column – OR operation; row – AND operation



2's Complement Arithmetic

1) Add the 2's complement of subtrahend to the minuend.
2) Check the MSB bit (sign bit).
3) If sign bit is zero (0), answer is true binary.
   If MSB bit is one (1), answer should be in 2's compliment form.
4) If the carry generated, ignore the carry.

$x \rightarrow$ minuend
$-y \rightarrow$ subtrahend

Case (i)

$+9 \rightarrow 0\ 1001$
$-4 \rightarrow 1\ 1100$
$1\ 0\ 0101 \Rightarrow (+5)$

Ignore carry
Sign bit (0)

$+4 \rightarrow 10100 \rightarrow$ sign mag
$\downarrow$
1 1100 → 2's Comp

$-9 \rightarrow 1\ 1001 \rightarrow$ sign mag
1 0111 → 2's Comp

Case (ii)

$-9 \rightarrow 1\ 0111$
$+4 \rightarrow 0\ 0100$
$1\ 1011$

Sign bit (1) -ve
2's comp
1 0101 = −5
Find 2's comp of result

Case (iii)

$+9 \rightarrow 0\ 1001$
$+4 \rightarrow 0\ 0100$
$0\ 1101 = +13$

Sign bit (0) +ve

Case (iv)

$-9 \rightarrow 10111$

$-4 \rightarrow 11100$

Ignore carry ← 1 10011

Sign bit

(1)11101 = −13

Sign bit

Case (v)

$-9 \rightarrow 10111$

$+9 \rightarrow 01001$

(0)00000 = 0

Sign bit

1's Complement Arithmetic

Step 1) Add the 1's complement of subtrahend to the minuend.

2) If the carry is generated make it end around carry. (i.e add carry to LSB)

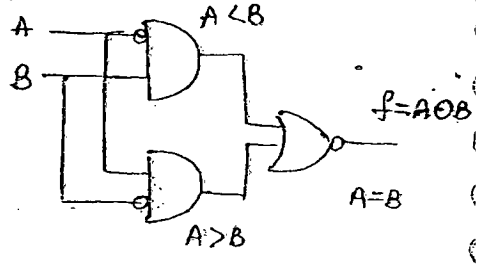
3) Then check the MSB bit (Sign bit) :

4) If MSB bit is zero, answer is true binary.

5) If MSB bit is one, answer should be in 1's complement form.

One bit Comparator

		A'·B	A⊙B	A·B'
A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0



for 4 bit :-

$A \rightarrow A_3\ A_2\ A_1\ A_0$

$B \rightarrow B_3\ B_2\ B_1\ B_0$

⇒ if $A_3 > B_3$: $1\ \times\ \times\times$, $0\ \times\ \times\times$ → $A>B$

⇒ if $A_3 < B_3$: $0\ \times\ \times\times$, $1\ \times\ \times\times$ → $A<B$

⇒ if $A_3 = B_3$: $A_3\ 1\ \times\ \times$, $B_3\ 0\ \times\ \times$ → $A>B$

4 bit Comparator

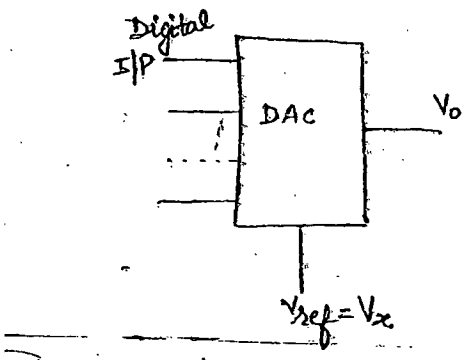
$A_3\ B_3$	$A_2\ B_2$	$A_1\ B_1$	$A_0\ B_0$	$A<B$	$A=B$	$A>B$
$A_3 > B_3$	×	×	×	0	0	1
$A_3 < B_3$	×	×	×	1	0	0
$A_3 = B_3$	$A_2 > B_2$	×	×	0	0	1
$A_3 = B_3$	$A_2 < B_2$	×	×	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	×	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	×	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	1	0

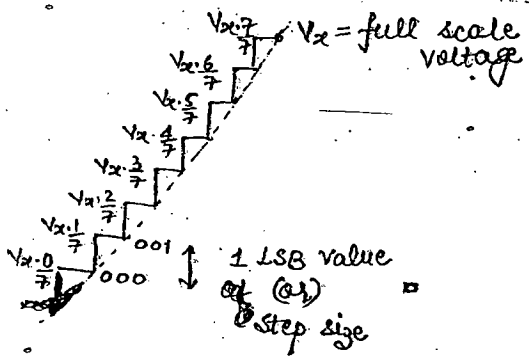
$A<B \Rightarrow A_3' \cdot B_3 + (A_3 \odot B_3)\cdot(A_2' \cdot B_2) + (A_3 \odot B_3)(A_2 \odot B_2)(A_1' \cdot B_1) + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0' \cdot B_0)$

$A=B \Rightarrow (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$

$A>B \Rightarrow A_3 \cdot B_3' + (A_3 \odot B_3)\cdot(A_2 \cdot B_2') + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \cdot B_1') + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \cdot B_0')$

DAC





$$V_o = K\left[2^{n-1} b_{n-1} + 2^{n-2} b_{n-2} + \ldots + 2^1 b_1 + 2^0 b_0\right]$$

Decimal equivalent binary data

1 1 0 1

$$\left[1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0\right]$$

decimal eq. binary data

Parameters of DAC

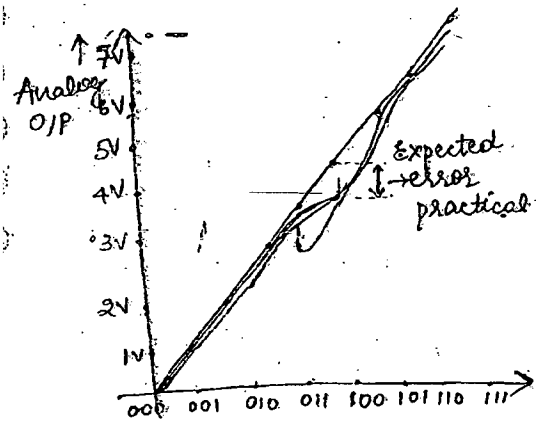
Number of steps $= 2^n - 1$

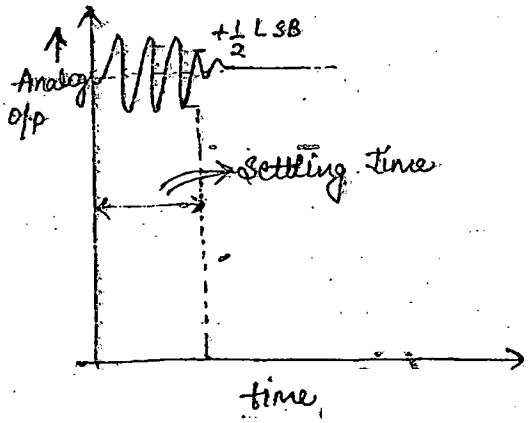
1 LSB value (or) step size $= V_x \cdot \frac{1}{7}$

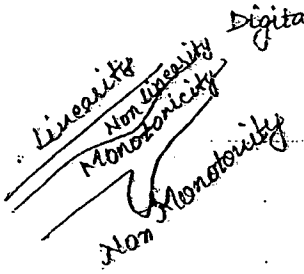
$$= \frac{V_{f.s}}{2^n - 1}$$

$$= \frac{\text{full scale voltage}}{\text{no. of steps}}$$

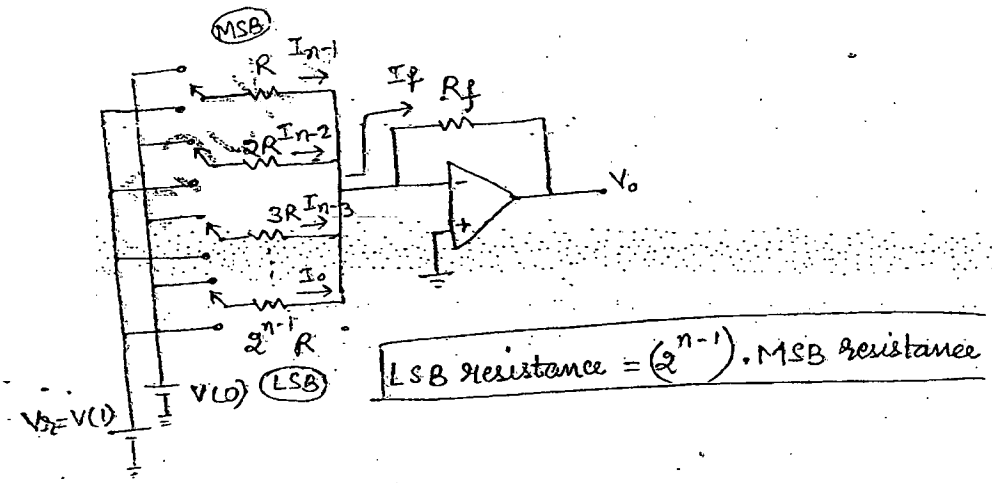
$$\% \text{ Resolution} = \frac{1}{2^n - 1} \times 100\%$$







Weighted Resistor type DAC



$$V_o = -I_f \cdot R_f$$

$$= -R_f[I_0 + I_1 + I_2]$$

$$= -R_f\left[\frac{V_2}{2^2R} + \frac{V_2}{2^1R} + \frac{V_2}{2^0R}\right]$$

$$= -\frac{V_2 R_f}{R}\left[\frac{1}{2^2} + \frac{1}{2^1} + \frac{1}{2^0}\right]$$

$$V_o = \frac{-V_R \cdot R_f}{R}\left[\frac{1}{2^{n-1}} + \cdots\cdots + \frac{1}{2^1} + \frac{1}{2^0}\right]$$

$$= \frac{-V_R}{2^{n-1}} \cdot \frac{R_f}{R}\left[1 + 2^1 + \cdots\cdots + 2^{n-1}\right]$$

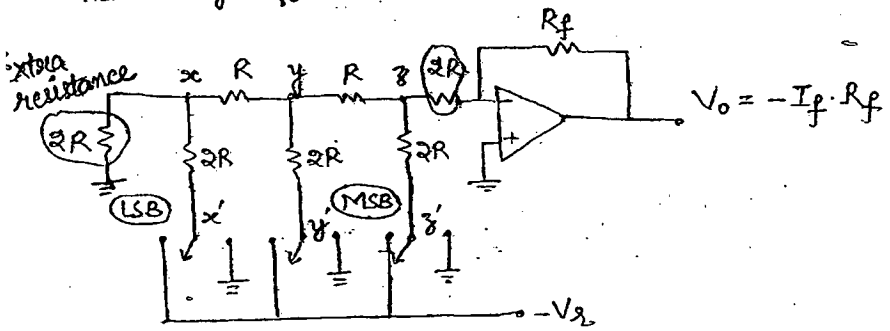
$$V_o = \left|\frac{V_R}{2^{n-1}} \cdot \frac{R_f}{R}\right|\left[2^{n-1} b_{n-1} + 2^{n-2} b_{n-2} + \cdots\cdots + 2^1 b_1 + 2^0 b_0\right]$$

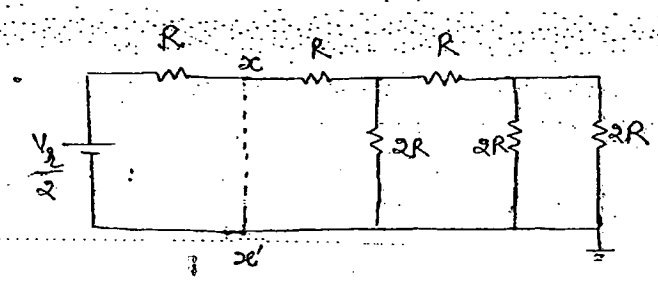
$$\boxed{V_o = K\left[2^{n-1} b_{n-1} + \cdots\cdots + 2^1 b_1 + 2^0 b_0\right]}$$

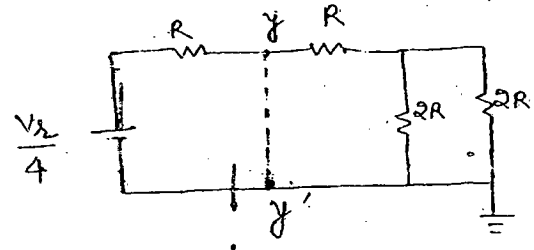
where $\boxed{K = \left|\frac{V_R}{2^{n-1}} \cdot \frac{R_f}{R}\right|}$

R-2R Ladder type DAC

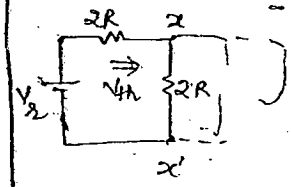
(Inverting type)

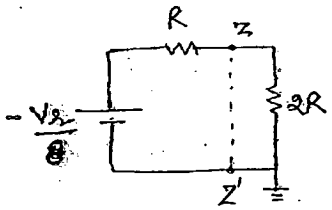


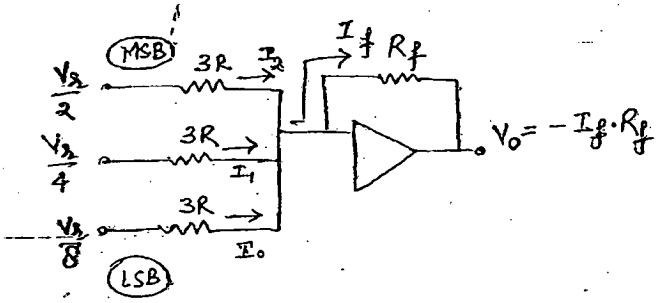




$$V_{Th} = \frac{V_R \cdot 2R}{2R + 2R} = \frac{V_R}{2}$$







$$V_o = -I_f \cdot R_f$$

$$V_o = -R_f [I_0 + I_1 + I_2]$$

$$V_o = -R_f \left[\frac{V_2}{8(3R)} + \frac{V_2}{4(3R)} + \frac{V_2}{2(3R)} \right]$$

$$V_o = \frac{-R_f V_2}{3R} \left[\frac{1}{2^3} + \frac{1}{2^2} + \frac{1}{2} \right]$$

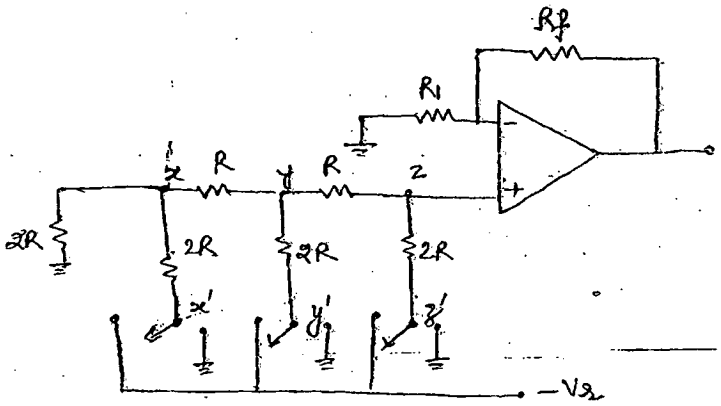
$$V_o = -\frac{R_f \cdot V_2}{3R} \left[\frac{1}{2^n} + \frac{1}{2^{n-1}} + \cdots + \frac{1}{2^1} \right]$$

$$V_o = \left[\frac{-V_2}{2^n} \right] \left[\frac{R_f}{3R} \right] \left[2^{n-1} + 2^{n-2} + \cdots + 2^1 + 2^0 \right]$$

$$V_o = \left[\frac{V_2}{2^n} \right] \left[\frac{R_f}{3R} \right] \left[2^{n-1} b_{n-1} + \cdots + 2^1 b_1 + 2^0 b_0 \right]$$

$$\boxed{V_o = \left| \frac{V_2}{2^n} \right| \left(\frac{R_f}{3R} \right) \left[\sum_{i=0}^{n-1} 2^i b_i \right]}$$

(Non. Inverting Type)



$$V_o = \left|\frac{V_s}{2^n}\right| \left(1 + \frac{R_f}{R_1}\right) \left(\sum_{i=0}^{n-1} 2^i b_i\right)$$

Workbook

Pg 69 ADC & DAC

6) $V_o = \left|\frac{V_s}{2^n}\right| \left(1 + \frac{R_f}{R_1}\right) \left(\sum_{i=0}^{n-1} 2^i b_i\right)$

$= \left(\frac{1}{2^4}\right) \left(1 + \frac{7}{1}\right) \left(1010_{(2)} = 10\right)$

$= \left(\frac{1}{16}\right)(8)(10) = \left(\frac{1}{16}\right)(8)(10)$

$= 5V$

2) $V_o = K[\text{Dec eq. Binary data}]$

$V_o = K[11011011_{(2)} = 219]$

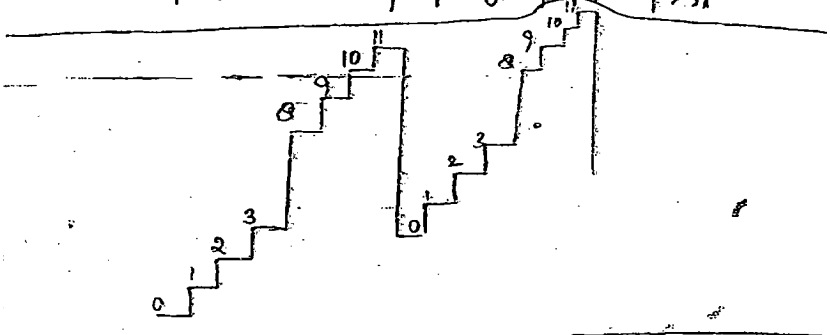
$V_o = K(219)$

$20 = K[11111111_2 = 255]$

$20 = K \times 255$

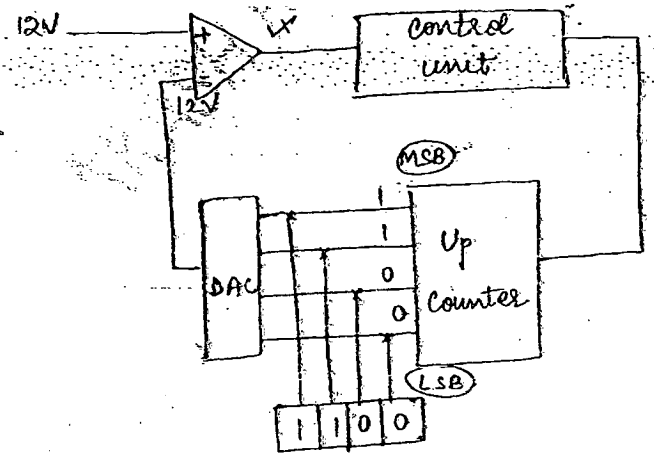
Q)

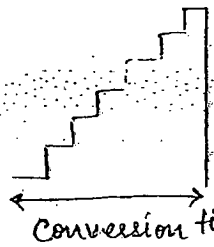
	Q_2	Q_1	Q_0	D_3	D_2	D_1	D_0	
0	0	0	0	0	0	0	0	→ 0
1	0	0	1	0	0	0	1	→ 1
2	0	1	0	0	0	1	0	→ 2
3	0	1	1	0	0	1	1	→ 3
4	1	0	0	1	0	0	0	→ 8
5	1	0	1	1	0	0	1	→ 9
6	1	1	0	1	0	1	0	→ 10
7	1	1	1	1	0	1	1	→ 11



ADC

Digital Ramp type ADC (or) Counter type ADC

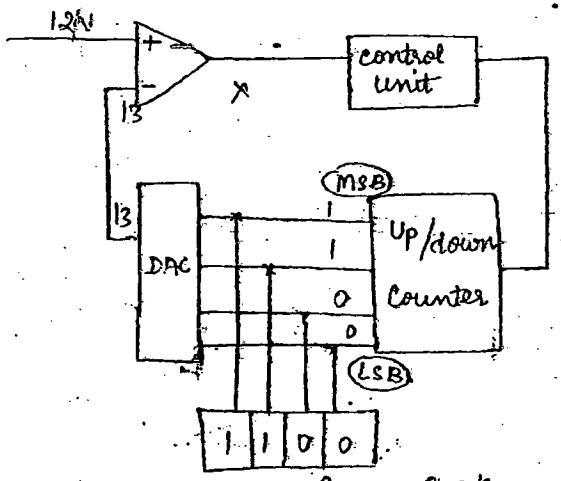




$t_c(max) = (2^n - 1)$ clk cycles

** Operation :- Digital Theory Book

Successive Approximate type ADC

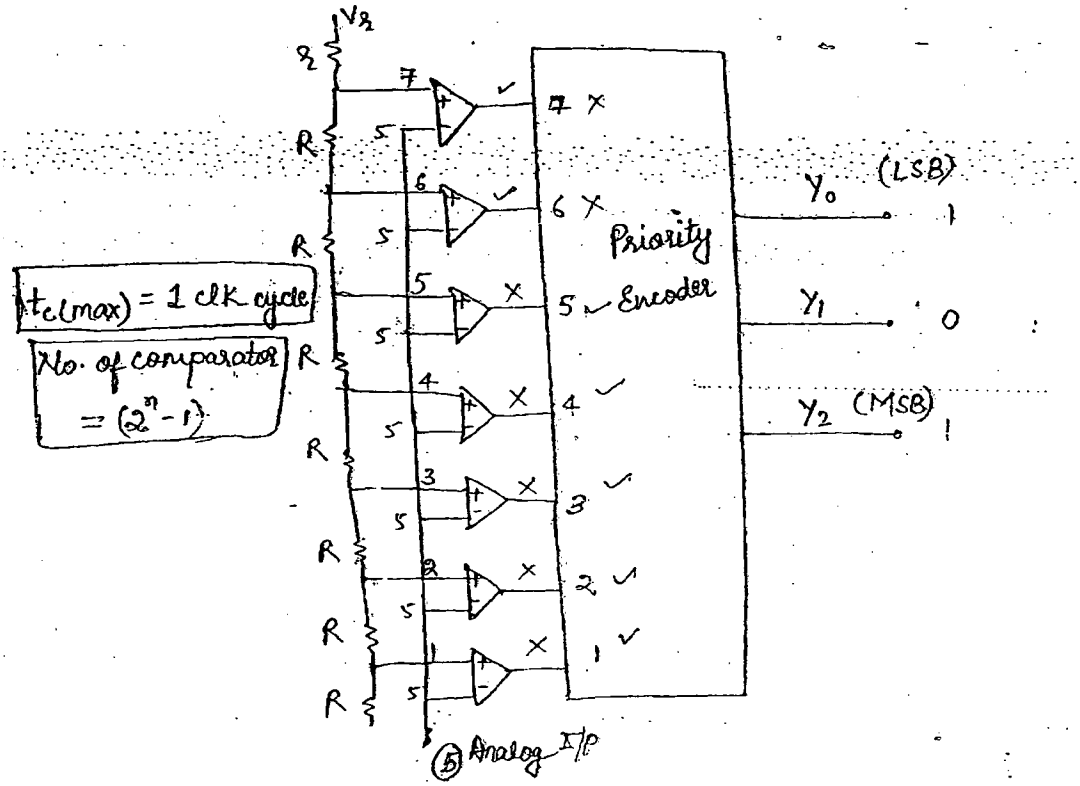


* Keep MSB as 1 in starting (always)

$t_{c(max)}$ = N. clk cycles

operation : Digital Theory Book

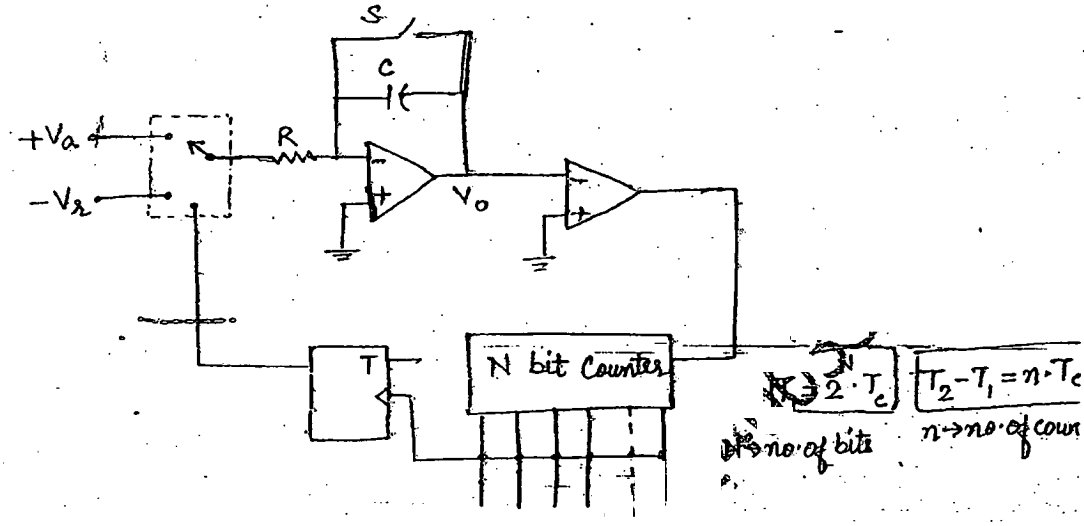
Flash type ADC (or) Simultaneous type ADC (or) Parallel type ADC

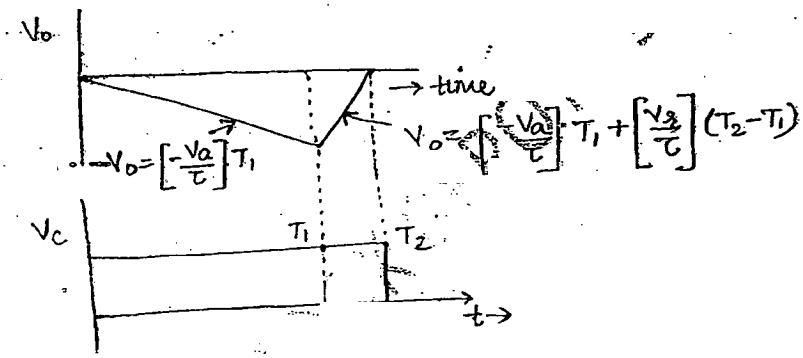


$t_{c(max)}$ = 1 clk cycle

No. of comparator = $(2^n - 1)$

Dual Slope ADC





$$V_o = -\frac{1}{\tau}\int_0^{T_1} V_a\,dt$$

$$V_o = \left[-\frac{V_a}{\tau}\right]T_1 \quad \text{—— (1)}$$

$$V_o = \left[-\frac{V_a}{\tau}\right]T_1 + \left\{-\frac{1}{\tau}\int_{T_1}^{t}(-V_r)\,dt\right\}$$

$$V_o = \left[-\frac{V_a}{\tau}\right]T_1 + \left[\frac{V_r}{\tau}\right](t - T_1)$$

at $t = T_2$, $V_0 = 0$

$$0 = \left[\frac{-V_a}{\tau}\right] T_1 + \left[\frac{V_2}{\tau}\right] [T_2 - T_1]$$

$$\frac{V_a}{\cancel{\tau}} T_1 = \frac{V_2}{\cancel{\tau}} [T_2 - T_1]$$

$$V_a [2^N \cdot \cancel{T_c}] = V_2 \cdot (n \cdot \cancel{T_c})$$

$$V_a = \frac{V_2}{2^N} \cdot n$$

if $V_2 = 2^N$

$$\boxed{V_a = n}$$

$$t_{c(max)} = T_2$$

$$t_{c(max)} = T_1 + n T_c$$

$$= 2^N T_c + n T_c$$

$$= (2^N + n) T_c$$

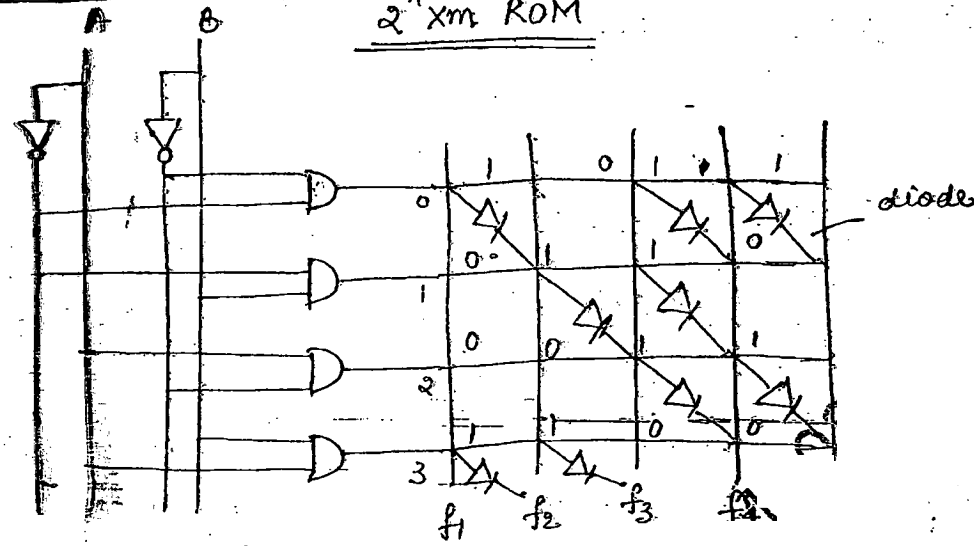
$$= (2^N + 2^N) T_c$$

$$= 2 \cdot 2^N T_c$$

$$\boxed{t_{c(max)} = (2^{N+1}) T_c}$$

MEMORIES

$2^n \times m$ ROM



16

4×4

$2^2 \times 4$

$2^n \times m$

n → select line

m → data line

$f_1 = \Sigma m(0,3)$

$f_2 = \Sigma m(1,3)$

$f_3 = \Sigma m(0,1,2)$

$f_4 = \Sigma m(0,2)$

3 bit Squarer

$x^2 = 49$

32	16	8	4	2	1
1	1	0	0	0	0
f_6	f_5	f_4	f_3	f_2	f_1

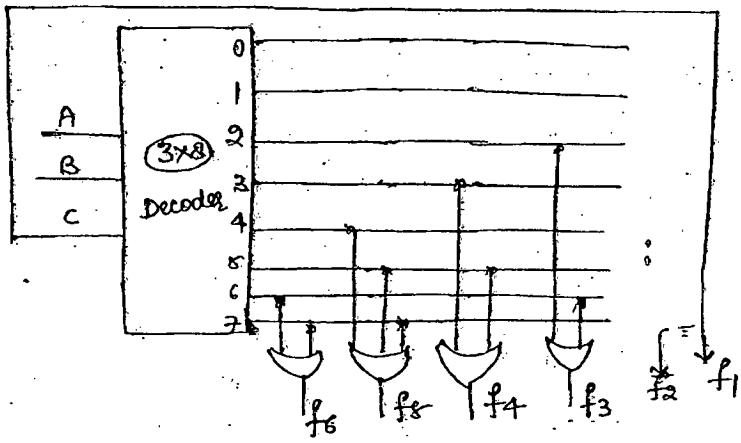
Size of ROM $= 2^n \times m$

$= 2^3 \times 6$

$= 48$

	A	BC	f_6	f_5	f_4	f_3	f_2	f_1	
0	0	00	0	0	0	0	0	0	'0'
1	0	01	0	0	0	0	0	1	'1'
2	0	10	0	0	0	1	0	0	'4'
3	0	11	0	0	1	0	0	1	'9'
4	1	00	0	1	0	0	0	0	'16'
5	1	01	0	1	1	0	0	1	'25'
6	1	10	1	0	0	1	0	0	'36'
7	1	11	1	1	0	0	0	1	'49'

~~Reduced ROM~~ ROM ⇒ Decoder + Encoder



Reduced ROM

$= 2^n \times m$

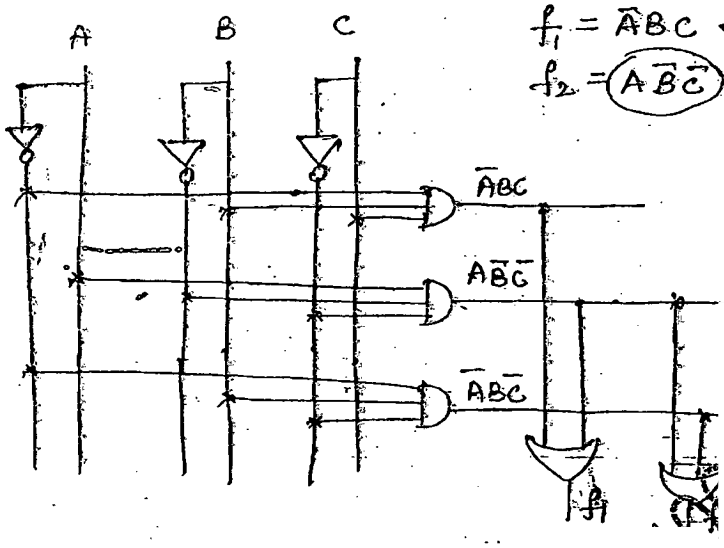
$= 2^3 \times 4$

$= 8 \times 4$

$= 32$

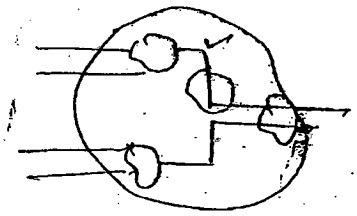
$f_1 = \bar{A}BC + A\bar{B}\bar{C}$

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



	AND Array	OR Array
ROM ⇒	fixed	programmable
PLA ⇒	programmable	programmable
PAL ⇒	programmable	fixed

Hazard



When time delay of both i/p ckts is not same.

Hazard

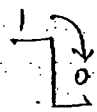
Because of different propagation delays at different parts of the circuits, the output of the circuit may not be accurate known as Hazard in the ckt. It can be eliminated by Redundant group in the circuit.

Glitch:- For temporary interval of time we get wrong result known as glitch in the circuit.

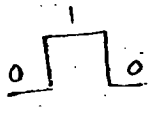
Hazards are three types:

Static '0'

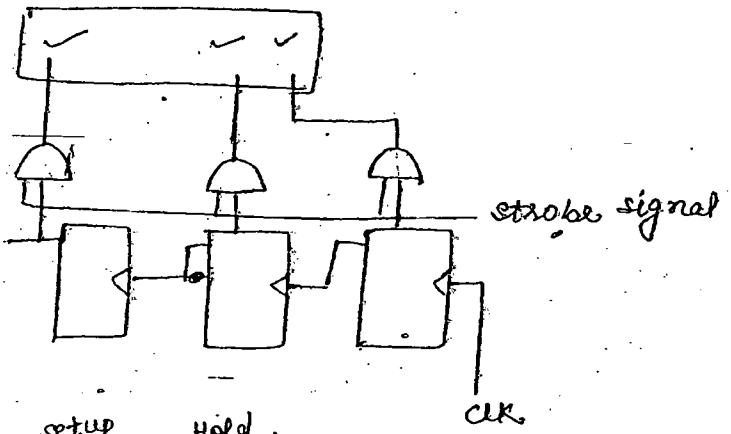
Static '1'

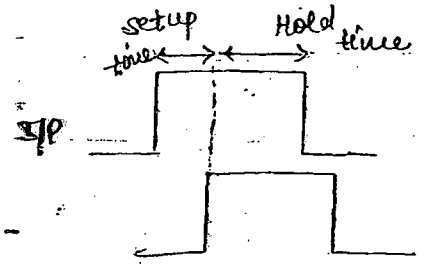


Dynamic



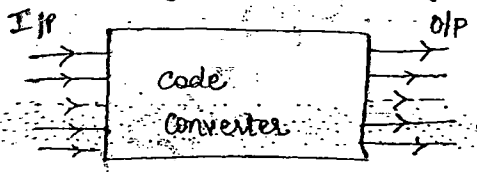
Strobe Signal



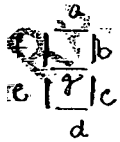


→ for calculating propagation delay, set up time should be considered

Seven Segment Display



B·C·D



I/P B·C·D A B C D	Seven Segment a b c d e f g	
0 0 0 0	1 1 1 1 1 1 0	'0'
0 0 0 1	0 1 1 0 0 0 0	'1'
0 0 1 0		
0 0 1 1		

Master slave J-K Flip Flop

It consists of two section, Master and slave. They triggered from the same clock generator but the second section is triggered through NOT gate so both sections can not be ON simultaneously. So the final output occurence time, the input section is in the OFF state so there is no repetition of toggle & race around problem is avoided.

NOTE: External feedback is taken from section second section only.