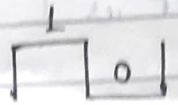
Logic Design

Books :-

→ Morris Mano

Logic = Validity of Thoughts



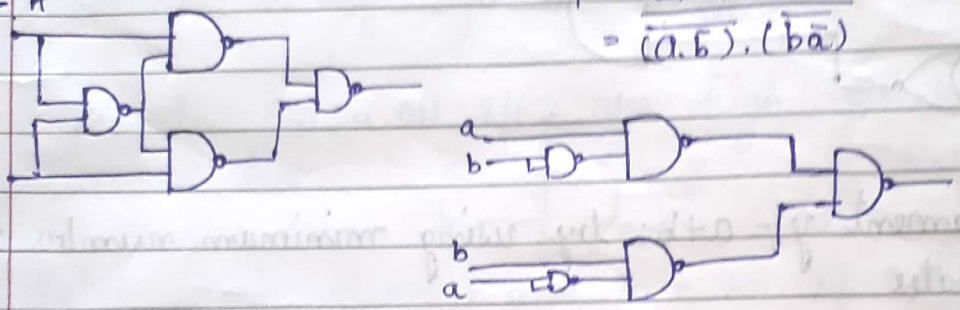
Q. Implement EX-OR gate by using NAND gate.

Ans- A

B

$Y = a\bar{b} + b\bar{a}$

$= \overline{(a.\bar{b})}.\overline{(b\bar{a})}$

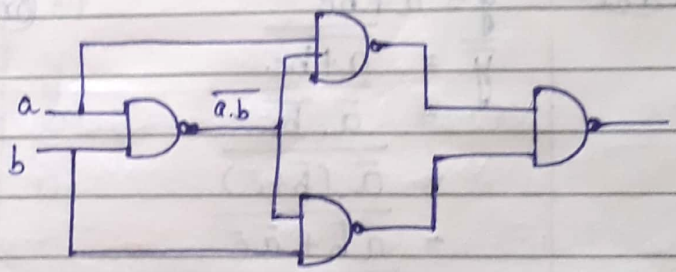


Q. $Y = (a+b)(\bar{a}+\bar{b})$

Ans- $Y = (a+b)(\overline{a.b})$

$= [a.(\overline{a.b}) + b.(\overline{a.b})]$

$= \overline{a.(\overline{a.b})}.\overline{b.(\overline{a.b})}$



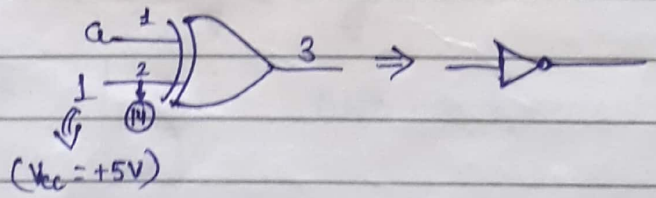
Q. Implement NOT gate by using EX-OR gate only.

Ans- $Y = a\bar{a} + b\bar{a} \Rightarrow \bar{a}$

If $b = 1$

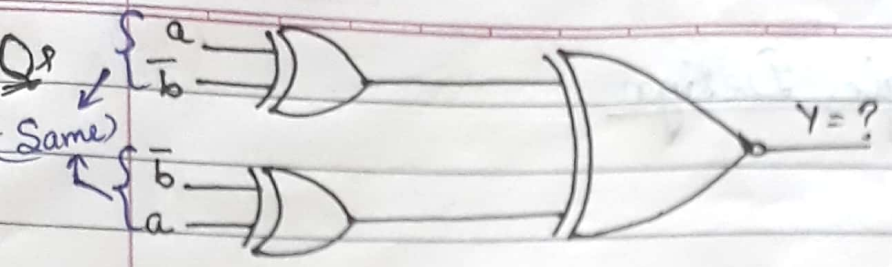
$\Rightarrow a.\bar{1} + 1.\bar{a}$

$\Rightarrow \bar{a}$

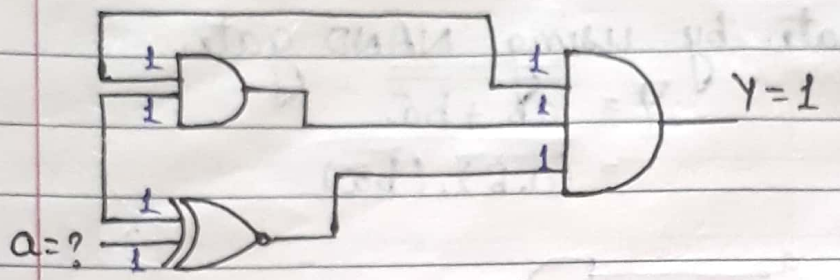


* NAND ↔ NOR

* X-OR ↔ X-NOR

Q. 

Ans- In EX-NOR, O/P is 1 when both I/Ps are same. So, Y = 1.

Q. 

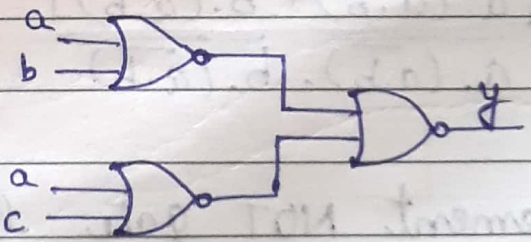
Q. Implement $y = a + bc$ by using minimum number of NOR gates.

Ans- $y = a + bc$

$$\bar{y} = \overline{a + bc}$$
$$= \bar{a} \cdot \overline{bc}$$
$$= \bar{a}(\bar{b} + \bar{c})$$
$$= \bar{a}\bar{b} + \bar{a}\bar{c}$$
$$= \overline{(\bar{a}+b)} + \overline{(\bar{a}+c)}$$

(or)

$$y = a + bc$$
$$= (a+b)(a+c)$$
$$\bar{y} = y = \overline{(\bar{a}+b)} + \overline{(\bar{a}+c)}$$



Logic Gates

Logic Gates are the basic building blocks of any digital circuit. It is of 3 types :-

1. Basic Gates - AND, OR, NOT are called basic gates because with the help of combination of these three gates, we can implement any boolean function or logical expression or digital circuit.

(i) AND Gate - When all I/P's are high, O/P becomes high, otherwise low.

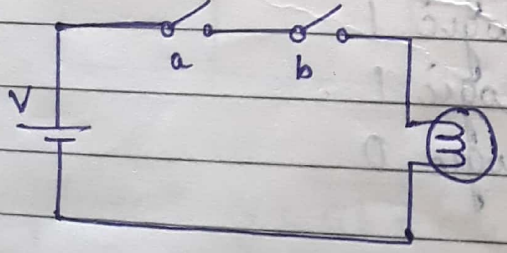
* Total combinations = 2^n ; n = No. of I/Ps

Truth Table →

I/P		O/P
a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

Q. Implement AND Gate by using switch.

Ans -



AND → Series

Duality Theorem

0 ⟷ 1

+ ⟷ .

While unchanging variable.

Types of Logic

It's of 2 types:-

1. Positive Logic
2. Negative Logic

1. Positive Logic - Higher of two voltage is logic 1 & lower of two voltage is logic 0.

Eg- +ve logic

+5 V	logic 1
0	logic 0
+3.2 V	logic 1
-1.5 V	logic 0
-2.5 V	logic 0
-1.2 V	logic 1

2. Negative Logic - Lower of two voltage is logic 1 & higher of two voltage is logic 0.

Eg- -ve logic

+5 V	logic 0
0	logic 1
+3.2 V	logic 0
-1.5 V	logic 1
-2.5 V	logic 1
-1.2 V	logic 0

Q Determine -ve logic of AND gate. (a.b)

Ans-

a	b	y
1	1	1
1	0	1
0	1	1
0	0	0

} OR gate (a+b)

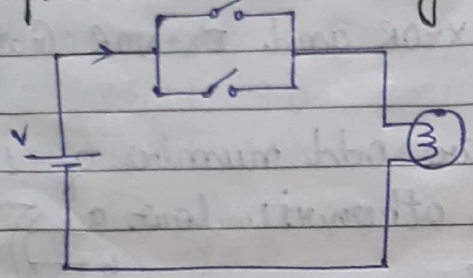
* Negative logic or dual of AND Gate is OR gate & vice-versa.

(ii) OR Gate – When all I/Ps are low, O/P is low, otherwise high.

(7432)

Qs Implement OR Gate by using switch.

Ans-



OR → Parallel

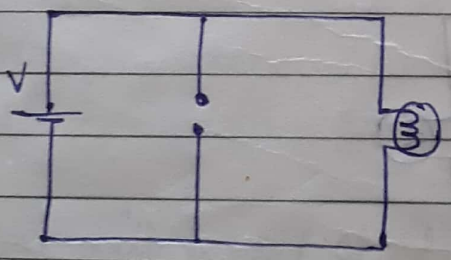
(iii) NOT Gate (Inverter) – It has only 1 I/P & 1 O/P.

Truth Table →

a	y
0	1
1	0

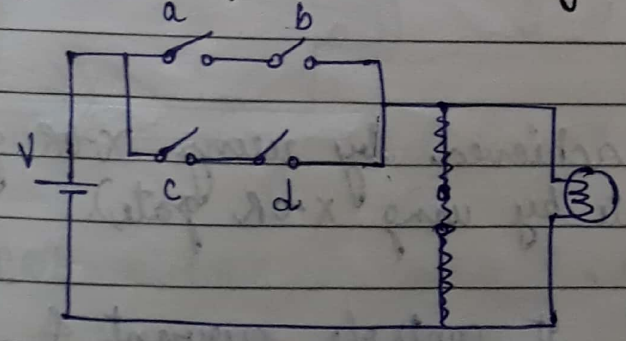
Qs Implement NOT Gate by using switch.

Ans-

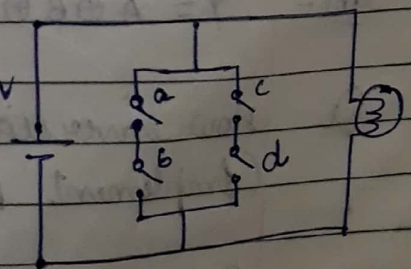


Qs Implement $y = \overline{ab + cd}$ by using switches.

Ans-

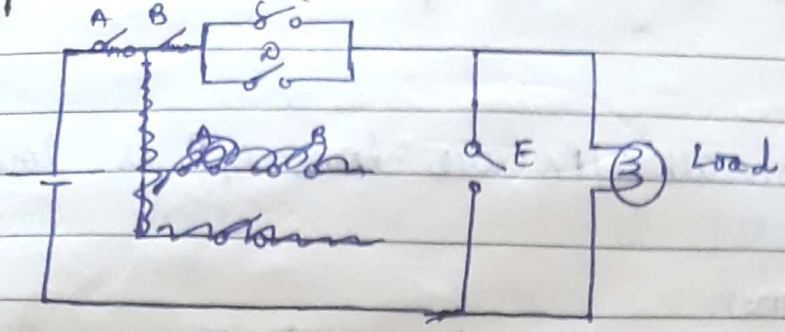


(OR)



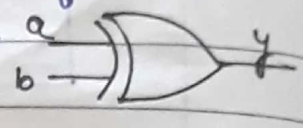
Q. Implement $Y=(AB)+(C\bar{D}).\bar{E}$

Ans-



2. Arithmatical Circuits - X-OR and EX-NOR Gate

(i) X-OR Gate - When there are odd number of 1's at input, o/p becomes high, otherwise low.



Eg - Staircase

Truth Table →

i/p a	i/p b	o/p y
0	0	0
0	1	1
1	0	1
1	1	0

SOP → o/p = 1

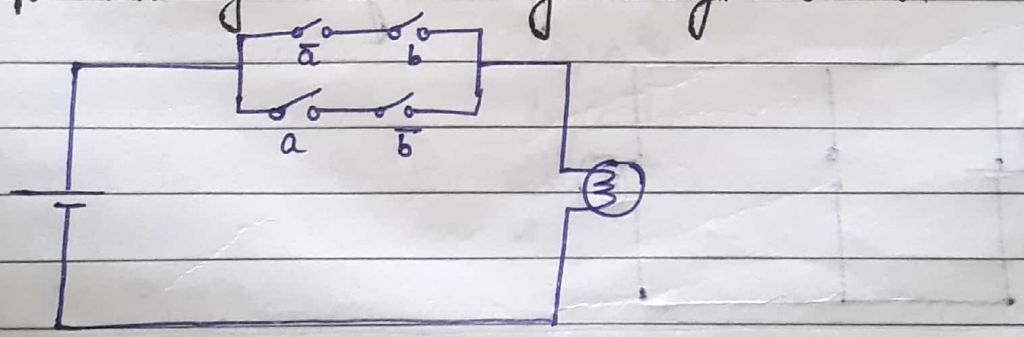
$x \to 1, \bar{x} \to 0$

$\Rightarrow y = \bar{a}b + a\bar{b}$ (SOP)

$y = (a+b).(\bar{a}+\bar{b})$ (POS)

Q. Implement $y = \bar{a}b + a\bar{b}$ by using Switch.

Ans -



Q. $Y = \sum^{(A,B,C)} m(1,2,4,7)$ 001 010 100 111 → odd no. of 1's

Ans - $Y = A \oplus B \oplus C$

Q. How inversion is achieved by using X-OR gate? (Implement NOT gate by using X-OR gate).

a	y
0	1
1	0

Buffer - 

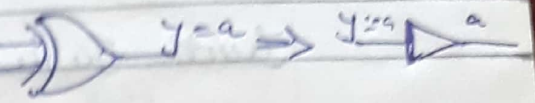
It controls current, & matches impedance.

Ans- $y = \bar{a}b + a\bar{b}$

If $b = 1$

$y = \bar{a}$

* If $b = 0$, then $y = a$ ⇒ Buffer



Q. Determine dual (-ve logic) of X-OR gate.

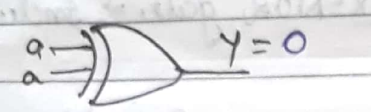
Ans- $y = \bar{a}b + a\bar{b}$

$= (a+b)(\bar{a}+\bar{b})$

$= ab + \bar{a}\bar{b}$ = X-NOR

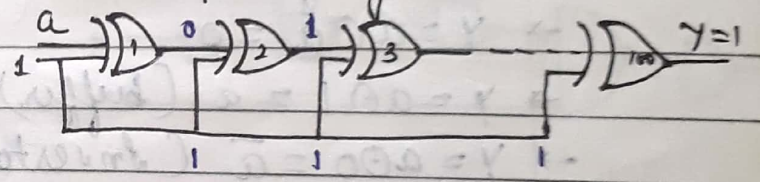
XOR ⇌ (Dual) X-NOR

Q.



Ans- $Y = a \oplus a = 0$

Q. If there is cascading of 100 X-OR gates, if o/p is 1, what will be value of a in the following circuit?



Ans- $a = 1$

Q. $Y = a \oplus a \oplus a \oplus \dots n$

Ans- If n = even, $Y = 0$

n = odd, $Y = a$

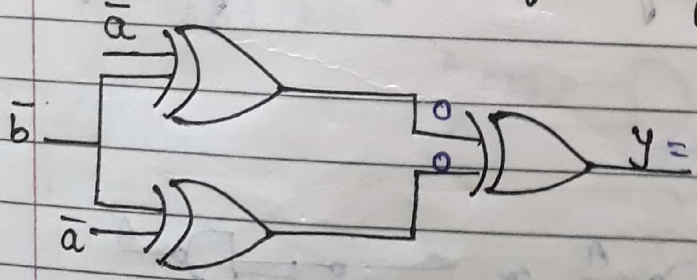
Q. $Y = a \oplus \bar{a}$

Ans- $Y = 1$

What will be o/p of gate no. 80 & gate no. 63?

Ans- G80 = 1, G63 = 0

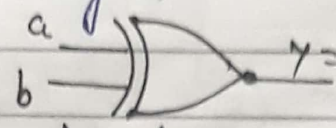
Q. What will be o/p of following ckt?



Q. If $a \oplus b = c$, $b \oplus c = a$, $c \oplus a = b$, $a \oplus b \oplus c = ?$

Ans- 0.

(ii) X-NOR Gate - It is the reverse process of X-OR Gate. When O/Ps become high, total no. of 1's at I/P are even, O/P is 1.



Truth Table →

a	b	y
0	0	1
0	1	0
1	0	0
1	1	1

Q. $Y = a \odot a = 1$

EX-NOR gate ⇒ Inclusive OR gate (Excluding)

Q. $Y = a \odot a \odot a \cdots n$

Ans- $Y = 1$, if n is even
$Y = 0$, if n is odd

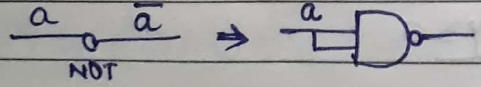
→ $Y = a \odot \bar{a} = 0$
→ $Y = a \odot 1 = a$ (buffer)
→ $Y = a \odot 0 = \bar{a}$ (Inverter)

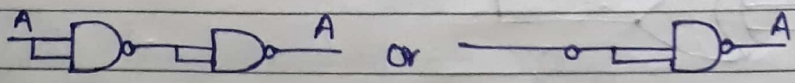
or Coincidence Ckt or Equivalent Detector

Uses → 1. Magnitude comparator

3. Universal Gates - NAND, NOR are universal gates, because we can implement any boolean function by using either NAND or NOR gate.

Implementation of any boolean func^n by using NAND gate

1. 

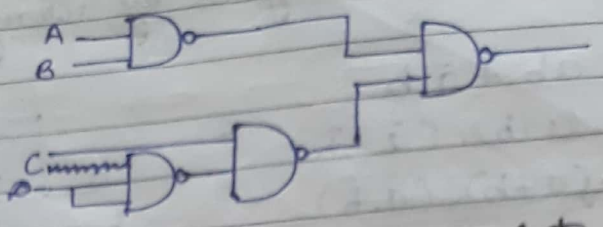
2. $\bar{\bar{A}} = A$ → 

3. Where '+' is available in any expression, by using DeMorgan's Theorem, convert it into '.'.

Q. $Y = AB + C\bar{D}$

Ans- $Y = \overline{\overline{AB + C\bar{D}}}$

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



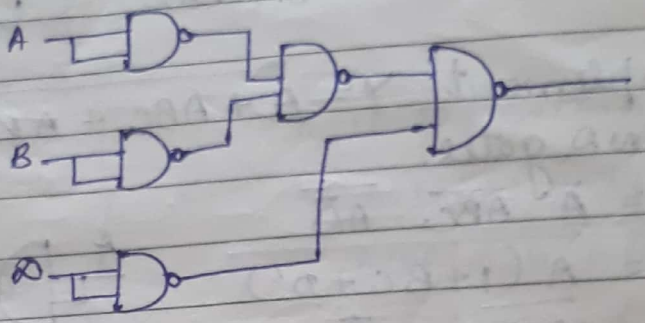
Q. Implement $Y = \overline{(A+B).\bar{D}}$ by using NAND Gate.

Ans- $Y = \overline{A\bar{D} + B\bar{D}}$

$Y = \overline{A\bar{D}}.\overline{B\bar{D}}$

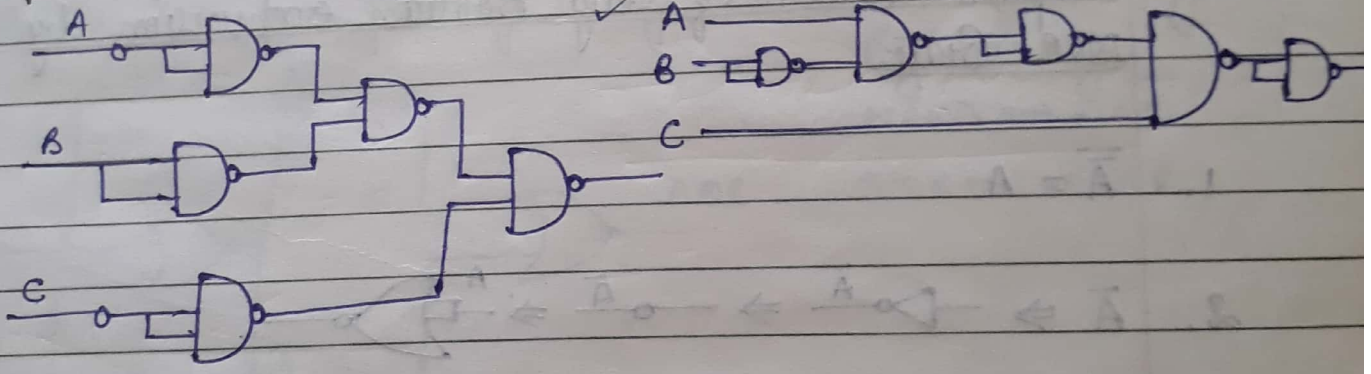
$Y = \overline{\overline{(A+B).\bar{D}}}$

$= \bar{A}.\bar{B}.\bar{D}$



Q. Implement $Y = A\bar{B}C$ by using 2-Input NAND gate.

Ans-

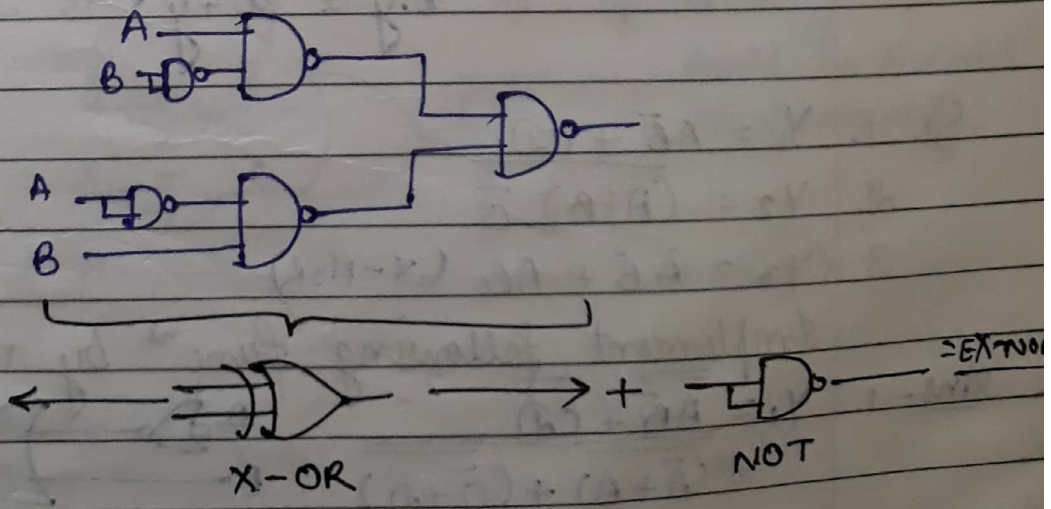


Q. Implement X-OR gate by using NAND gate only.

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

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

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



Q. Implement x-OR gate by using min^m no. of NAND gates.

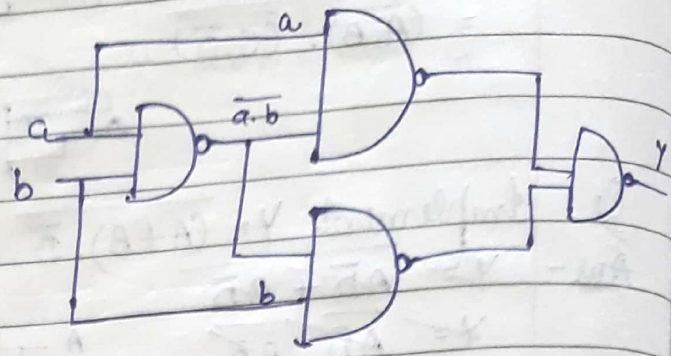
Ans- $Y = a\bar{b} + \bar{a}b$

$= (a+b).(\bar{a}+\bar{b})$

$= \overline{(a+b).(\bar{a}.\bar{b})}$

$= \overline{a.(\bar{a}.\bar{b})} + \overline{b.(\bar{a}.\bar{b})}$

$= a.\overline{(a.b)}.\ b.\overline{(a.b)}$

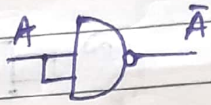


Q. Implement $Y = \overline{A + ABC + AB'}$ by using min^m no. of NAND gates.

Ans- $Y = \overline{\bar{A}.\overline{ABC}.\overline{A\bar{D}}}$

$= \overline{A(1+BC+D')}$

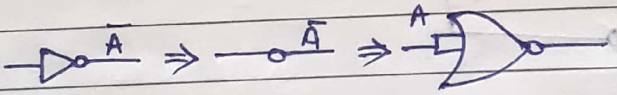
$= \overline{A.1} = \bar{A}$



$\begin{cases} 1+x=1 \\ 0.x=0 \end{cases}$

Implementation of any Boolean expression by using NOR Gate

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

2. $\bar{A} \Rightarrow$ 

3. Remove '.' by '+' by using DeMorgan's Theorem.

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

Q. 1. $Y_1 = A\bar{B} + C\bar{D}$

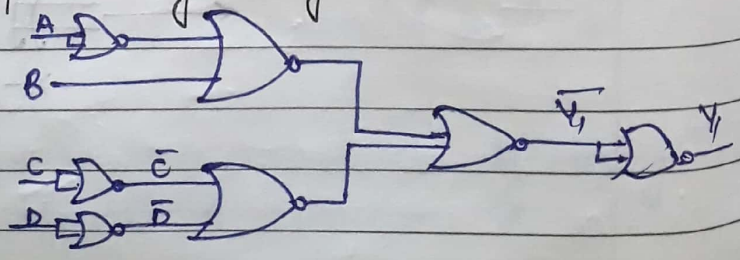
2. $Y_2 = \overline{(A+B).\bar{C}}$

3. $Y_3 = \bar{A}\bar{B} + AB$ (X-NOR)

Implement following func^ns by using NOR Gate.

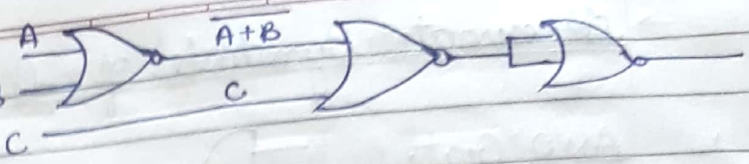
Ans- 1. $Y_1 = \overline{\overline{A\bar{B}} + \overline{C\bar{D}}}$

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



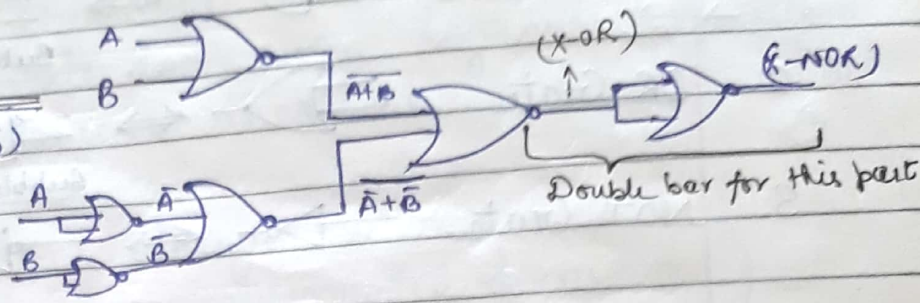
2. $Y_2 = \overline{(A+B).\bar{C}}$

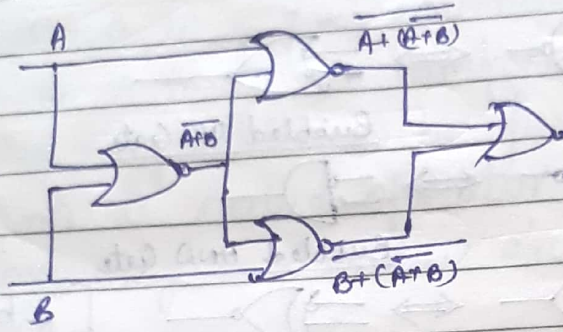
$= \overline{(\bar{A}+\bar{B})} + C$



3. $Y_3 = \overline{\bar{A}\bar{B} + \overline{AB}}$

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





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

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

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

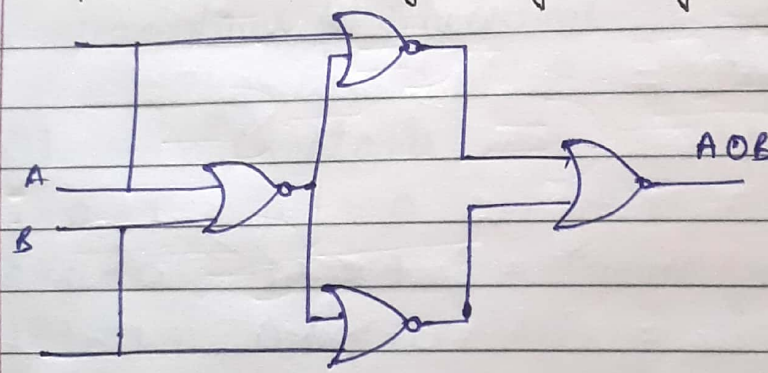
$(\bar{A}.\bar{B})(\bar{A}+\bar{B}) = (A+B)(\bar{A}+\bar{B})$

Q. Implement X-NOR gate by using min^m no. of NOR gates.

Ans- By duality,

NAND ↔ NOR

X-OR ↔ X-NOR.



Q. Implement $A+BC$ by using min^m no. of NOR gates.

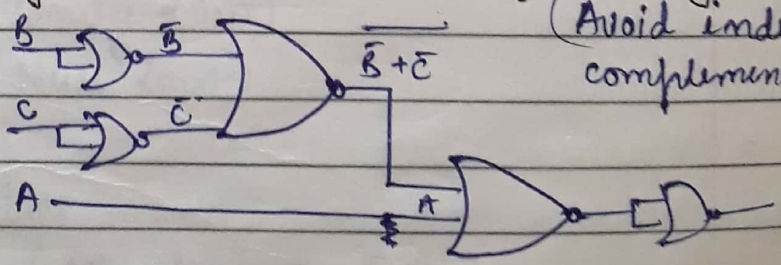
Ans- $Y = \overline{A+BC}$

$= \overline{\bar{A}.\overline{BC}}$

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

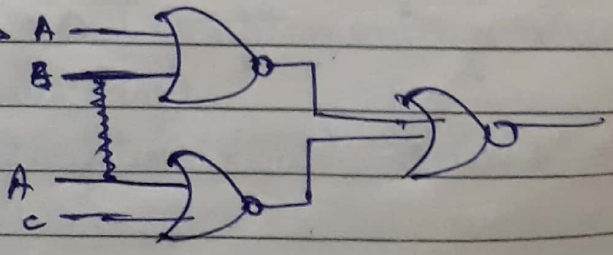
$= \overline{A+(\bar{B}+\bar{C})}$

(Avoid individual complement).

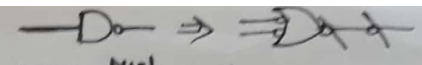


$= \overline{\bar{A}\bar{B} + \bar{A}\bar{C}}$

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



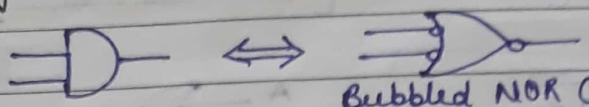
(Min^m)

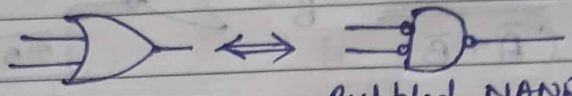


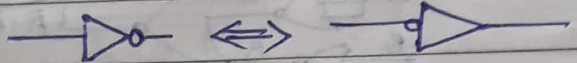
NAND $\xrightarrow{\text{dual}}$ NOR
+
Bubble in I/P & O/P.

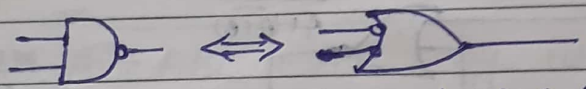
Page No: 12
18 | 7 | 18

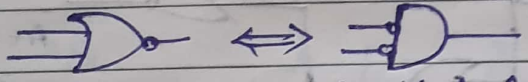
Alternate Symbols of Gates

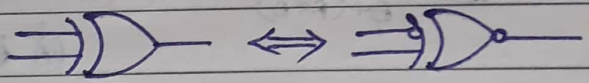
1. AND Gate  $\left[\overline{a \cdot b} = \bar{a} + \bar{b}\right]$
Bubbled NOR Gate

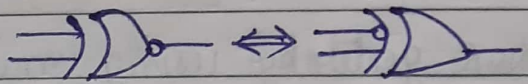
2. OR Gate 
Bubbled NAND Gate

3. NOT Gate 

4. NAND Gate 
Bubbled OR Gate

5. NOR Gate 
Bubbled AND Gate

6. X-OR Gate 
$\left[\begin{aligned} &\bar{a}b + a\bar{b} \\ &= \bar{\bar{a}}b + \bar{a}b \quad [a = \bar{\bar{a}}] \\ &= a\bar{b} + b\bar{a} \end{aligned}\right]$

7. X-NOR Gate 

Boolean Algebra

In boolean algebra, there are constants, variables and operators and we write down any logical expression in terms of (by using) some laws & theorems.

Constant - Value doesn't change : 0 or 1
Variables - a to z (in uncomplemented form : x)
or A to Z (in complemented form : $\bar{x}$)
Operator - Operation to be performed on variables & constants is called operator.
There are only 3 operators in boolean algebra: NOT, (−) AND, OR. (·) (+)

Boolean Laws & Theorems

(a) Operation on Constants

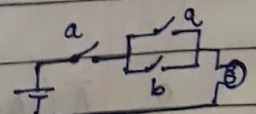
1. $1+0=1$, $0.1=0$
2. $0+0=0$, $1.1=1$
3. $1+1=1$, $0.0=0$
4. $\bar{1}=0$, $\bar{0}=1$

(b) Operation on Constant & Variables

1. $0+a=a$, $1.a=a$ (Unity Law)
2. $1+a=1$, $0.a=0$ (Null Law)

(c) Operation on Variables

1. $a+a=a$, $a.a=a$ (Idempotent Rule)
2. $a+a'=1$, $a.a'=0$ (Complementary Law)
3. $\bar{\bar{a}}=a$ (Involution Law)
4. $a+ab=a$ (Absorption Law), $a.(a+b)=a$



Total no. of expressions possible = 2

→ Commutative law: $a+b = b+a$, $a.b = b.a$
All gates follow commutative law.

→ Associative law: $(a+b)+c = a+(b+c)$, $(a.b).c = a.(b.c)$

Q. Determine whether a) NAND gate b) XOR gate follow associative law or not.

Ans- NAND: $\overline{(a.b).c}$ No ⇒ $\overline{a.b.c}$ $\overline{a.bc}$

XOR: $a \oplus (b \oplus c)$ $\overline{(\bar{a}+b).c}$ $\overline{c.(\bar{b}+\bar{c})}$

Yes. $\bar{a}c+\bar{b}c \neq a\bar{b}+a\bar{c}$

→ Distributive law: $a.(b+c) = a.b+a.c$, $(a+b).c = a.c+b.c$

→ DeMorgan's Theorem: $\overline{x+y+z} = \bar{x}.\bar{y}.\bar{z}$

Complement of sum = Product of complement of individual terms

or NOR gate = Bubbled AND gate

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

Product's complement = Sum of individual complements

or NAND gate = Bubbled OR gate

Q. Minimize the following expression by using Boolean laws & theorems:-

1. $Y = (A+C)(AD+A\bar{D})+AC+C$

Ans- $Y = (A+C).A.1 + C(A+1)$ [By comp. theorem, $\because x+\bar{x}=1$, $1+x=1$]

$= (A+C).(A+C)$

$= (A+C)$.

2. $\bar{A}(A+B)+(B+AA)(A+\bar{B})$

Ans- $Y = \bar{A}B+(B+A)(A+\bar{B}) = \bar{A}B+AB+B\bar{B}+A.A+A\bar{B}$ (B·B̄ → 0, A·A → A)

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

Consensus Theorem

$ab + bc + c\bar{a} = ab + c\bar{a}$

In this, there are 3 variables and each variable is available two times in any two product terms and one variable is in complemented form only once in a product term only, and answer is taken according to that complemented variable where it is available, either in complemented or in non-complemented form.

This theorem is also called Redundancy Theorem.

$\Rightarrow ab + bc + c\bar{a} = ab + c\bar{a}$ (SOP)

LHS $ab + bc + c\bar{a}$

$= ab + bc(a+\bar{a}) + c\bar{a}$

$= ab + abc + \bar{a}bc + c\bar{a}$

$= ab(1+c) + \bar{a}c(1+b)$

$= ab + \bar{a}c$ = RHS

$\Rightarrow (a+b).(b+c).(c+\bar{a}) = (a+b).(c+\bar{a})$ (POS)

LHS $(a+b).(b+c+0).(c+\bar{a})$

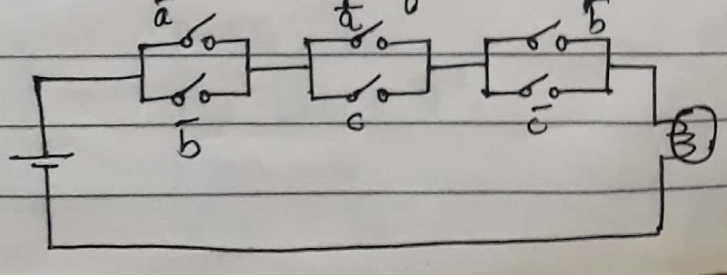
$= (a+b).(b+c+a.\bar{a}).(c+\bar{a})$

$= (a+b+c.\bar{c}).(b+c+a).(b+c+\bar{a}).(c+\bar{a}+0)$

$= (a+b+0.c).(c+\bar{a}+0.b)$

$= (a+b).(c+\bar{a})$ = RHS.

Q. What will be the simplified expression for the following circuit?



Ans- $(\bar{a}+b).(a+c).(\bar{b}+\bar{c})$
$= (\bar{a}+b).(\bar{a}+c)$ (By Consensus Theorem)

Q. In any circuit, o/p is low, when there are at least one I/P high, otherwise high. Implement that function by using NOR gate.

Ans-

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

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

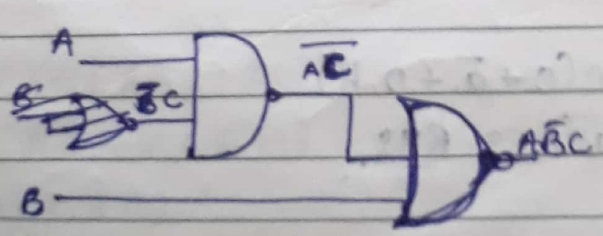
Q. If $A \oplus B = C$, what will be value of $B \oplus C$?

Ans- $B \oplus C = A$.

(Q) $a \oplus b = c$
& $a \oplus b \oplus c = 0$
If $b \oplus c = a$, then only eq^n satisfies, i.e,
$a \oplus a = 0$
$\therefore \boxed{b \oplus c = a}$

Q. Implement $Y = A\bar{B}C$ by using NAND and NOR gate. If cost of each gate is ₹10, what will be the cost of this circuit by implementing.

Ans- $Y = \overline{A\bar{B}C} = \overline{AC.\bar{B}} = \overline{\bar{A}\bar{C}+B}$



Cost = ₹20.

Number System

Base/Redix of a Number System:- Total no. of digits in a number system is called base/radix of that number system.
For eg, in Decimal no. system, base/redix = 10, because there are 10 digits from 0 to 9.
Similarly, in Hexadecimal no. system, base/redix = 16, because there are 16 digits from 0 to F.

* Minimum value of digit is 0, and max^{m} value of digit in any no. system is $(b-1)$ or $(r-1)$.

Positional Weight = r^{i} (redix)

$\cdots 2\ 1\ 0\ -1\ -2\ -3 \cdots -i$
$\Rightarrow xyz.abc$

Q. What is the positional weight of digit 8 & digit 2 in $(2783)_9$?

$3\ 2\ 1\ 0 \rightarrow i$

Ans- $(2\ 7\ 8\ 3)_9$
$9^3\ 9^2\ 9^1\ 9^0$

P.W of $8 = 9^1 = 9$
" " $2 = 9^3 = 729$

MSD (Most Significant Digit) - The digit which has highest positional weight.
LSD (Least Significant Digit) - The digit which has lowest positional weight.

Q. Determine MSD & LSD and positional weight of MSD & LSD in the following ques^{ns}:-

1. $(235.76)_9$ — MSD = 2, Pos. wt. $= 9^2$; LSD = 6, P.W $= 9^{-2}$
2. $(738A)_{16}$ — MSD = 7, P.W $= 16^3$; LSD = 10, P.W $= 16^0$

3. $(11011.01)_2$ MSD = 1, P.W = 2^4 LSD = 1, P.W = 2^{-2}

4. $(235.127)_8$ MSD = 2, P.W = 8^2 LSD = 7, P.W = 8^{-3}

Conversion from Base 10 to any Number System whose Base is r

Integer No = $\sum_{i=0}^{N-1} a_i r^i = a_0 + a_1 r^1 + a_2 r^2 + \dots$ —①

where, a_i = value of digit at position i
r = Radix / Base
r^i = Positional weight

Dividing eq^n ① by r,

q	rem.
$a_1 + a_2 r$	a_0
a_2	a_1
0	a_2

Steps → 1. Divide decimal no. by r in which number system we want to convert.
2. Separate quotient and remainder & repeat above process until quotient is 0.
3. Final answer will be taken of remainders in reverse order.

Qs Convert :-

(i) $(325)_{10} \rightarrow (\ )_8$
(ii) $(45)_{10} \rightarrow (\ )_{16}$
(iii) $(67)_{10} \rightarrow (\ )_2$

21	7	18

Ans-(i)

8 | 325		
8 | 40	5	↑
5	0	

$(325)_{10} = (505)_8$

(ii)

16 | 45		
2	13	↑

$(45)_{10} = (2D)_{16}$

45
32
13
10 11 12
D

(iii)

2 | 67		
2 | 33	1	
2 | 16	1	
2 | 8	0	
2 | 4	0	
2 | 2	0	
1	0	

$(67)_{10} = (1000011)_2$

(67) 64 32 16 8 4 2 1
1 0 0 0 0 1 1

Fractional Part →

Steps→ 1. Multiply fractional part by r and separate integer & fractional part.
2. Repeat above process until fractional part is 0, or any above step is repeated.
3. Final answer is taken in forward direction of integer.

Q1. $(0.625)_{10} \rightarrow (\ )_2$

	I	F
$0.625 \times 2 =$	1.	250
$0.25 \times 2 =$	0.	50
$0.50 \times 2 =$	↓1.	00

$(0.625)_{10} \rightarrow (0.101)_2$

2. Convert $(125.315)_{10} \rightarrow (\ )_8$

Ans-

8	125	
8	15	5
	1	7

$0.315 \times 8 = 2.520$
$0.520 \times 8 = 4.160$
$0.160 \times 8 = 1.280$
$0.28 \times 8 = 2.24$
$0.24 \times 8 = 1.92$

$(125.315)_{10} \rightarrow (175.24121)_8$

3. Convert $(79.125)_{10} \rightarrow (\ )_{16}$

Ans-

16	79	
	4	15

$0.125 \times 16 = 2.000$

$(79.125)_{10} \rightarrow (4F.20)_{16}$

Conversion from Base r to Base 10

$$\text{Decimal No} = \sum_{i=-M}^{N-1} a_i r^i$$

where, a_i = value of digit at position i
r^i = positional weight

Q Perform the following operations :-

1. $(11011.101)_2 \rightarrow (\ )_{10}$
2. $(AB.5)_{16} \rightarrow (\ )_{10}$
3. $(1212.121)_3 \rightarrow (\ )_{10}$
4. $(1241)_8 \rightarrow (\ )_{10}$

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

$= 1 + 2 + 8 + 16 + \frac{1}{2} + \frac{1}{8}$

$= 27 + \frac{4+1}{8} = 27 + \frac{5}{8} = (27.625)_{10}$

2. $11\times16^{0} + 10\times16^{1} + 5\times16^{-1}$

$= 11 + 160 + \frac{5}{16} = 171 + 0.3125 = (171.3125)_{10}$

3. $2\times3^{0} + 1\times3^{1} + 2\times3^{2} + 1\times3^{3} + 1\times3^{-1} + 2\times3^{-2} + 1\times3^{-3}$

$= 2 + 3 + 18 + 27 + \frac{1}{3} + \frac{2}{9} + \frac{1}{27}$

$= 50 + \frac{9+6+1}{27} = 50 + \frac{16}{27} = (50.592)_{10}$

4. $1\times8^{0} + 4\times8^{1} + 2\times8^{2} + 1\times8^{3}$

$= 1 + 32 + 128 + 512$

$= (673)_{10}$.

Conversion from Base x to Base y

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

* $( )_{16} \xrightarrow{8421 \text{ Code}} ( )_2$ (Make group of 4)
* $( )_8 \xrightarrow{421 \text{ Code}} ( )_2$ (Make group of 3)

Q. $(A5.7)_{16} \rightarrow ( )_2$

8421 8421 8421

Ans - $(1010\,0101.0111)_2$

Q Convert $(1011011.1001)_2 \rightarrow ( )_{16} \rightarrow ( )_8$

Ans - $0101\,1011.1001 = (5B.9)_{16}$

$001\,011\,011.100\,100 = (133.44)_8$

Q In which base it is possible?

(i) $\sqrt{41} = 5$

(ii) $\frac{320}{12} = 13.1$

(iii) $\frac{55}{5} = 11$

(iv) $\frac{312}{20} = 13.1$

Ans- (i) $\sqrt{41} = 5$

$\sqrt{1\times b^0 + 4\times b^1} = 5\times b^0$

$1 + 4b = 25$

$\boxed{b = 6}$

(ii) $3\times b^2 + 2\times b^1 + 0\times b^0 = 1\times b^1 + 3\times b^0 + 1\times b^{-1}$

$1\times b^1 + 2\times b^0$

$\frac{3b^2+2b}{b+2} = b+3+\frac{1}{b} = \frac{b^2+3b+1}{b}$

$(3b^2+2b)\,b = (b+2)(b^2+3b+1)$

$3b^3+2b^2 = b^3+3b^2+b+2b^2+6b+2$

$2b^3-3b^2 = 7b+2$

$b(2b^2-3b-7) = 2$

Can't determine answer.

(iii) $\frac{5\times b^1 + 5\times b^0}{5\times b^0} = 1\times b^1 + 1\times b^0$

$\frac{5b+5}{5} = b+1$

$b+1 = b+1$

$\boxed{b \geq 6}$

(iv) $\frac{312}{20} = 13.1$

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

$\frac{3b^2+b+2}{2b} = b+3+\frac{1}{b} = \frac{b^2+3b+1}{b}$

$3b^2+b+2 = 2b^2+6b+2$

$b^2-5b = 0$

$\boxed{b = 5}$

Q. How many 1's are possible in this :-

$5 + 9\times2^{4} + 7\times2^{12} + 3\times2^{16}$ (4 2 1 ; 8 4 2 1)

Ans- ~~1014 1001×2^4 + 111×2^12 + 0111×2^16~~

$5 + 9\times16 + 7*16^{3} + 3\times16^{4} = (37095)_{16}$

$(+0\times16^{2})$

2 3 0 2 2 = 9 1's.

r's Complement of N

r's Comp. of N = (r-1)'s Comp. + 1

r → 2, 8, 10, 16, ...

(r-1)'s Complement

(r-1)'s → 1, 7, 9, F ....

→ Subtract every digit from (r-1).
↳ radix

Eg- r=2, ∴ sub. from 1.

Q. Determine 1's & 2's Complement of 1011.

Ans- (r=2)

1's Comp. of 1011

1111
−1011
0100

2's Comp. of 1011

⇒ 1's C + 1

0100
+0001
0101

Q. Determine 9's & 10's Complement of 725.

Ans-

999
−725
274 → (9's C)

274
+ 1
275 → (10's C)

Q. Determine 8's C of 100.

Ans-

777
888
−100
677

677
+ 1
888
−8 −8
700

27	7	18

Arithmatical Operations in different bases

$(35)_{10} \rightarrow (b)$
$+(27)_{10}$
12
$-10\ (b)$
62

$(34)_5$
$+(03)_5$
7
-5
42

$(E2)_{16}$
$-(AB)_{16}$
$(37)_{16}$

Q. $(AB)_{16} + (37)_{16}$
18
16
$14\ 2 = (E2)_{16}$

Q. $(36.7)_8 + (15.5)_8$
12 12
-8 -8
$5\ 4$ 4 $= (54.4)_8$

Q. $(8+4)_5 - (232)_5 = (32)_5$

Q. $(23)_{10} \times (7)_{10}$
16 21
-10 -10
6 11
-10
1

Q. $(123)_5 \times (12)_5$
-5 -5
1
123 X
1 5 8 3 1
-5
0
2031

Q. $(35)_{16} \times (A)_{16}$
50
-16
34
-16
18
-16
2

33
-16
17
-16
1

2182

Scanned by CamScanner

(a-b) by using (r-1)'s (1's or 9's C)

Steps→ 1. Determine 1's or 9's C of b, & add it to a.

2. Case-I If a>b, in this case, carry is generated and add this carry to LSD of result obtained in STEP-1.

Case-II If a<b, in this case, carry is not generated. Again, take 1's or 9's C of result obtained from STEP-1, & put -ve sign.

Q Perform the following operations:-

(a) 1011 - 0011 using 1's C method.

(b) 0101 - 1101 using 1's C method.

(c) 325 - 125 using 9's C method.

Ans-(a) 1011 a

− 0011 → 1's C b

1100

+ 1011

10111

+ 1

1000

(b) 0101

− 1101 → 1's C

0010

+ 0101

0111 → 1's C → −1000

(c) 325

−125 → 9's C

999

−125

874

+325

1199

+ 1

200

Q Perform the following operations by using 9's C method.

(a) 4125 - 115

```
  9999
- 0115
  9884
+ 4125
 14009
+    1
  4010
```

(a-b) by using r's Complement Method (2's C or 10's C)

s→ 1. Determine r's C (2's or 10's C) of b, then add it to a
2. Case-I If a > b, in this case, carry is generated, & remove or discard this carry.
Case-II If a < b, in this case, again take 2's or 10's C of result obtained in STEP-1 with -ve sign.

Q Perform the following operations :-

(a) 1011 - 0011 by using 2's C
(b) 325 - 425 by using 10's C.

Sol-(a) 0011 $\xrightarrow{2's\,C}$ 1101

```
 1101
+1011
11000   = 1000.
```

(b)

```
 999
-425
 574
+  1
 575   (10's C)
```

```
 575
+325
 900
```

```
 999
-900
 099
+  1
-100 ✓
```

(3 marks)

Advantages of 2's C over 1's C :-

1. In case of 1's C method, we again add carry to LSB, while during 2's C method, we discard this carry. So, only 1 adder circuit is required. Hence, less hardware, so less cost & more speed.
So, computer uses 2's C method instead of 1's C method.

Signed Number

There are 3 methods to represent any signed number :-

1. Sign Magnitude Method - In this method, MSB represents sign bit. If sign bit is 0, it represents no. is +ve, & if MSB is 1, it represents no. is -ve, and remaining bits represent magnitude of that signed no.

Eg - +5 → 0 | 101
(MSB | Magnitude)

-5 → 1101

Q Represent -33 by using Sign Mag. Method in 8-bits.

Ans.

2	33	
2	16	1
2	8	0
2	4	0
2	2	0
	1	0

33 → 0 0 0 1 0 0 0 0 1
-33 → 1 0 0 1 0 0 0 0 1

32 16 8 4 2 1
1 0 0 0 0 1

2. 1's C Method - +ve no. representation is same as in above method, & -ve no. is determined by finding 1's C of that +ve no.

Eg - +5 → 0101
-5 → 1's C of 0101 → 1010

2's C or 10's C → r − r − N + 1

0101 → $2^4 - 2^0 - 0101 + 1$
= 16 − 0101
= 100000 − 00101

Q. Represent +13 & −13 by using 1's C method in 6 bit.

Ans- +13 → 001101
−13 → 110010

1 8 4 2 1
1 1 0 1

Range of Numbers for N-Bit

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

3. 2's C Method:- In this method, again, +ve no. is represented same as in above two methods. -ve no. is represented by determining 2's C of that +ve no.

−N ⟶ 2's Comp. of (+N).

Eg- +5 → 0101
−5 → 2's C of 0101 ⟶ 1011

Q. Represent +13 & −13 in 2's C form.

Ans- +13 → 01101
−13 → 10011

Range of Numbers

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

Q. For N=6 bit, what will be range of number by using 2's C method?

Ans- Range = -2^5 to $+(2^5-1)$
= −32 to +31

* In 2's C method, 0 has the same representation for +ve & −ve.

+0 → 000
−0 → 111
+1000

→ N + 2's C of N = 0

Arithmatical Operations by using 2's Complement Method

Q Perform the following operations :-

(a)
```
  +15
  +13
 01111
+01101
 11100
```
+ve, 28

(b)
```
  +15
  -13
 01111
+10011
 00010
```
MSB, 2

(c)
```
  -15
  -13
 10001
+10011
 00100
```
2's C → 11011 + 1 = 11100 = (-28)

+15 → 01111
-15 → 10001
+13 → 01101
-13 → 10011

(d)
```
  -15
  +13
 10001
+01101
 11110
```
2's C → 00001 + 1 = 00010 = (-2)

SOP (Sum of Product)

It is the ORing of AND terms.
Eg- $Y = AB + A\bar{B}C + ABC$
In this, in any product term, variable is missing in complemented or uncomplemented form.

Standard / Canonical SOP

In this SOP, all product terms contain all variables, either in complemented or uncomplemented form.

Conversion from SOP to C-SOP

If x variable is missing, multiply by $(x+\bar{x})$ in that product term. Then, use distributive & idempotent theorem.

* In this, $x = 1$ & $\bar{x} = 0$.

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

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

$= ABC + AB\bar{C} + A\bar{B}C + ABC$ $[\because x(y+z) = xy + xz]$

$= \underset{1\,1\,1}{ABC} + \underset{1\,1\,0}{AB\bar{C}} + \underset{1\,0\,1}{A\bar{B}C}$ $[\because x+x = x$ (Idempotent law)]

Minterm - It is the product term which contains all the variables.

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

$Y = \Sigma m(5, 6, 7)$

$= \Pi M(0, 1, 2, 3, 4)$

→ C-SOP is Sum of Minterms.

In case of minterms, O/P is high in truth table.

POS (Product of Sum)

It is ANDing of OR terms.

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

Conversion from POS to C-POS

If x variable is missing in any OR term, add $(x.\bar{x})$ in that OR term.

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

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

* minterm = $\overline{\text{MAXTERM}}$
* $\overline{\text{MAXTERM}}$ = $\overline{\text{minterm}}$

$Y = (\bar{A}+B+C)(\bar{A}+B+\bar{C})(A+B+C)(A+\bar{B}+C)(\bar{A}+\bar{B}+C)$
$= \pi M(0,4,5)$
$= \Sigma m(1,2,3,6,7)$

Maxterm - It is the sum term which contains all the variables.

* If there are 'N' minterms & there are 'n' variables, then total maxterms = $2^n - N$.

→ C-POS is Product of Maxterms.

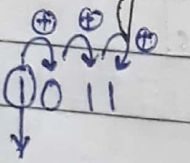
Ex: 1. $m_{13} = ab\bar{c}d$
2. $M_{21} = \bar{a}+b+\bar{c}+d+\bar{e}$
3. $M_{13} = \bar{a}+\bar{b}+c+\bar{d}$

$Y = a\bar{b}c\bar{d}$
$\bar{A}+B+\bar{C}+D$

⇒ m_0, m_1, m_2 } EX-OR, m_3

$Y = \Sigma m(1,2)$
$= m_1 + m_2$
$= \bar{m_1} \cdot \bar{m_2}$
$= M_1 \cdot M_2$
$= M_0 \cdot M_3$
$= \pi M(0,3)$

Binary to Gray Code Conversion



1110 ⇒ Gray Code

When no. of variables increases from 3, by using Boolean Algebra, it becomes complex

10	8	18

K-Map

When no. of variables increases from 3, by using Boolean algebra, it becomes complex.
To reduce complexity, we use K-Map and it is the graphical representation which is used to minimise any boolean function. It uses Gray Code or Unit Distance Code.

Adjacent Cells - From going one cell to next cell, if only one variable changes, these two cells are called adjacent cells.

Total cells $= 2^n$; n = no. of variables.
We can do grouping of only adjacent cells in the power of 2.

For eg - 2^i

$i = 1 \rightarrow$ Pair $\Rightarrow$ Eliminates 1 variable.
$i = 2 \rightarrow$ Quad $\Rightarrow$ Eliminates 2 variables
$i = 3 \rightarrow$ Octet $\Rightarrow$ Eliminates 3 variables.

i.e., In $2^i \Rightarrow i$ var. eliminates.

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

(or)

BC \ A	0	1
00	0	4
01	1	5
11	3	7
10	2	6

Prime Implicant (PI)

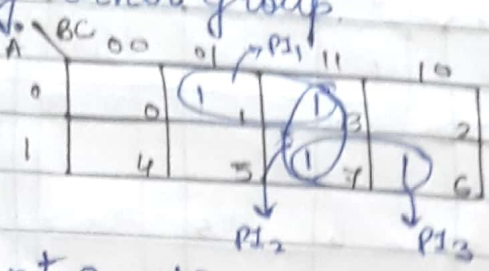
It is the minterm or group of minterms.

Essential Prime Implicant (EPI)

It is the PI in which there is atleast 1 minterm which

is not involved in any other group.

$Y = \Sigma m(1,3,6,7)$



$EPI_1 \rightarrow PI_1$

PI_2 is not EPI (redundant group)

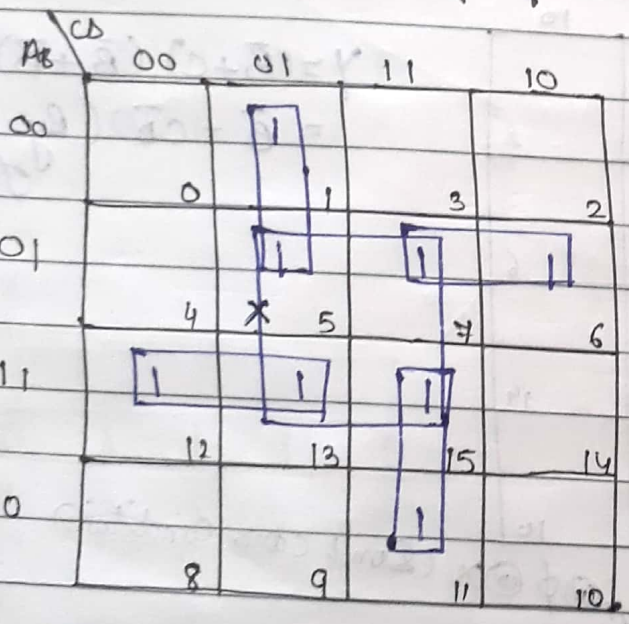
$EPI_2 \rightarrow PI_3$

$\therefore Y = EPI_1 + EPI_2$

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

So, it is dependent on 2 variables & independent of 1 variable.

Minimise the following func^{n} by using K-Map:-

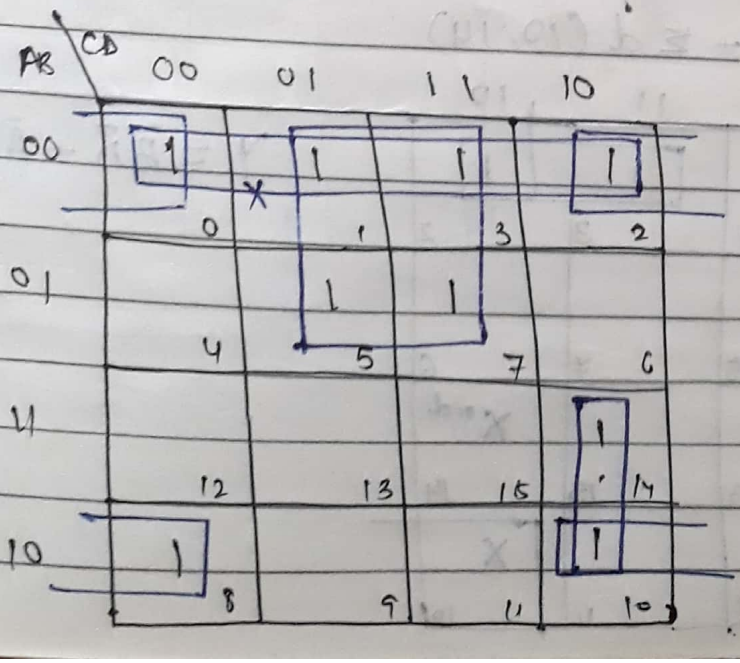


$Y = \Sigma m(1,5,6,7,11,12,13,15)$

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

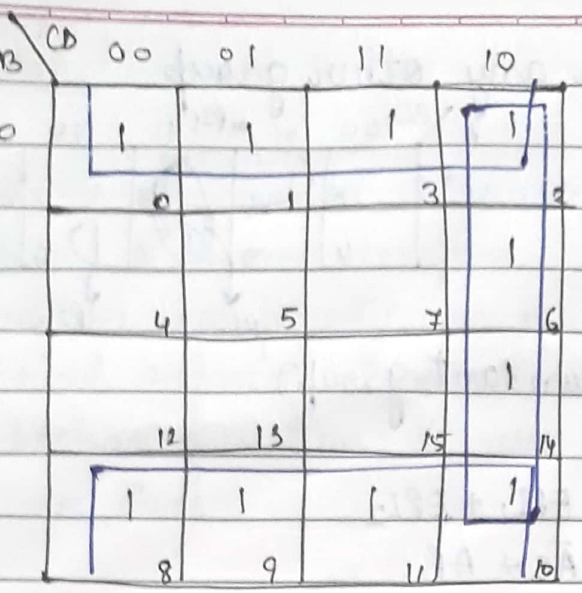
~~$= \bar{A}BC$~~

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



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

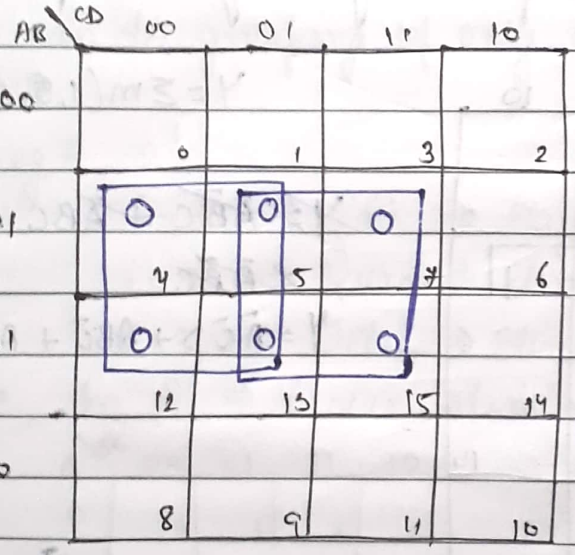
3.



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

Q. Minimise the following fⁿˢ by using K-Map :-

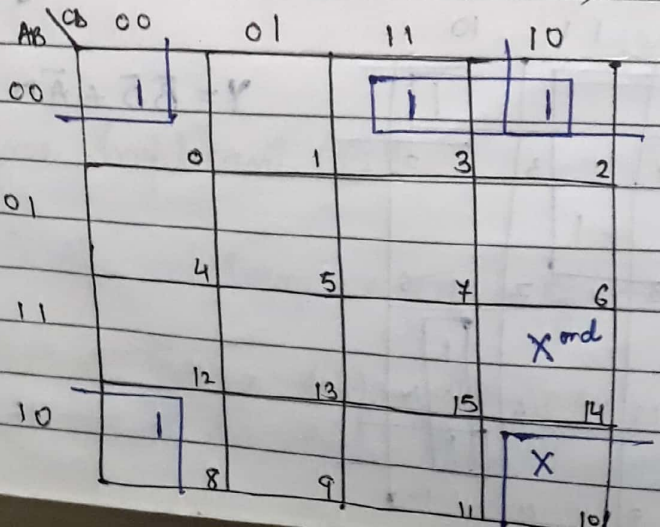
1. $Y = \Pi M (4, 5, 7, 12, 13, 15)$



$Y = (\bar{B}+C).(\bar{B}+\bar{D})$
$= \bar{B} + C\bar{D}$ (By Distributive theorem)

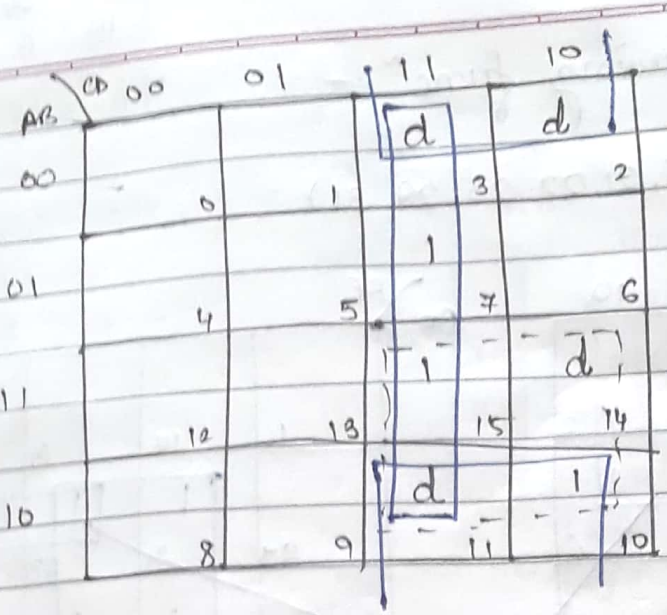
(or) ϕ (or) X (Don't care condition)

2. $Y = \Sigma m (0, 2, 3, 8) + \Sigma d (10, 14)$



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

Q. 3.

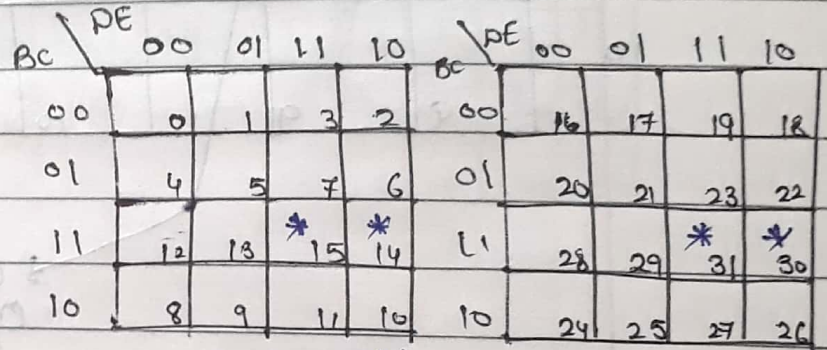


$Y = CD + \bar{B}C$ or $Y = AC + CD$ } Write both

5 Variable K-Map.

Total cells $= 2^5 = 32$

$A \rightarrow 0\ (\bar{A})$ $\quad$ $A \rightarrow 1\ (A)$

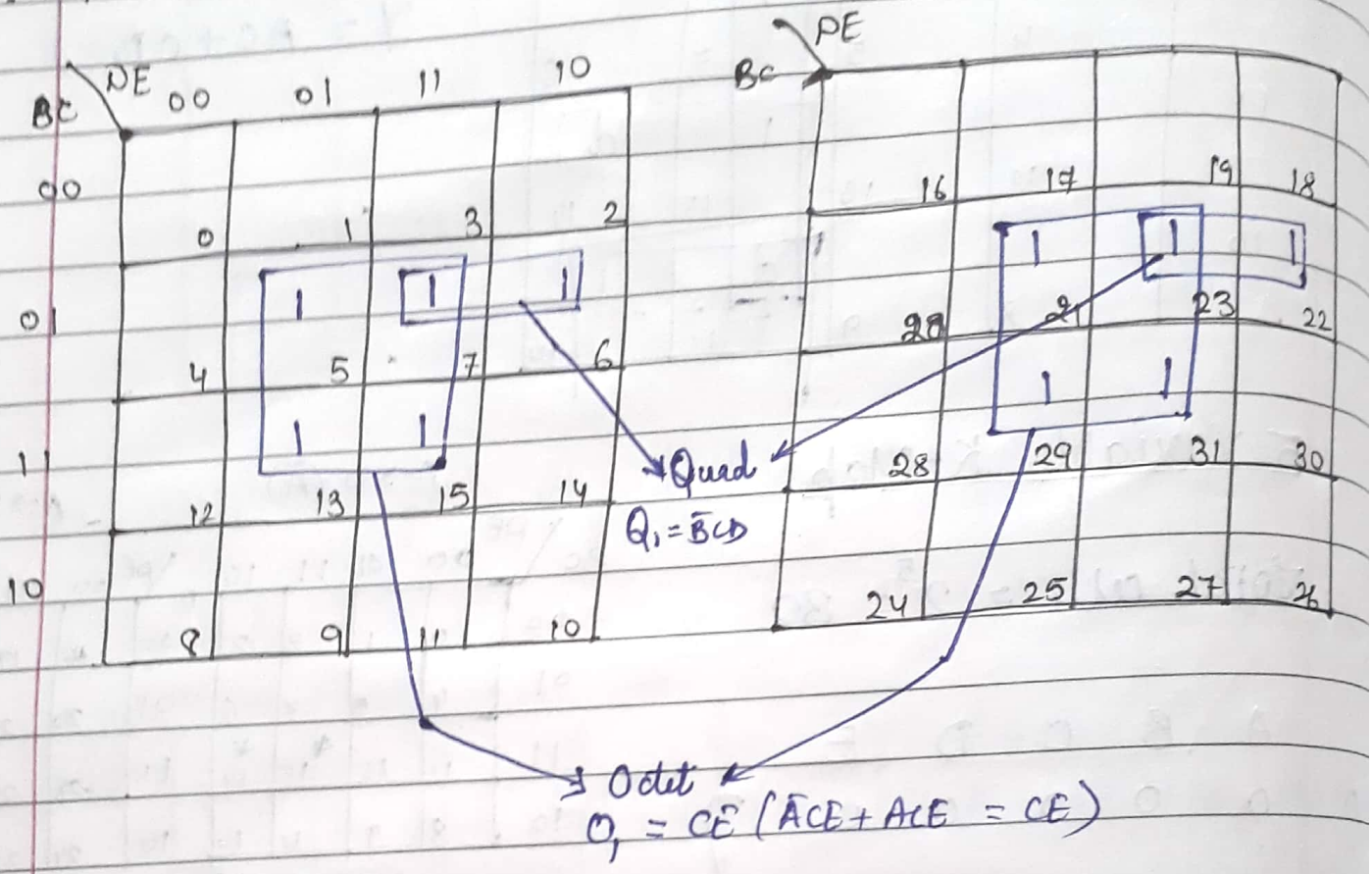


Sq^1 $\xrightarrow{adj'}$ Sq^2

	A	B	C	D	E	
0	0	0	0	0	0	Sq_1 ($\bar{A}$)
15	0	1	1	1	1	
16	1	0	0	0	0	Sq_2 (A)
31	1	1	1	1	1	

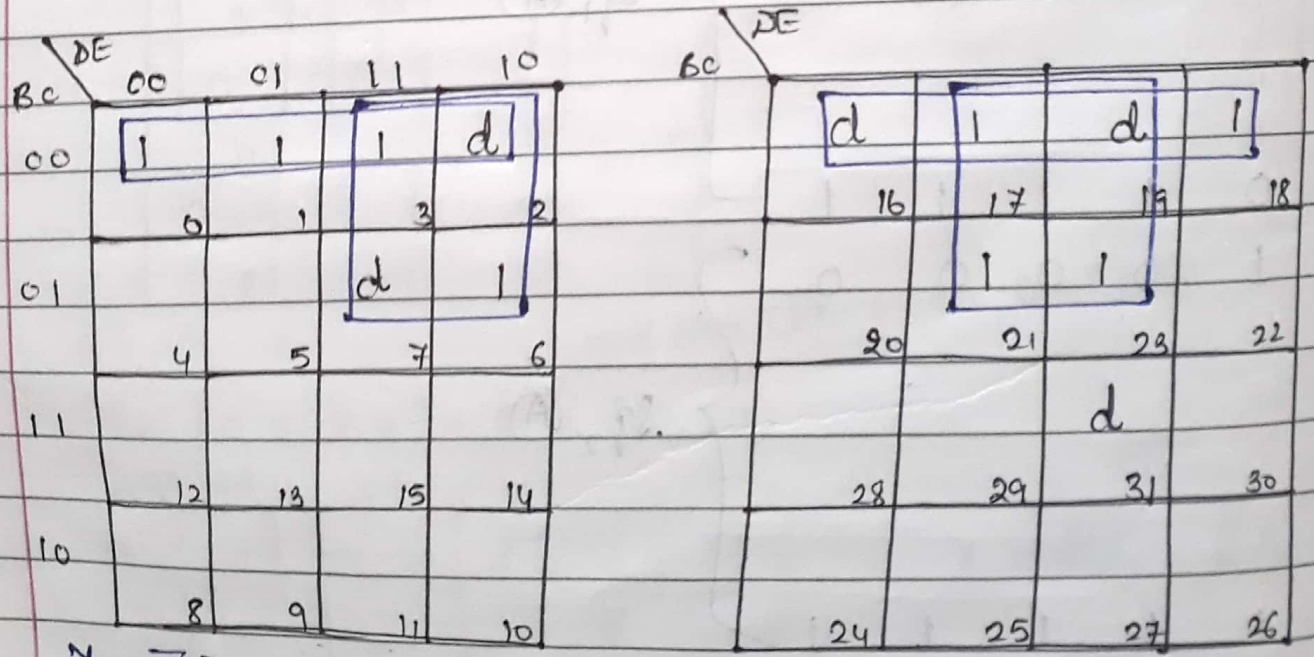
Q. Minimise the following func^{n} :-

1. $Y = \Sigma m (5, 6, 7, 13, 15, 21, 22, 23, 29, 31)$



$Y = CE + \bar{B}CD$.

2.

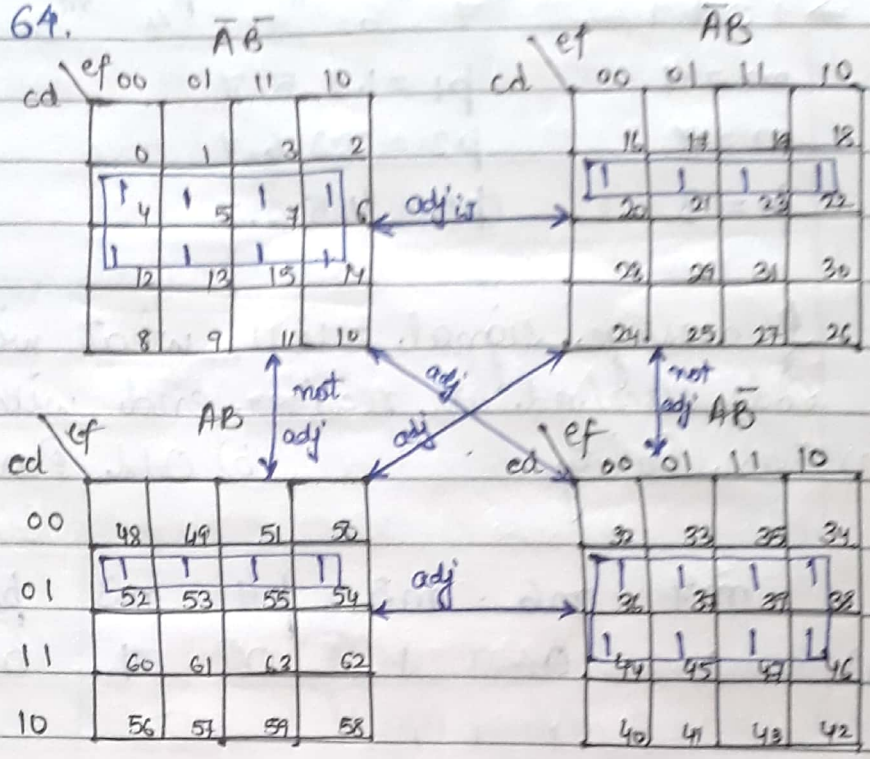


$Y = \bar{B}\bar{C} + \bar{A}\bar{B}D + A\bar{B}E$

6 Variables K-Map

Total cells = $2^6 = 64$.

a b c d e f



$Y = \bar{b}d + \bar{c}d$

Hamming Code

If total no. of bits at message signals = m, & total bits received at receiver = n, total parity bits = n − m.

$$\begin{cases} n = m + p \\ p = n - m \end{cases}$$

Parity bits are added with message signal (m) & transmitted to receiver & at receiver, we check even parity or odd parity in total 'n' bits

Even Parity – If total no. of 1's is even, it's called even parity.

Odd Parity – If total no. of 1's is odd, it's called odd parity.

Codes

A code is a binary representation of decimal numbers.

Types of Codes → 1. Weighted Codes
2. Non-Weighted Codes
3. Error detecting Codes
4. Error detecting & correcting codes

Properties: – Self-complimenting Codes
– Reflective codes or Cyclic Codes.

	BCD (Binary Coded Decimals) 8421	2421	6 4 2 -3	Excess 3 (Add 3 in every BCD digit)	8421 Even Parity
0	0000	0000	0000	0011	0000 0
1	0001	0001	0101	0100	0001 1
2	0010	0010	0010	0101	0010 1
3	0011	0011	1001	0110	0011 0
4	0100	0100	0100	0111	0100 1
5	0101	1011	1011	1000	0101 0
6	0110	1100	0110	1001	0110 0
7	0111	1101	1101	1010	0111 1
8	1000	1110	1010	1011	1000 1
9	1001	1111	1111	1100	1001 0

Self-Complimenting Codes (Bit Sum = 9) — Weighted Codes — Error-detecting Code

Parity → 1. Even – Adding 0 or 1 to make even no. of 1's
2. Odd

If even parity = 0, then there is no error, & if it is = 1, then there is error.
It can detect only single-bit errors.

Hamming Code

Error detecting & Correcting Code

7-bit code :-

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

2 1's out of 5 bits.
(Can't detect more than 1-bit error)
⇒ Error Detecting ~~& Correcting~~ Code

Bit	1	2	3	4	5	6	7
Positions →	P_1 (1,3,5,7)	P_2 (2,3,6,7)	m_1	P_3 (4,5,6,7)	m_2	m_3	m_4

Message Bit = 4

1	0	1	0	1	0	1

No Error : 1 0 1 0 1 0 1

	C_1 (4,5,6,7)	C_2 (2,3,6,7)	C_3 (1,3,5,7)
	0	0	0 ← No Error.

0 0 1 0 1 0 1

C_1	C_2	C_3	
0	0	(1)	→ Error.

↳ Tells position of error, here, it's in bit position 1.

Corrected Code → 1 0 1 0 1 0 1

Weighted Code – Follow positional weight.
Eg – 8421 Code, BCD Code.

Non-weighted Code – Not follow positional weight.
Eg – Gray Code, XS-3 Code.

BCD → 0 to 9
XS-3 → 3 to 12

7-Bit Hamming Code

$m_7\ m_6\ m_5\ p_4\ m_3\ p_2\ p_1$

⇒ 1011

$m = 4$

$n = 7$

$p = 3$

$p_1 = 1, 3, 5, 7$

$p_2 = 2, 3, 6, 7$

$p_4 = 4, 5, 6, 7$

4 2 1

001 → p_1

010 → p_2

100 → p_4

Q. If message signal = 1011, what will be 7-bit Hamming Code received at receiver end with :-

a) Even Parity

b) Odd Parity.

Ans-

a)

m_7	m_6	m_5	p_4	m_3	p_2	p_1
1	0	1	0	1	0	1

$p_1 \rightarrow 1$ 3, 5, 7 (1, 0, 1)

$p_2 \rightarrow 0$ 3, 6, 7 (1, 0, 1)

$p_4 \rightarrow 0$ 5, 6, 7

b)

m_7	m_6	m_5	p_4	m_3	p_2	p_1
1	0	1	1	1	1	0

$p_1 \rightarrow 0$

$p_2 \rightarrow 1$

$p_4 \rightarrow 1$

Q. Determine 7-bit Hamming Code for 1000 for odd parity.

Ans-

m_7	m_6	m_5	p_4	m_3	p_2	p_1
1	0	0	0	0	0	0

$p_1 \rightarrow 0$ 3, 5, 7 (0, 0, 1)

$p_2 \rightarrow 0$ 3, 6, 7 (0, 0, 1)

$p_4 \rightarrow 0$ 5, 6, 7

Q. A 7-bit Hamming Code is received as 1010111 at receiver end. Determine whether this code is correct or not. If not, correct it for even parity.

Ans- 1 0 1 0 1 1 1
m7 m6 m5 p4 m3 p2 p1

p1 → 1, 3, 5, 7 ⇒ 1, 1, 1, 1 (even) → 0 (No error)
p2 → 2, 3, 6, 7 ⇒ 1, 1, 0, 1 (odd) → 1 (Error)
p4 → 4, 5, 6, 7 ⇒ 0, 1, 0, 1 (even) → 0 (No error)

(010) → 2, i.e, error is in second bit (p2).

So, corrected Code = 1010101

Q. 1100110 (Odd Parity)

Ans- 1 1 0 0 1 1 0
m7 m6 m5 p4 m3 p2 p1

p1 → 1,3,5,7 ⇒ 0, 1, 0, 1 ⇒ (even) → 1 (Error)
p2 → 2,3,6,7 ⇒ 1, 1, 1, 1 ⇒ (Even) → 1 (Error)
p4 → 4,5,6,7 ⇒ 0, 0, 1, 1 ⇒ (Even) → 1 (Error)

(111) → 7

Error is in 7th bit.
So, corrected Code = 0100110

15-Bit Hamming Code.

Message bits = 11
Parity bits = 4

m_{15} m_{14} m_{13} m_{12} m_{11} m_{10} m_9 p8 m7 m6 m5 p4 m3 p2 p1

p1 = 1, 3, 5, 7, 9, 11, 13, 15 — 0001 → p1
p2 = 2, 3, 6, 7, 10, 11, 14, 15 — 0010 → p2
p4 = 4, 5, 6, 7, 12, 13, 14, 15 — 0100 → p4
p8 = 8, 9, 10, 11, 12, 13, 14, 15 — 1000 → p8

Q. 1101011001011101

Ans- Even: p1 → Even → 0
p2 → Odd → 1
p4 → odd → 1
p8 → Odd → 1

(1110) → 14

Corrected Code = 1001011001011101

Tabular Method or Q-Method or Quine-Mcklusky Method

Q. Minimise the following func^n by using Tabular Method:
$Y = \sum m(1, 3, 5, 7, 9, 11, 13, 15)$

Ans-

Group	Minterm	Variables abcd
G1	1	0001 ✓
G2	3	0011 ✓
	5	0101 ✓
	9	1001 ✓
G3	7	0111 ✓
	11	1011 ✓
	13	1101 ✓
G4	15	1111 ✓

Group	Minterm	abcd
G1G2	(1,3)	00-1 ✓
	(1,5)	0-01 ✓
	(1,9)	-001 ✓
G2G3	(3,7)	0-11 ✓
	(3,11)	-011 ✓
	(5,7)	01-1 ✓
	(5,13)	-101 ✓
	(9,11)	10-1 ✓
	(9,13)	1-01 ✓

G_3G_4	(7,15)	-111 ✓
	(11,15)	$1-11$ ✓
	(13,15)	$11-1$ ✓

Group	Minterm	a b c d
G_1G_2 G_2G_3	(1,3,5,7)	$0--1$ ✓
	(1,3,9,11)	$-0-1$ ✓
	(1,5,3,7)	$0--1$ –
	(1,5,9,13)	$--01$ ✓
	(1,9,3,11)	$-0-1$ –
	(1,9,5,13)	$--01$ –
G_2G_3 G_3G_4	(3,7,11,15)	$--11$ ✓
	(3,11,7,15)	$--11$ –
	(5,7,13,15)	$-1-1$ –
	(5,13,7,15)	$-1-1$ ✓
	(9,11,13,15)	$1--1$ ✓
	(9,13,11,15)	$1--1$ –

⇒ "–" are the repeated ones.

PI Chart :-

Group	1 ✓	3 ✓	5 ✓	7 ✓	9 ✓	11 ✓	13 ✓	15 ✓
1,3,5,7,9,11,13,15 ✓ (d)	X	X	X	X	X	X	X	X

a b c d
– – – 1

$\boxed{Y = d}$

Minimise the following f^n by using Tabular method :-

$Y = \Sigma m\,(1, 3, 9, 11, 12, 13, 14, 15)$

Group	Minterm	a b c d
G_1	1	0 0 0 1 ✓
G_2	3	0 0 1 1 ✓
	9	1 0 0 1 ✓
	12	1 1 0 0 ✓
G_3	11	1 0 1 1 ✓
	13	1 1 0 1 ✓
	14	1 1 1 0 ✓
G_4	15	1 1 1 1 ✓

Group	Minterm	a b c d
G_1G_2	(1,3)	0 0 - 1 ✓
	(1,9)	- 0 0 1 ✓
G_2G_3	(3,11)	- 0 1 1 ✓
	(9,11)	1 0 - 1 ✓
	(9,13)	1 - 0 1 ✓
	(12,13)	1 1 0 - ✓
	(12,14)	1 1 - 0 ✓
G_3G_4	(11,15)	1 - 1 1 ✓
	(13,15)	1 1 - 1 ✓
	(14,15)	1 1 1 - ✓

Group	Minterm	a b c d
$G_1G_2G_2G_3$	(1,3,9,11)	- 0 - 1
$G_2G_3G_3G_4$	(9,11,13,15)	1 - - 1
	(12,13,14,15)	1 1 - -

18	8	18

PI Chart :-

Group	1 ✓	3 ✓	9 ✓	11 ✓	12 ✓	13 ✓	14 ✓	15 ✓
(1,3,9,11) $\bar{b}d$ ✓	×	×	×	×				
(9,11,13,15) ad ✓			×	×		×		×
(12,13,14,15) ab ✓					×	×	×	×

$$Y = \bar{b}d + ab$$

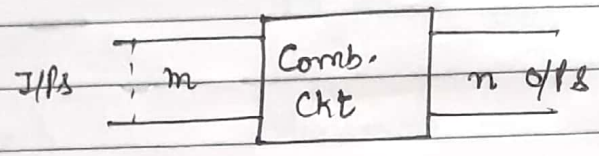
UNIT-2

Combinational Circuits

Combinational Circuit

In this, O/P depends upon present I/Ps only, not on past value.

Design & Analysis Process →



There are 'm' I/Ps & 'n' O/Ps.

STEP-1 Determine no. of I/Ps & no. of O/Ps.

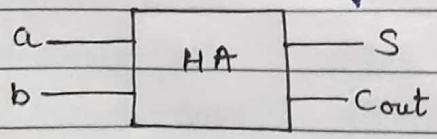
STEP-2 Draw truth table of given combinational ckt.

STEP-3 Now, with the help of either K-Map or Boolean algebra, determine the reduced expression of every o/p w.r.t I/Ps

STEP-4 Design combinational circuit by using logic gates.

Half Adder

Half Adder simply adds 2 bits.



HA → a+b

Truth Table →

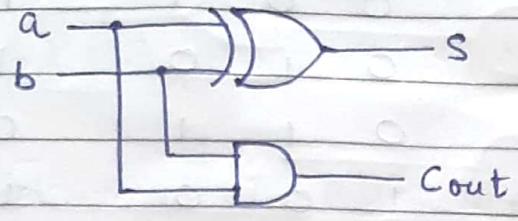
I/Ps		O/Ps	
a	b	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = \Sigma m(1,2)$ (X-OR Gate)

$= a \oplus b$

$Cout = \Sigma m(3)$ (AND Gate)

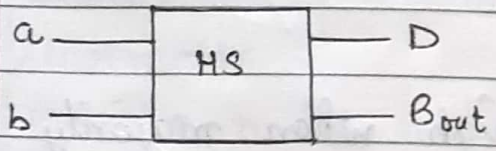
$= a.b$



Half Adder Circuit

In half adder, there is no provision to add Carry.

Half Subtractor



$HS \rightarrow a - b$

Truth Table →

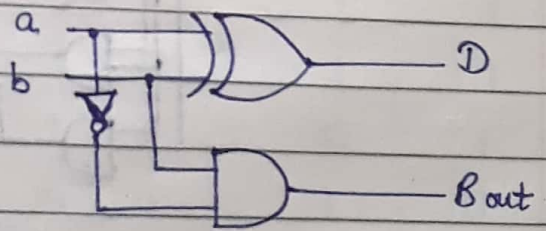
I/Ps		O/Ps	
a	b	D	Bout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

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

$= a \oplus b$

$Bout = \Sigma m(1)$

$= \bar{a}.b$



Half Subtractor Circuit

Full Adder

In half adder, there is no provision to add carry bit & this limitation is avoided by full adder.

$F.A \rightarrow a + b + Cin$

Truth Table →

I/Ps			O/Ps	
a	b	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

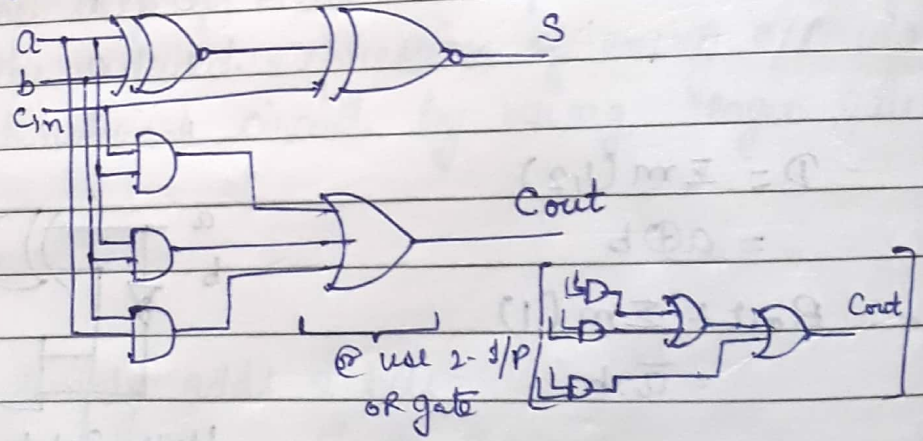
$S = \Sigma m(1,2,4,7)$ (∵ Odd no. of 1's at I/P, ∴ X-OR Gate)

$= a \oplus b \oplus C_{in}$

$C_{out} = \Sigma m(3,5,6,7)$

$= ab + bc_{in} + c_{in}a$

* Cout in full adder becomes high when mayority no. of 1's is high.

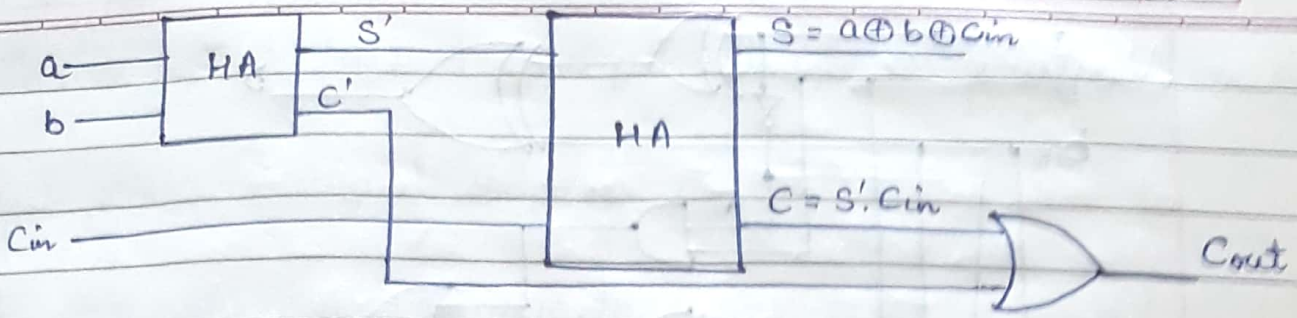


Full Adder Circuit

Q. Implement full adder by using half adder & suitable logic gate.

Ans - $S' = a \oplus b$ $\quad S = S' \oplus C_{in}$

$C' = a \cdot b$ $\quad C = (a \oplus b) \cdot C_{in}$



$C' = (a \oplus b) \cdot C_{in}$

$C' = a \cdot b$

$C_{out} = \bar{a}bC_{in} + a\bar{b}C_{in} + ab\bar{C}_{in} + abC_{in}$

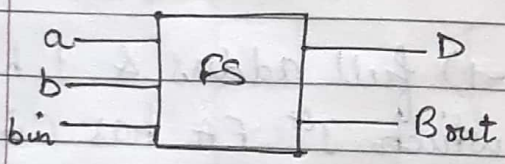
$= C_{in}(\bar{a}b + a\bar{b}) + ab(C_{in} + \bar{C}_{in})$

$= C_{in}(a \oplus b) + ab$

Full Subtractor.

There's no provision to subtract borrow bit in half subtractor.

FS → a - b - bin = a - (b + bin)



Truth Table →

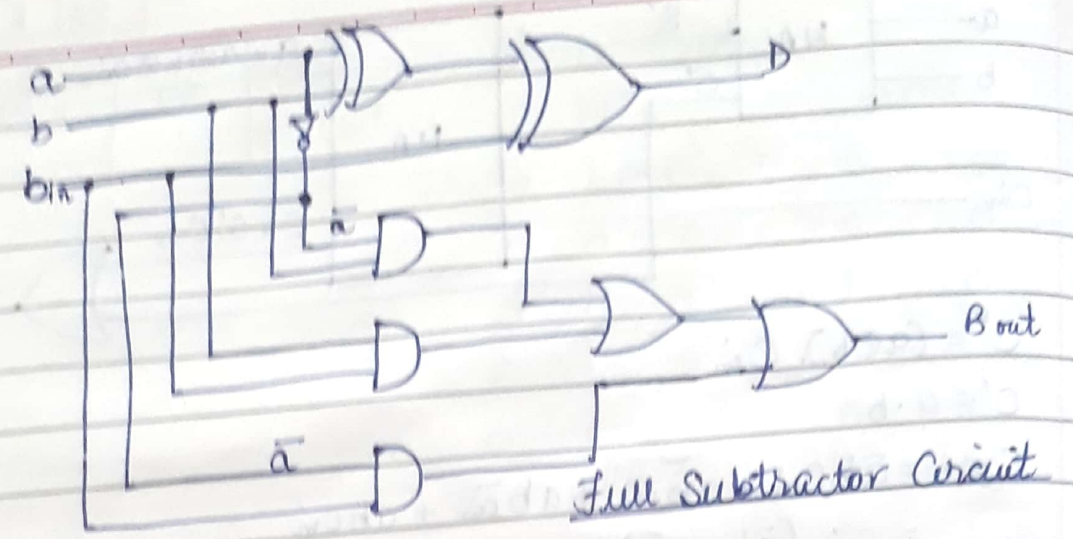
I/Ps			O/Ps	
a	b	bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

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

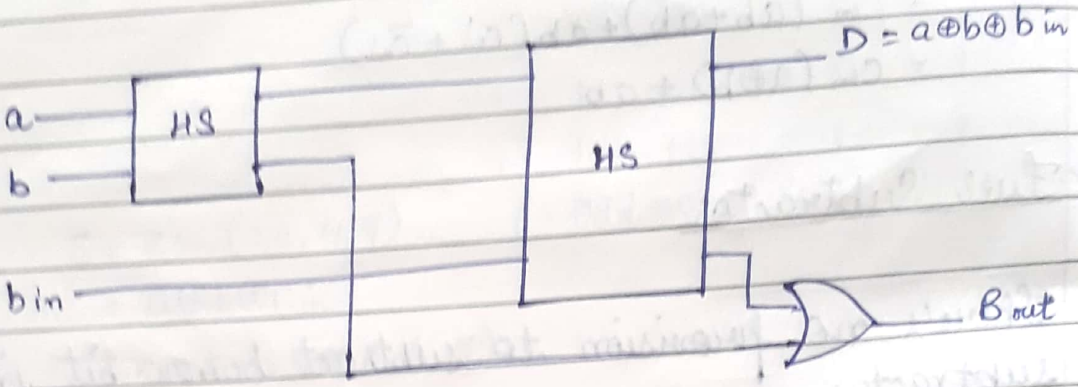
$= a \oplus b \oplus bin$

$B_{out} = \Sigma m(1,2,3,7)$

$= \bar{a}b + b\,bin + bin\,\bar{a}$

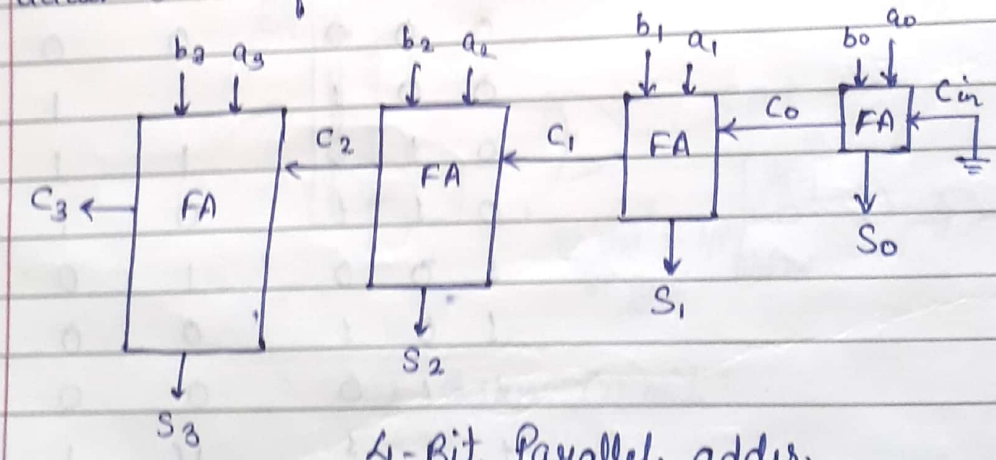


Full Subtractor Circuit



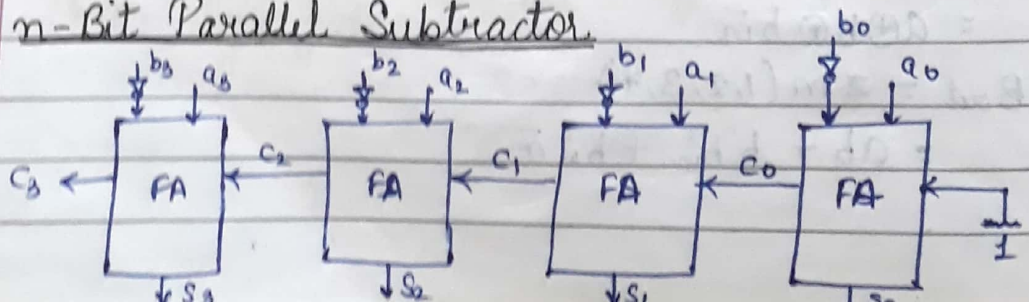
n-Bit Parallel Adder

To add n-bits, we require (n-1) full adders & 1 half adder or n full adders in which 1st F.A has $C_{in} = 0$.



4-Bit Parallel adder

n-Bit Parallel Subtractor



4-Bit 1's Subtractor

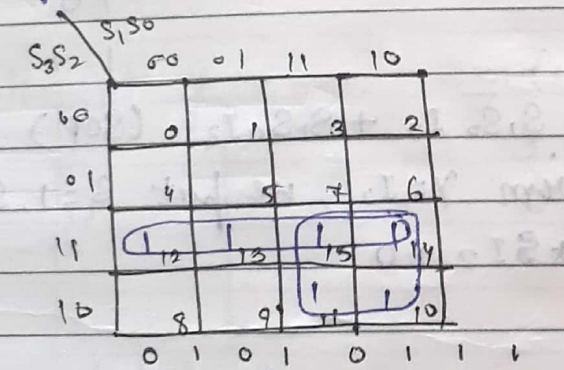
This is not a fast technique. It causes propagation delay.

BCD Adder / Decimal Adder

In BCD, there are 10 digits from 0–9

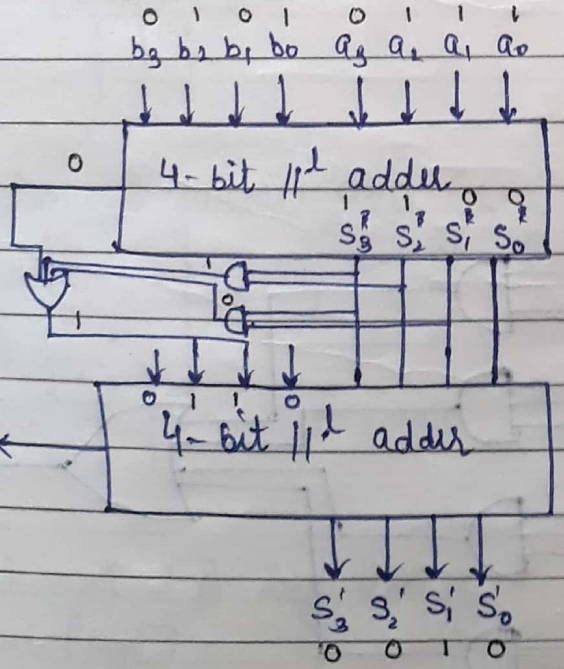
```
  5        0101
 +6      + 0110
 11       1011 → B     If res>9, add 6, otherwise 0
        + 0110
        00010001
          1   1
```

After adding two BCD digits, if sum > 9, we add 6 to its sum & it is possible in 10, 11, 12, 13, 14, 15 (after adding 6), and when result is < 9, we add 0.



$Y = S_3S_2 + S_3S_1$

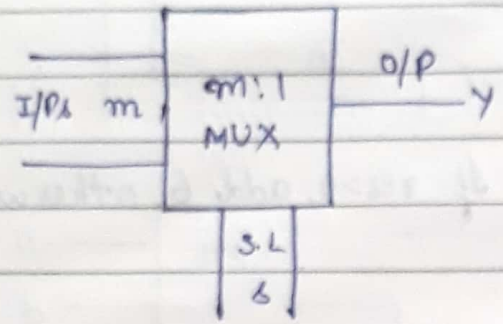
BCD → 8421
4 bit req.




```
  1100        7
 +0110       +5
 1 0010      12
   Co
```

Multiplexer

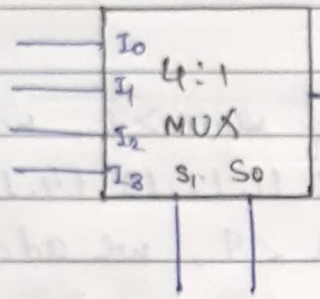
It has many inputs and only 1 O/P & depending upon the value of selected lines, it connects 1 I/P with O/P. That's why, it's also called Data Selector.



$m = 2^s$ ← SL (Selected lines)

I/Ps

Eg-



$4 = 2^2 = 2^s$

$s = 2$

Truth Table →

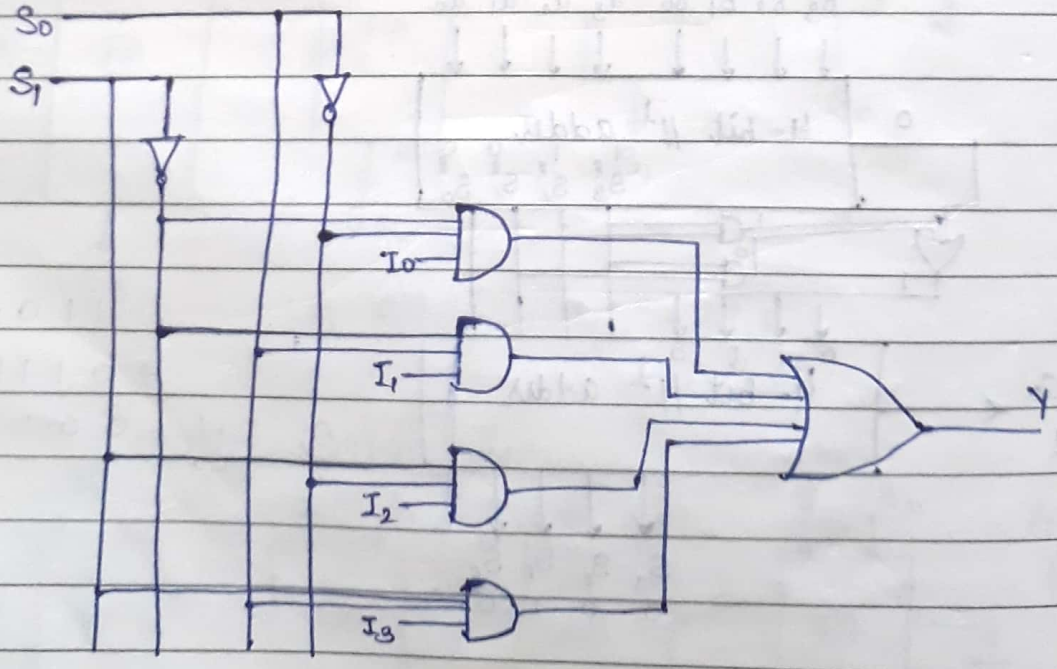
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$Y = \bar{S_1}\bar{S_0}I_0 + \bar{S_1}S_0I_1 + S_1\bar{S_0}I_2 + S_1S_0I_3$ (SOP)

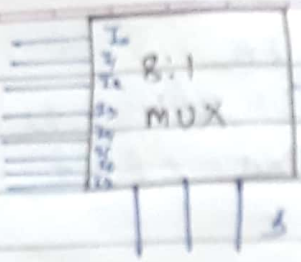
Now, in order to confirm $Y = I_2$, we put $S_1 = 1, S_0 = 0$

$\therefore Y = 0 + 0 + 1 * 1 * I_2 + 0$

$= I_2$



Sol-



$8 = 2^n$; 2^4

$n = 3$

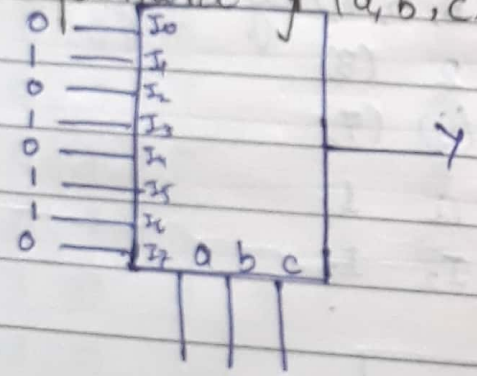
Truth Table →

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

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

Q Implement $F(a,b,c) = \Sigma m(1,3,5,6)$ by using 8:1 MUX.

Ans-

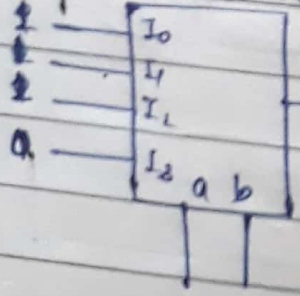


1 → +5V or $+V_{cc}$

0 → GND ($\perp$).

Q Implement NAND gate (2-I/P) by using 4:1 MOX.

Ans-



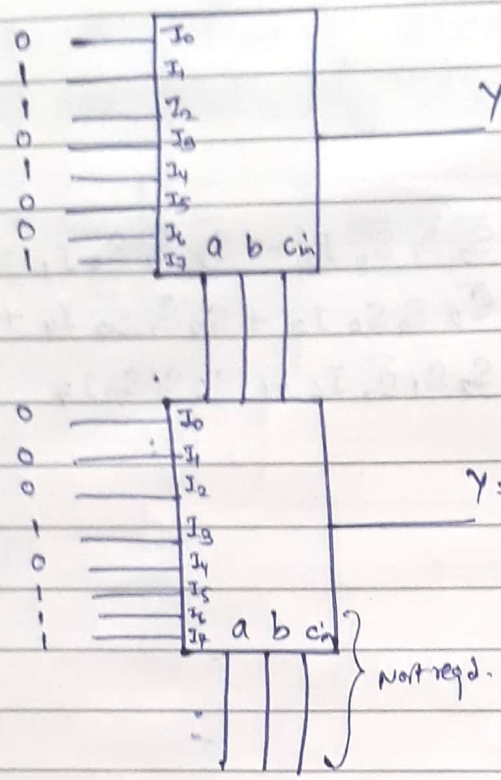
$Y = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$

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

Q Implement FA by using 8:1 MUX.

Ans- (Next Page)

In FA,
$S = \Sigma(1,2,4,7)$
& $C_{out} = \Sigma(3,5,6,7)$

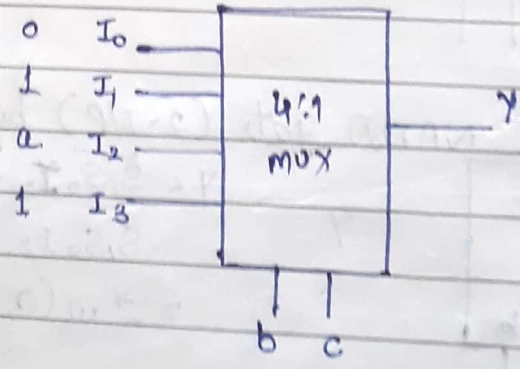


Qs. Implement $Y(a,b,c) = \Sigma m(1,3,5,6,7)$ by using 4:1 MUX.

Ans-

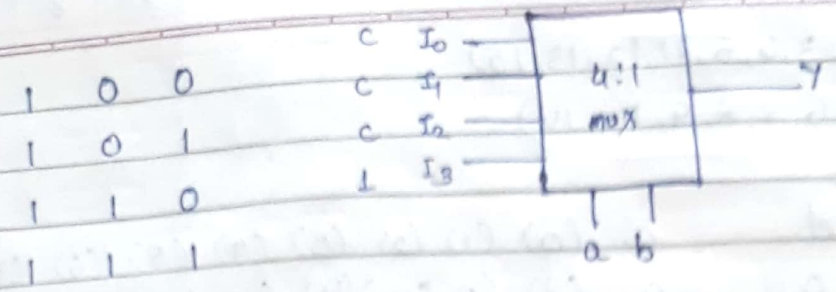
a	b	c
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

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



a	b	c
0	0	0
0	0	1
0	1	0
0	1	1

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



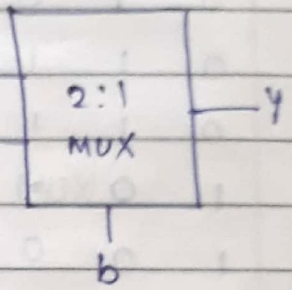
Q 1. Implement 2-I/P NOR gate by using 2:1 MUX.
2. Implement full adder by using 4:1 MUX.
3. Implement $Y = \prod^{(a,b,c,d)} M(1,2,3,4,5,11,12,13,15)$ by using 8:1 MUX

Ans-1.

a	b	Y	
0	0	1	→ I_0
0	1	0	→ I_1
1	0	0	→ I_2
1	1	0	→ I_3

$= \sum m(0)$

$\bar{a}$	(0)	1	
a	2	3	
$\bar{a}$	I_0		
0	I_1		



2.

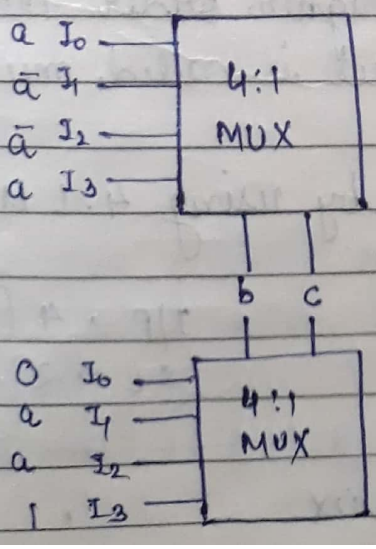
a	b	C_{in}
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

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

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

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

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

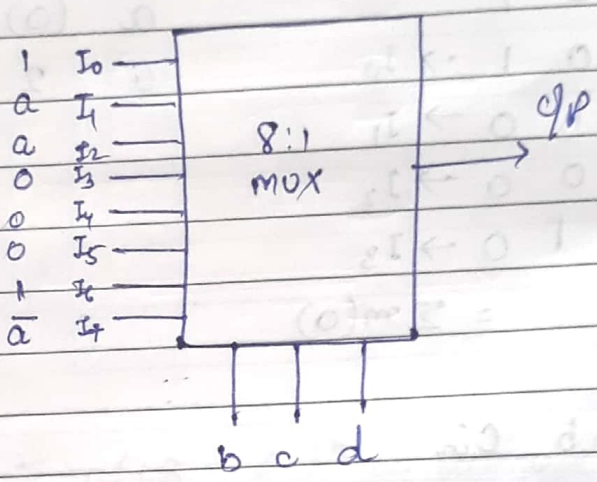


3. $Y = \pi M(1,2,3,4,5,11,12,13,15)$
$= \Sigma m(0,6,7,8,9,10,14)$

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	ϕ	
1	ϕ	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

	$\bar{a}$	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)
a	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	
1	a	a	0	0	0	1	$\bar{a}$		
0	$\bar{a}$	$\bar{a}$	1	1	1	0	a		

0 $\bar{a}$ $\bar{a}$ 1 1 1 0 a



Multiplexer Tree

Implementation of higher order multiplexer by using lower order multiplexer is called multiplexer tree.

Q. Implement 16:1 MUX by using 4:1 MUX only.

Ans- 16:1 → 4:1

I/P → 16 ($I_0 - I_{15}$)

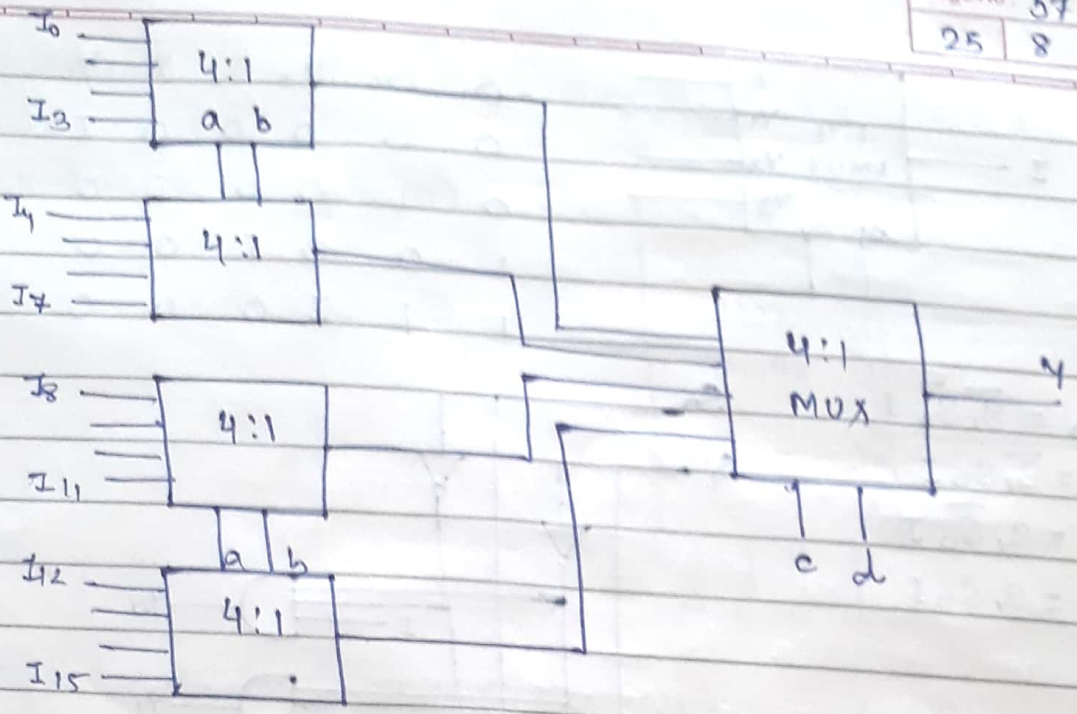
SL → 4 (a, b, c, d)

O/P → Y

$\frac{16}{4} = \frac{4}{1} = 1$ 4+1 = 5 MUX

I/P → 4 ($I_0 - I_3$)

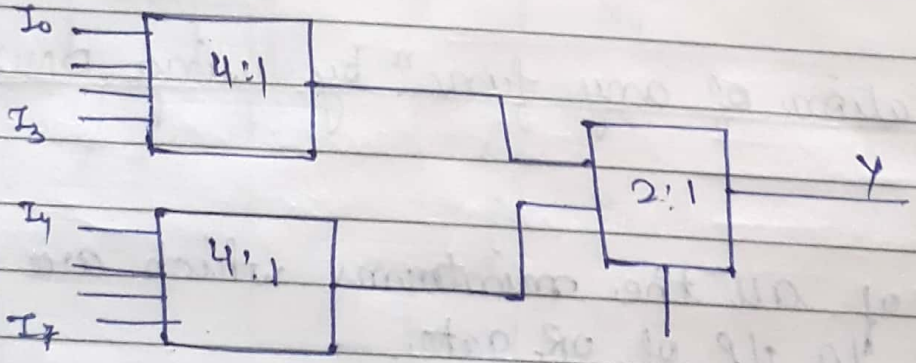
S.L → 2



Q. Implement 8:1 MUX by 4:1 MUX & 2:1 MUX.

Ans- $\frac{8}{4} = \frac{2}{2} = 1$

2+1 = 3 MUX



Multiplexer is also called Many-to-One, Data Selector & Parallel-to-Serial Convertor.

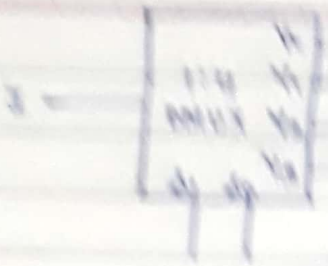
Demultiplexer

It is the reverse process of multiplexer. It has 1 I/P & m O/Ps and 's' selected lines.

$m = 2^s$

Eg- 1:2 DEMUX, 1:4, 1:8 DMUX

1:4 DMUX →



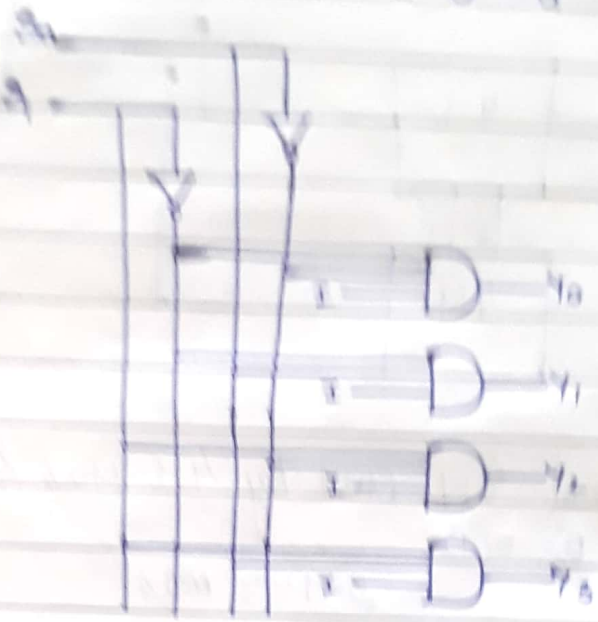
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

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

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

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

$Y_3 = S_1 S_0 I$

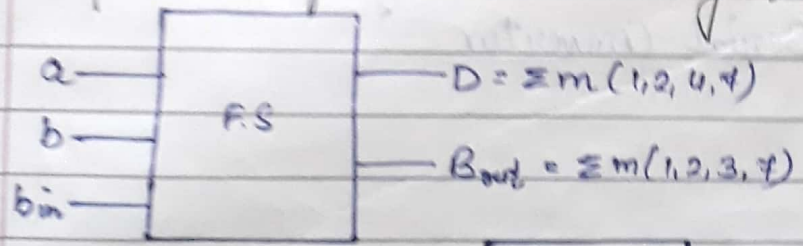


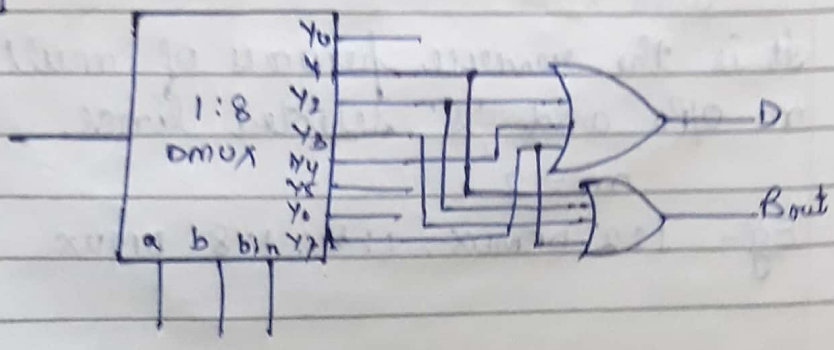
Implementation of any func^n by using DMUX / Decoder.

Connect o/p of all the minterms which are given in the func^n to i/p of OR gate.

Q. Implement full subtractor by using 1:8 DMUX.

Ans-



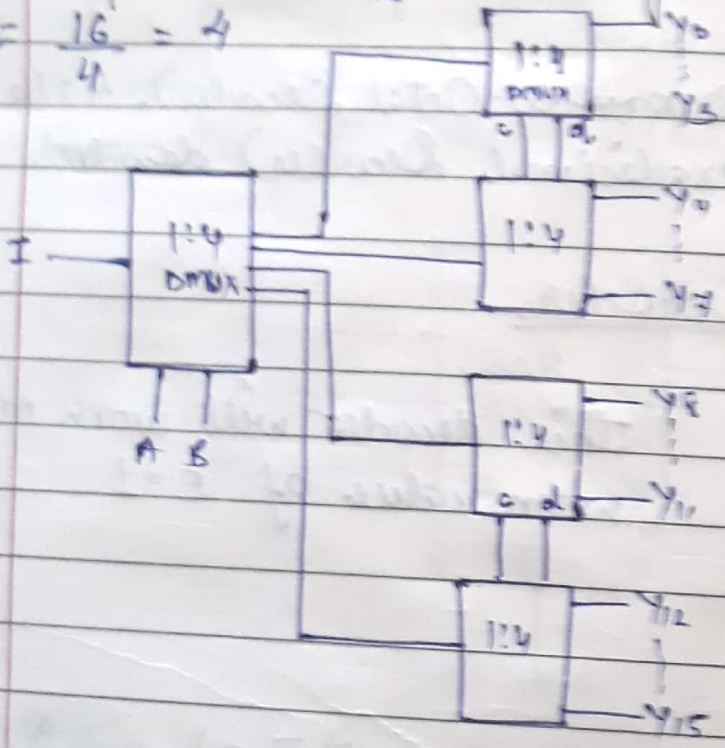


Q. Implement $Y = \Pi M(0,1,2,3,4,11,12,13,14,15)$ by using 1:16 DMUX.

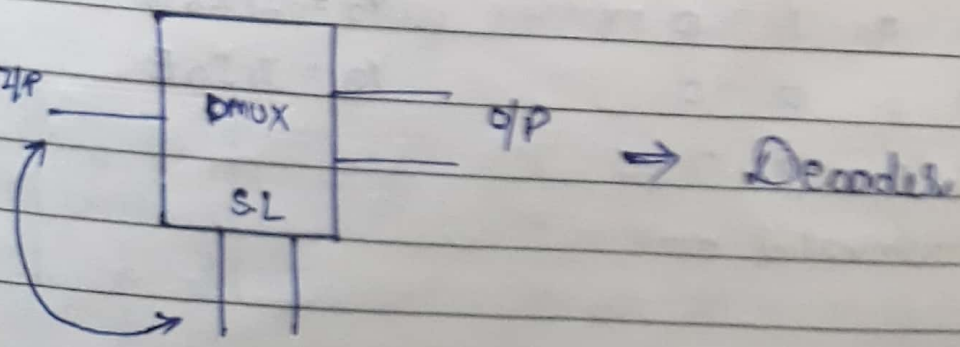
Sol- $Y = \Sigma m(5,6,7,8,9,10)$

Q. Implement 1:16 DMUX by using 1:4 DMUX.

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

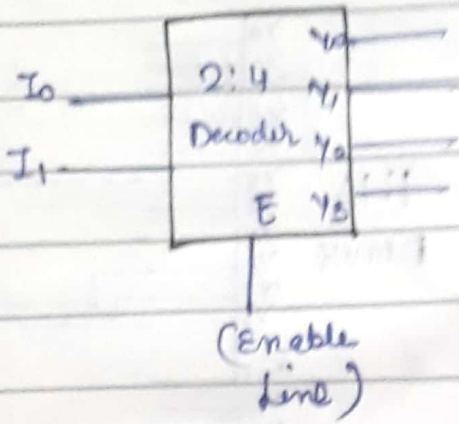


Decoder



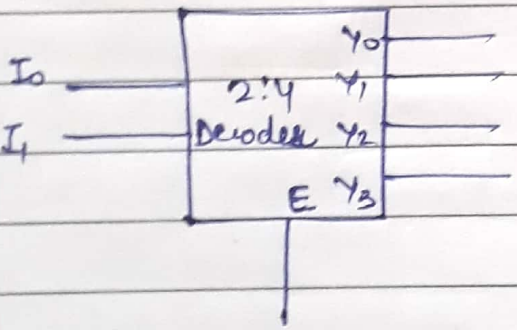
⇒ Decoder

I/P : O/P



It converts m I/P lines to 2^m O/P lines & it has one enable line.
The decoder only works when enable line is activated.
For eg, 2:4, 3:8 (Binary-to-Octal Decoder), 4:16 (Binary-to-Hexadecimal Decoder) decoder.

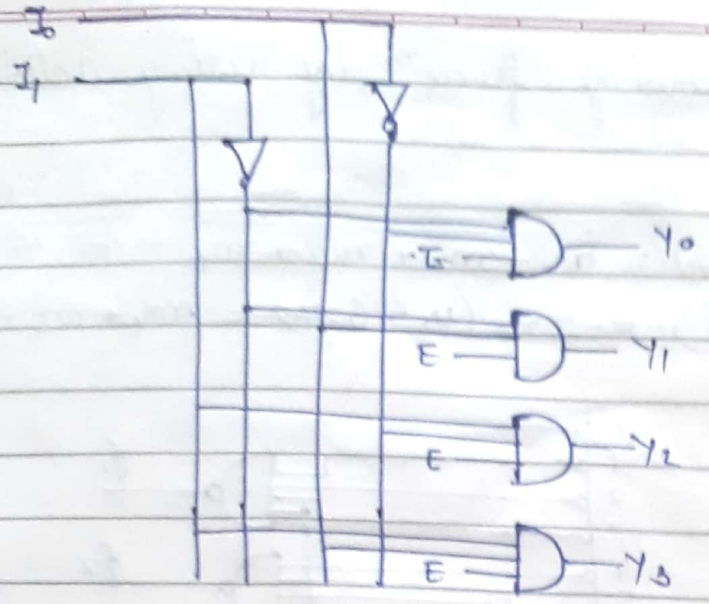
2:4 Active-High Decoder



This decoder will work only when value of E = 1.

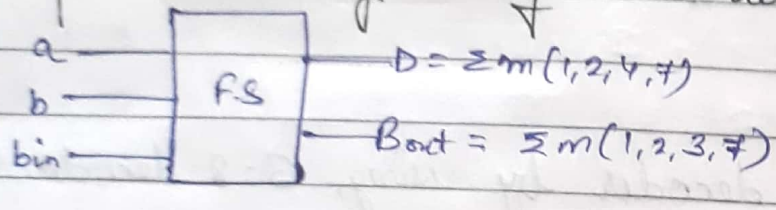
E	I_1	I_0	Y_3	Y_2	Y_1	Y_0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0
0	x	x	— Invalid —			

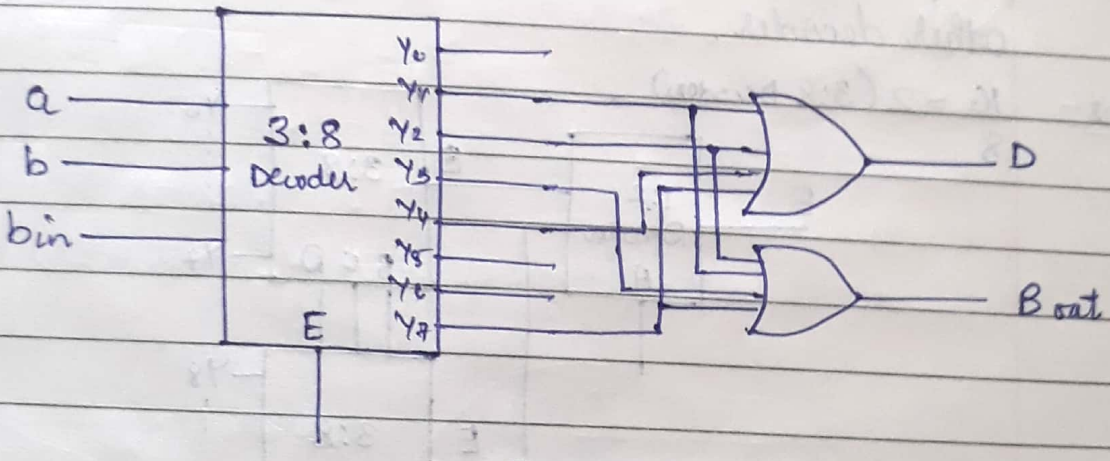
$Y_0 = \bar{I}_1 \bar{I}_0 E$
$Y_1 = \bar{I}_1 I_0 E$
$Y_2 = I_1 \bar{I}_0 E$
$Y_3 = I_1 I_0 E$



Q. Implement F.S by using 3:8 active-high decoder.

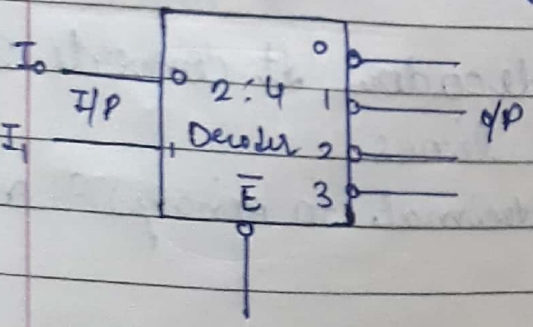
Ans-





Active Low Decoder

In this, decoder only works when value of enable line=0.



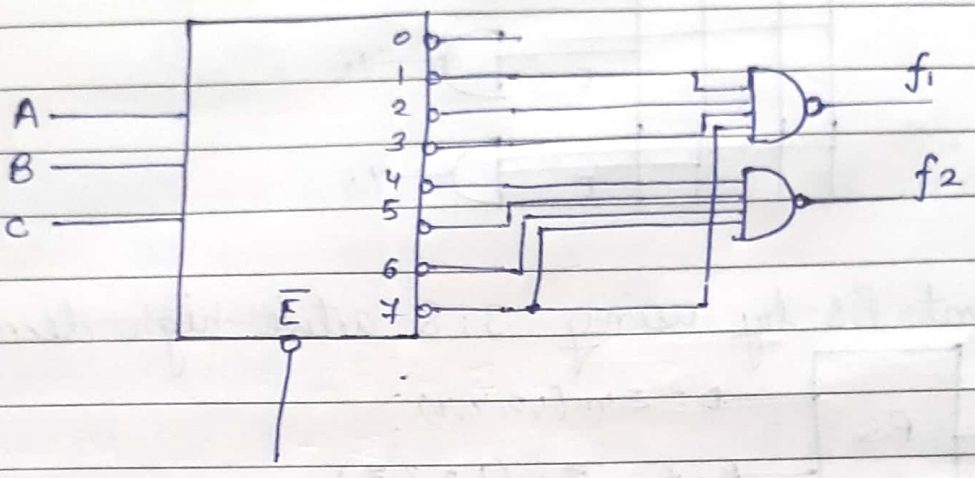
$\bar{E}$	I_1	I_0	Y_3	Y_2	Y_1	Y_0
1	x	x	— Invalid —			
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

Q Implement following func^{ns} by using active-low decoder :-

(i) $f_1 = \Sigma m(1,2,3,7) = \overline{m_1 + m_2 + m_3 + m_7} = \overline{\overline{m_1} \cdot \overline{m_2} \cdot \overline{m_3} \cdot \overline{m_7}}$ (NAND)

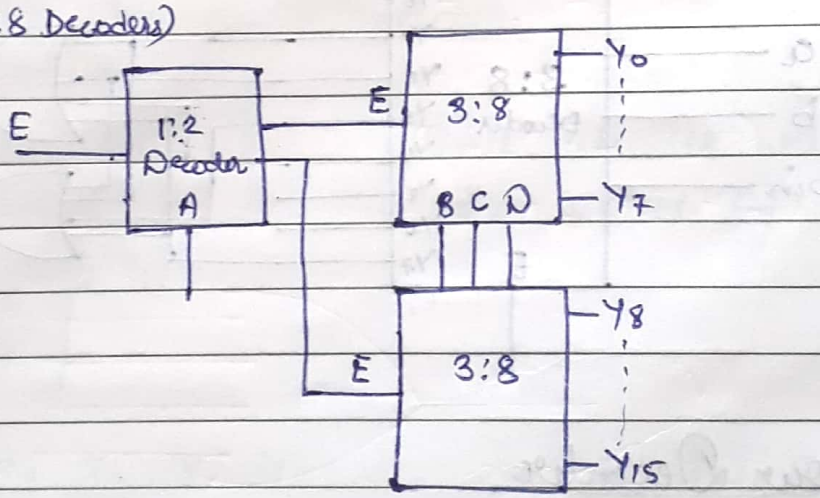
$f_2 = \pi M(0,1,2,3) = \Sigma m(4,5,6,7) = m_4 + m_5 + m_6 + m_7$

Ans-



Q Implement 4:16 decoder by using 3:8 decoder & other decoder.

Ans- $\frac{16}{8} = 2$ (3:8 Decoders)

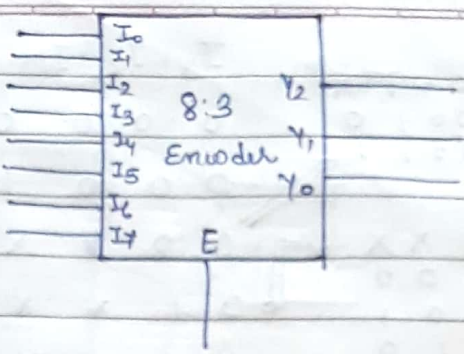


Encoder

It's the reverse process of decoder. It converts 2^m input lines into m output lines.

For eg, 4:2, 8:3, 16:4 (Hexadecimal to Binary) Encoders

8:3 Encoder →

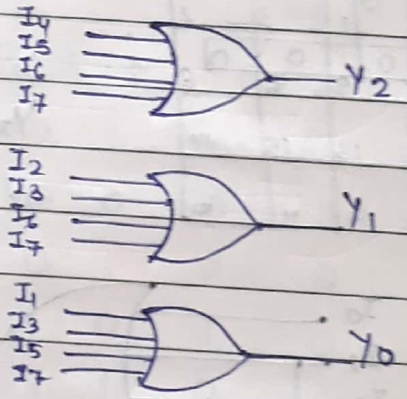


E	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0	(Active-High)
1	0	0	0	0	0	0	0	1	0	0	0	→ 0
1	0	0	0	0	0	0	1	0	0	0	1	→ 1
1	0	0	0	0	0	1	0	0	0	1	0	→ 2
1	0	0	0	0	1	0	0	0	0	1	1	→ 3
1	0	0	0	1	0	0	0	0	1	0	0	→ 4
1	0	0	1	0	0	0	0	0	1	0	1	→ 5
1	0	1	0	0	0	0	0	0	1	1	0	→ 6
1	1	0	0	0	0	0	0	0	1	1	1	→ 7

$Y_2 = I_4 + I_5 + I_6 + I_7$

$Y_1 = I_2 + I_3 + I_6 + I_7$

$Y_0 = I_1 + I_3 + I_5 + I_7$

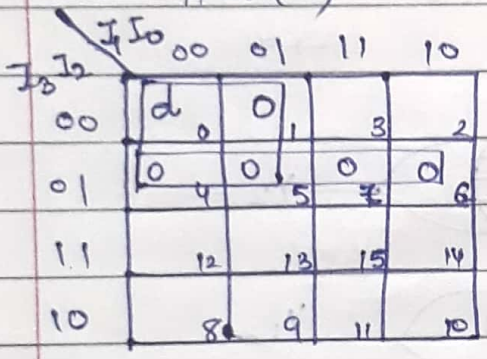


Priority Encoder

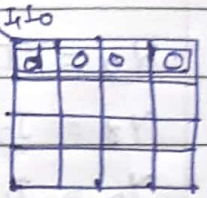
When more than 1 I/Ps are high, normal encoder is unable to give right O/P & this problem is avoided by priority encoder. In this, we set highest priority to that number which has highest binary value, then second highest binary value, & so on.

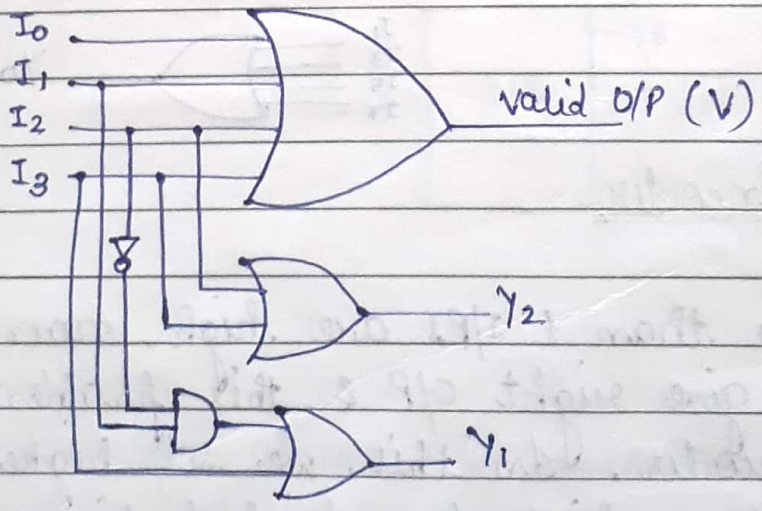
I_3	I_2	I_1	I_0	Y_2	Y_1	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

$$Y_1 = \pi M(1,4,5,6,7) + \Sigma d(0)$$
$$Y_2 = \pi M(1,2,3) + \Sigma d(0)$$
$$V = \pi M(0)$$



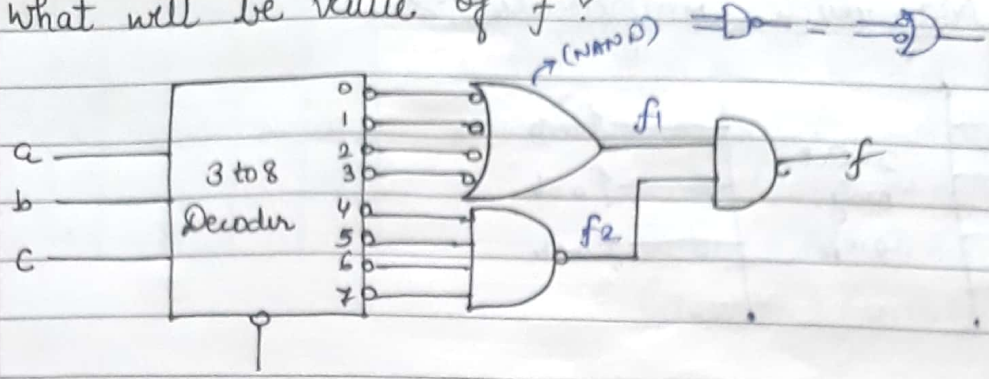
$$Y_1 = (I_3 + \overline{I_2})(I_3 + I_1)$$
$$= I_3 + I_1\overline{I_2}$$
$$Y_2 = I_3 + I_2 \Longrightarrow$$
$$V = I_3 + I_2 + I_1 + I_0$$





1	9	18

GATE Q8 What will be value of f?



Ans- $f_1 = \Sigma(0,1,2,3) = m_0 + m_1 + m_2 + m_3$

$f_2 = \Sigma(4,5,6,7) = m_4 + m_5 + m_6 + m_7 = \overline{f_1}$

$\therefore f = f_1 \cdot \overline{f_1} = 0.$

Magnitude Comparator

It is used to compare two numbers & gives three O/Ps : $f(a>b)$, $f(a=b)$ & $f(a<b)$.

1-Bit Magnitude Comparator →

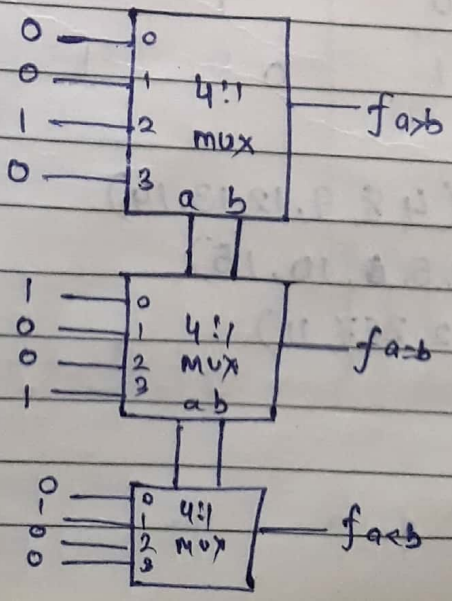
a	b	$f_{a>b}$	$f_{a=b}$	$f_{a<b}$
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$f_{a>b} = a\bar{b}$

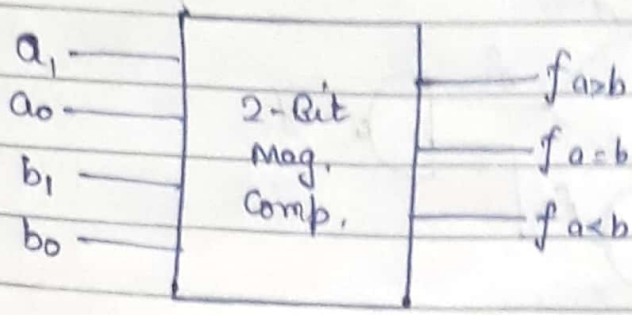
$f_{a=b} = a \odot b$

$f_{a<b} = \bar{a}b$

By 4:1 MUX :



2-Bit Magnitude Comparator →



I st NO.		II nd NO.		$f_{a>b}$	$f_{a=b}$	$f_{a<b}$
a_1	a_0	b_1	b_0			
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

$f(a>b) = \Sigma m(4, 8, 9, 12, 13, 14)$

$f_{a=b} = \Sigma m(0, 5, 6, 10, 15)$

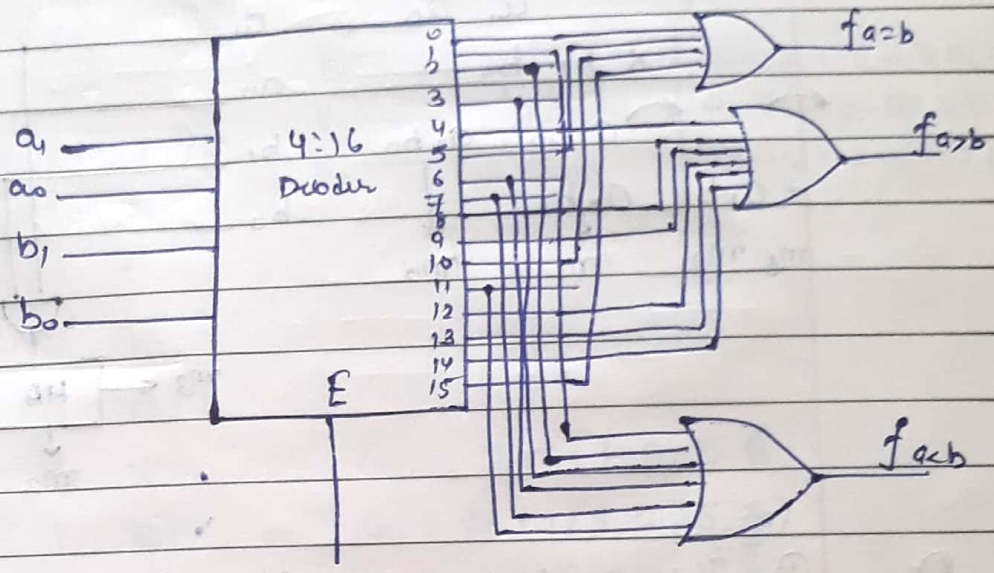
$f_{a<b} = \Sigma m(1, 2, 3, 7, 11)$

$f_{a=b}$

a_1a_0 \ b_1b_0	00	01	11	10
00	1			
01		1		
11			1	
10				1

$f_{a=b} = \bar{a}_1\bar{a}_0\bar{b}_1\bar{b}_0 + \bar{a}_1 a_0 \bar{b}_1 b_0 +$
$a_1\bar{a}_0 b_1\bar{b}_0 + a_1 a_0 b_1 b_0$
$= \bar{a}_1\bar{b}_1(\bar{a}_0\bar{b}_0 + a_0 b_0) + a_1 b_1(\bar{a}_0\bar{b}_0 + a_0 b_0)$
$= \bar{a}_1\bar{b}_1(a_0 \odot b_0) + a_1 b_1(a_0 \odot b_0)$
$= (a_0 \odot b_0)(a_1 \odot b_1)$

By Decoder :-



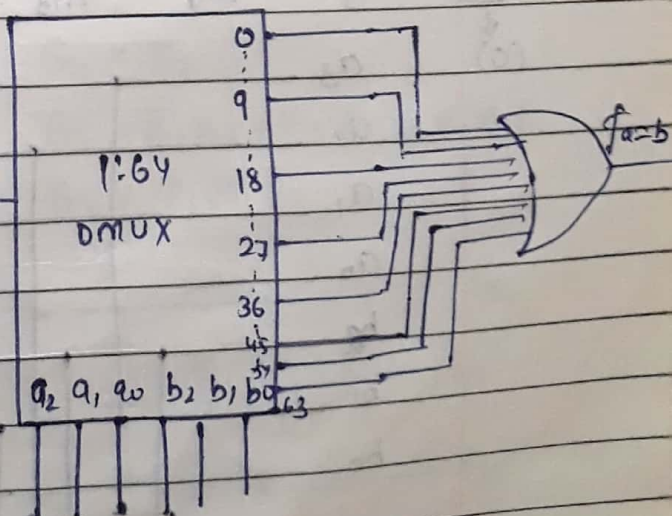
* Q: Implement $f_{a=b}$ of 3-bit mag. comp. by using 1:64 DMUX.

Ans- $f_{a=b} \Rightarrow a_2\ a_1\ a_0\ b_2\ b_1\ b_0$

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

(Divisible by 9)

$f_{a=b} = \Sigma m(0, 9, 18, 27, 36, 45, 54, 63)$



Binary Multiplier

It is a combinational circuit which multiplies two binary numbers of m & n bit.

1. 2 by 2 Bit Binary Multiplier

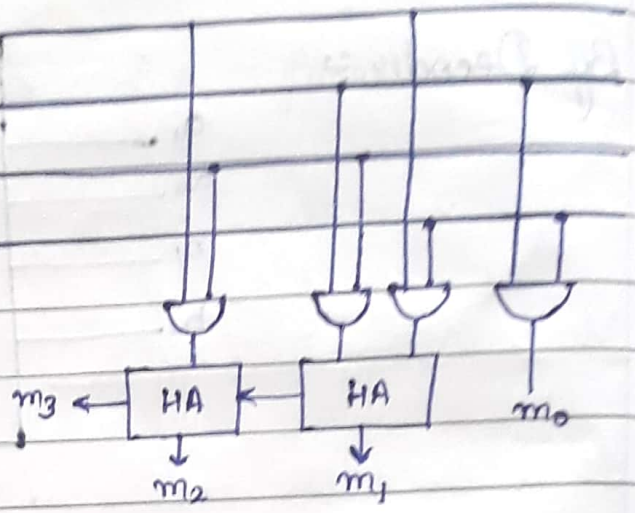
$a_1\ a_0$

$\times\ b_1\ b_0$

$a_1b_0\quad a_0b_0$

$+\ a_1b_1\quad a_0b_1$

$m_3\ m_2\quad m_1\quad m_0$



Q. Design a 4 by 3 bit Binary Multiplier.

Ans-

$a_3\ a_2\ a_1\ a_0$

$\times\ b_2\ b_1\ b_0$

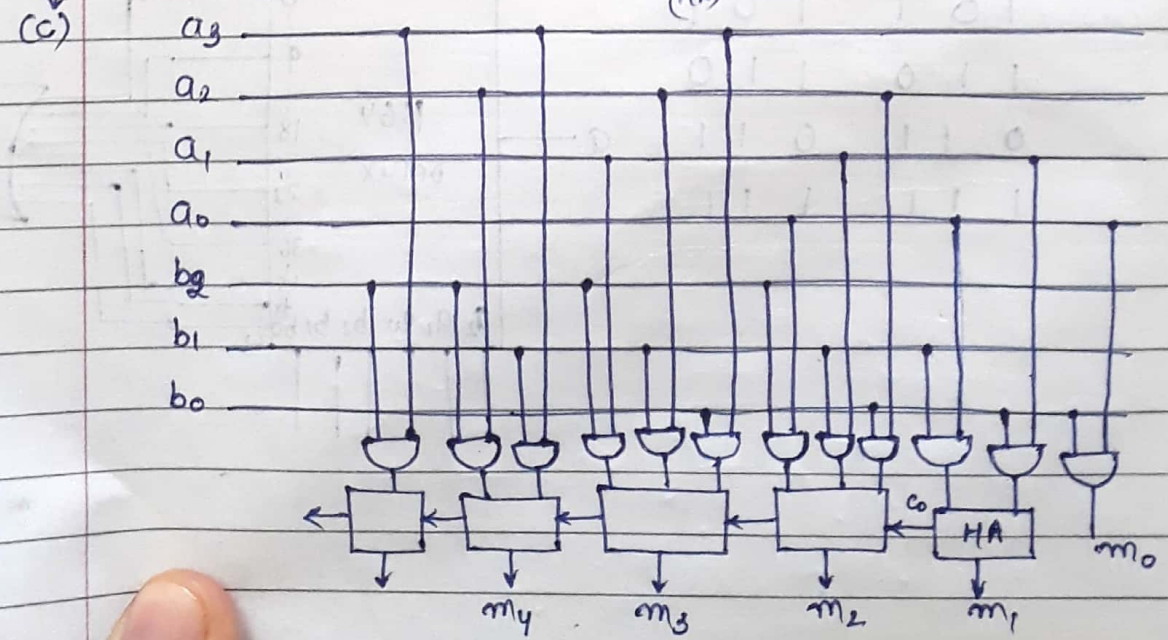
$a_3b_0\quad a_2b_0\quad a_1b_0\quad a_0b_0$

$+\ a_3b_1\quad a_2b_1\quad a_1b_1\quad a_0b_1$

$+\ a_3b_2\quad a_2b_2\quad a_1b_2\quad a_0b_2$

$m_6\ m_5\ m_4\ m_3\ m_2\ m_1\ m_0$

m_6 (C) m_1 (HA)



Code Convertor

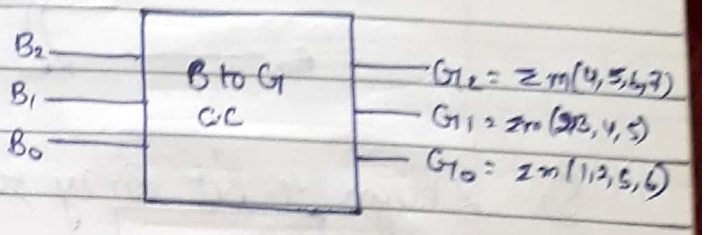
It converts one type of code into another type of code.

Q. Design 3-bit binary to gray code convertor.

Ans-

$(1\ 1\ 0\ 1)_2 \rightarrow (1\ 0\ 1\ 1)_{G_1}$

$(1\ 0\ 1\ 1)_{G_1} \rightarrow (1\ 1\ 0\ 1)_2$



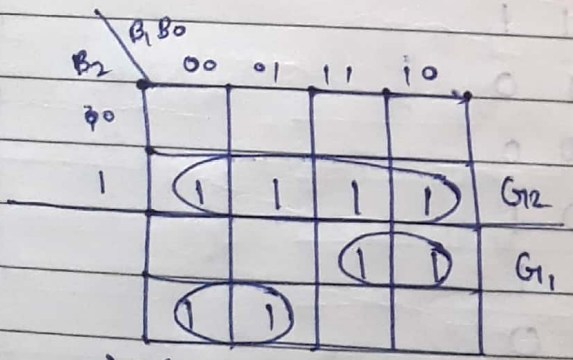
Truth Table →

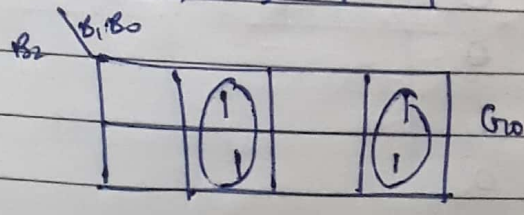
B_2	B_1	B_0	G_{12}	G_{11}	G_{10}
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

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

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

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

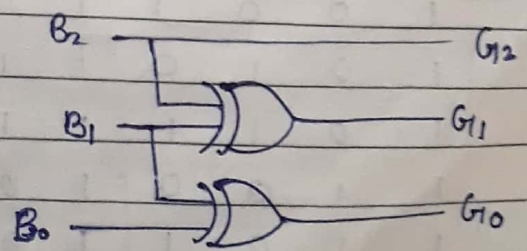




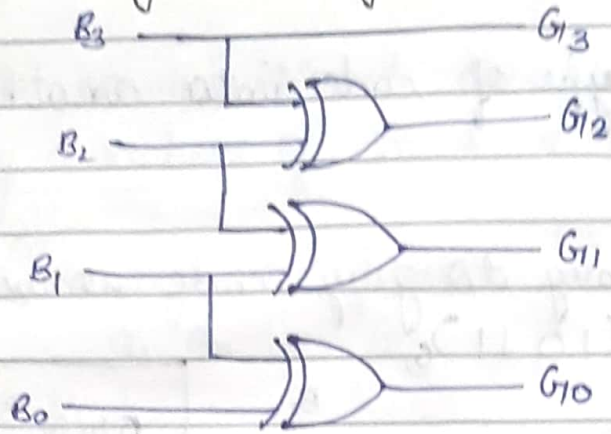
$G_{12} = B_2$

$G_{11} = \bar{B_2}B_1 + B_2\bar{B_1} = B_2 \oplus B_1$

$G_{10} = B_1 \oplus B_0$

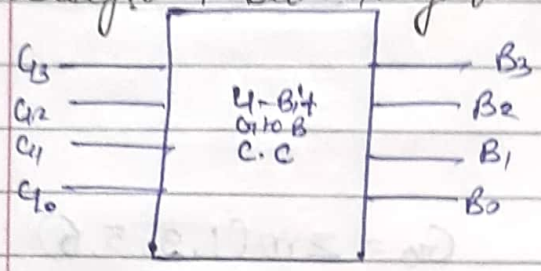


4-Bit Binary to Gray Code Convertor →



Q. Design 4-Bit Gray to Binary Code Convertor

Ans-



G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

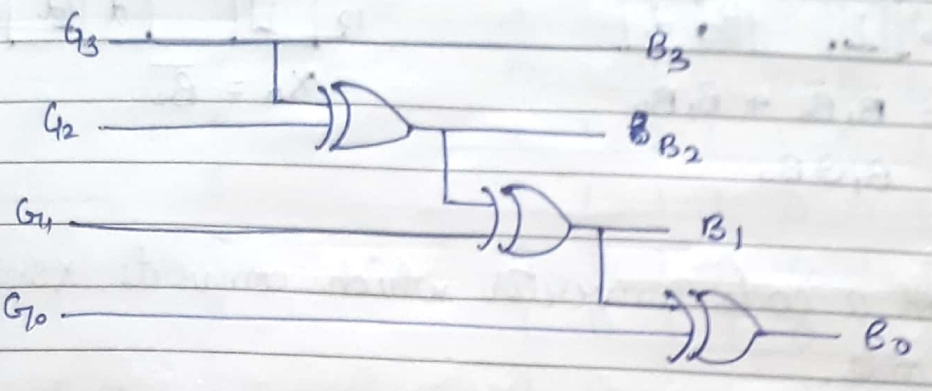
6	9	15

$B_3 = G_{13}$

$B_2 = B_3 \oplus G_{12} = G_{13} \oplus G_{12}$

$B_1 = B_2 \oplus G_{11} = G_{13} \oplus G_{12} \oplus G_{11}$

$B_0 = B_1 \oplus G_{10} = G_{13} \oplus G_{12} \oplus G_{11} \oplus G_{10}$



Q. Design a code convertor which converts a BCD code into XS-3 code.

Ans-

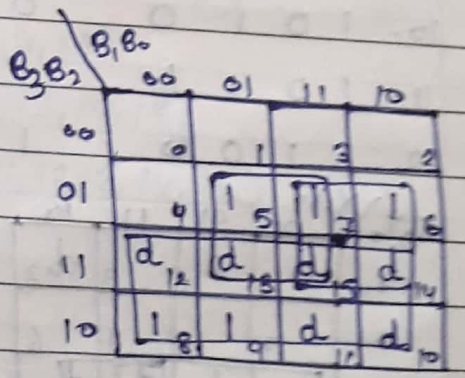
	$b_3 b_2 b_1 b_0$	$X_3 X_2 X_1 X_0$
0		0 0 1 1
1		0 1 0 0
2		0 1 0 1
3		0 1 1 0
4		0 1 1 1
5		1 0 0 0
6		1 0 0 1
7		1 0 1 0
8		1 0 1 1
9		1 1 0 0
10	→ d	
:		
15		

$X_3 = \Sigma m(5,6,7,8,9) + \Sigma d(10,11,12,13,14,15)$

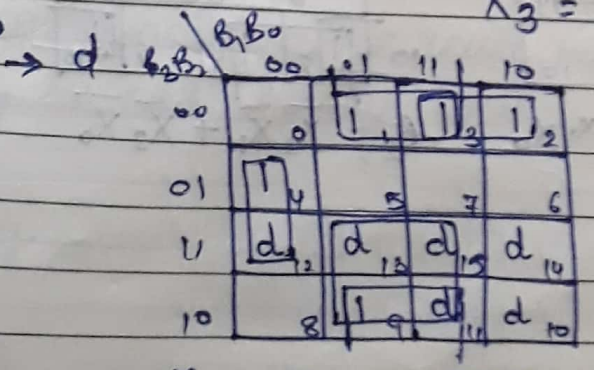
$X_2 = \Sigma m(1,2,3,4,9) + \Sigma d(10,11,12,13,14,15)$

$X_1 = \Sigma m(0,3,4,7,8) + \Sigma d(10,11,12,13,14,15)$

$X_0 = \Sigma m(0,2,4,6,8) + \Sigma d(10,11,12,13,14,15)$

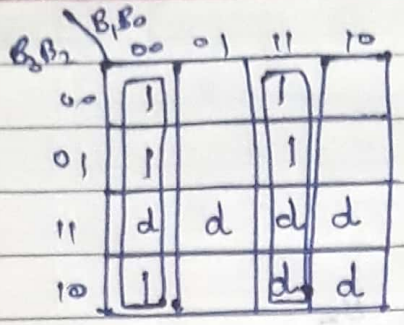


$X_3 = B_3 + B_2B_0 + B_2B_1$



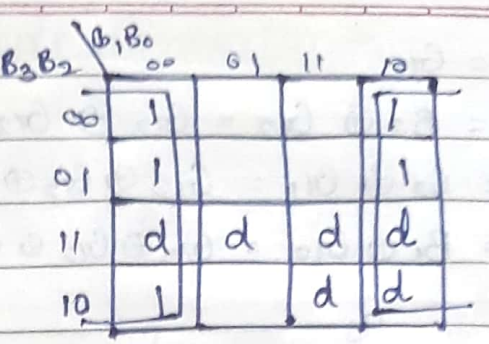
$X_2 = B_3B_0 + \overline{B_2}B_0 + B_2\overline{B_1}\,\overline{B_0} + \overline{B_2}B_2B_1$

$10_{BCD} \rightarrow 43_{BCD}$



$X_1 = \bar{B_1}\bar{B_0} + B_1 B_0$

$= B_1 \odot B_0$



$X_0 = \bar{B_0}$

Q Design a code converter which converts XS-3 code into BCD code.

Ans-

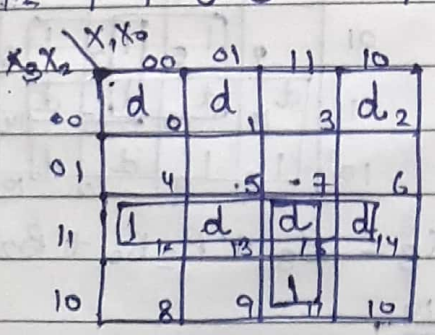
	X_3	X_2	X_1	X_0	B_3	B_2	B_1	B_0
3-	0	0	1	1	0	0	0	0
4-	0	1	0	0	0	0	0	1
5-	0	1	0	1	0	0	1	0
6-	0	1	1	0	0	0	1	1
7-	0	1	1	1	0	1	0	0
8-	1	0	0	0	0	1	0	1
9-	1	0	0	1	0	1	1	0
10-	1	0	1	0	0	1	1	1
11-	1	0	1	1	1	0	0	0
12	1	1	0	0	1	0	0	1

$B_3 = \Sigma m(11,12) + \Sigma d(0,1,2,13,14,15)$

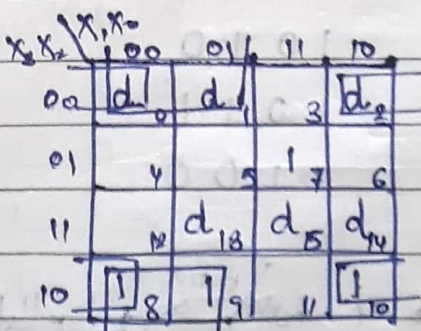
$B_2 = \Sigma m(7,8,9,10) + \Sigma d(0,1,2,13,14,15)$

$B_1 = \Sigma m(5,6,9,10) + \Sigma d(0,1,2,13,14,15)$

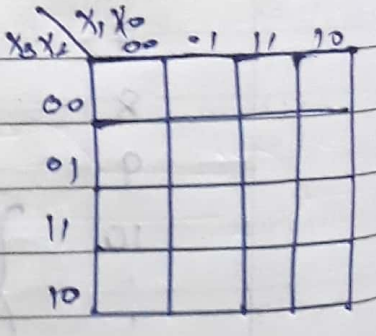
$B_0 = \Sigma m(4,6,8,10,12) + \Sigma d(0,1,2,13,14,15)$



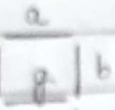
$B_3 = X_3 X_2 + X_3 X_1 X_0$



$B_2 = \bar{X_2}\bar{X_1} + \bar{X_2}\bar{X_0}$



BCD to 7-Segment Code Converter



BCD No.	Seven Segment Code						
	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

$a = \Sigma m(0,2,3,5,6,7,8,9) + \Sigma d(10,11,12,13,14,15)$

$b = \Sigma m(0,1,2,3,4,7,8,9) + \Sigma d(10,11,12,13,14,15)$

$c = \Sigma m(0,1,3,4,5,6,7,8,9) + \Sigma d(10,11,12,13,14,15)$

$d = \Sigma m(0,2,3,5,6,8,9) + \Sigma d(10,11,12,13,14,15)$

$e = \Sigma m(0,2,6,8) + \Sigma d(10,11,12,13,14,15)$

$f = \Sigma m(0,4,5,6,8,9) + \Sigma d(10,11,12,13,14,15)$

$g = \Sigma m(2,3,4,5,6,8,9) + \Sigma d(10,11,13,13,14,15)$

Parity Generator

This is a combinational circuit which generates either even parity or odd parity code.

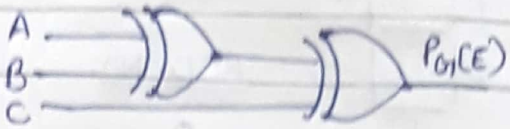
3-Bit Parity Generator (Even) →

I/Ps			Pa(E)
A	B	C	
0	0	0	0

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

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

$= A \oplus B \oplus C$



Parity Checker

Parity bit is added with message bits & it is transmitted from transmitter & at receiver, we check whether there is any error or not.

(4 I/P → 3 + 1 Parity Bit)

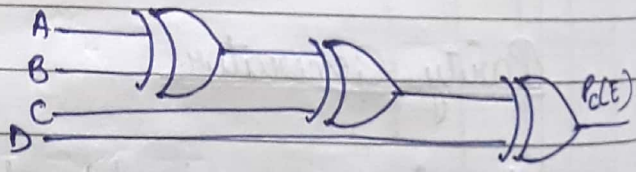
3-Bit Parity Checker →

I/Ps

A	B	C	D	$P_c(E)$	
0	0	0	0	0	(No Error)
0	0	0	1	1	(Error)
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	1	
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	1	
1	0	0	0	1	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	0	

$P_c(E) = \Sigma m(1,2,4,7,8,11,13,14)$

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

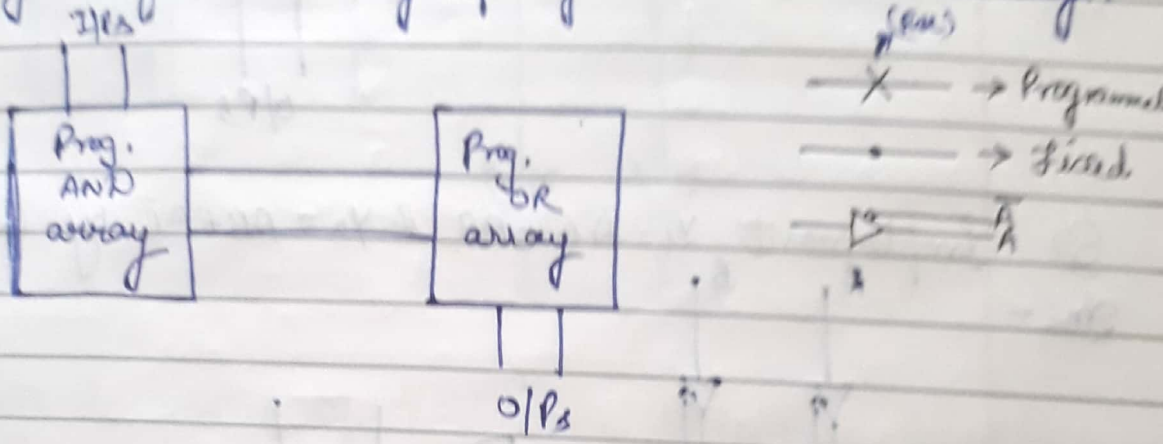


AB \ CD	00	01	11	10
00	0	1 1	3	1 2
01	1 4	5	1 7	6
11	12	1 13	15	1 14
10	1 8	9	1 11	10

$F_0(E) = A \oplus B \oplus C \oplus D$

Programmable Logic Array (PLA)

Programmable AND array is followed by programmable OR array.

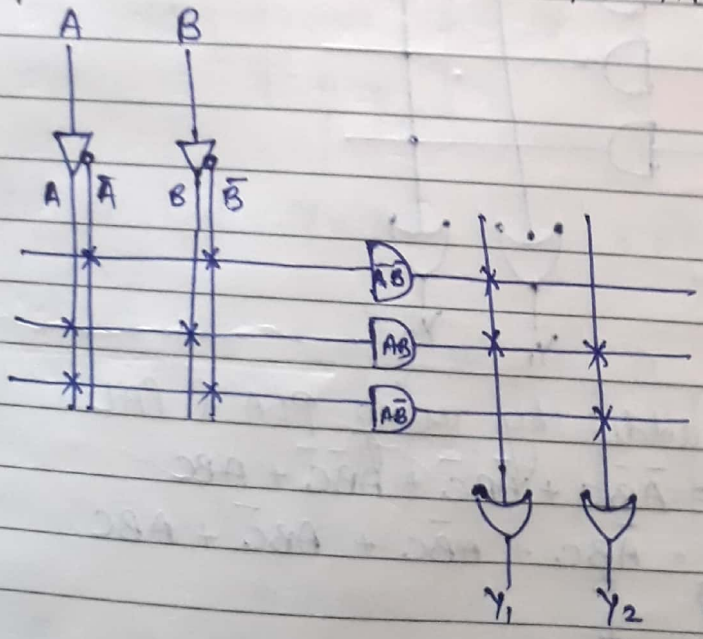


Qs Implement $Y_1 = \bar{A}\bar{B} + AB$ & $Y_2 = AB + A\bar{B}$ by using PLA.

$Y_1 = \Sigma m(0,3)$

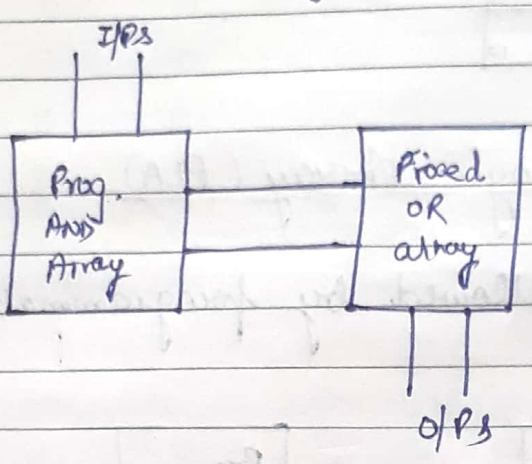
$Y_2 = \Sigma m(2,3)$

Ans



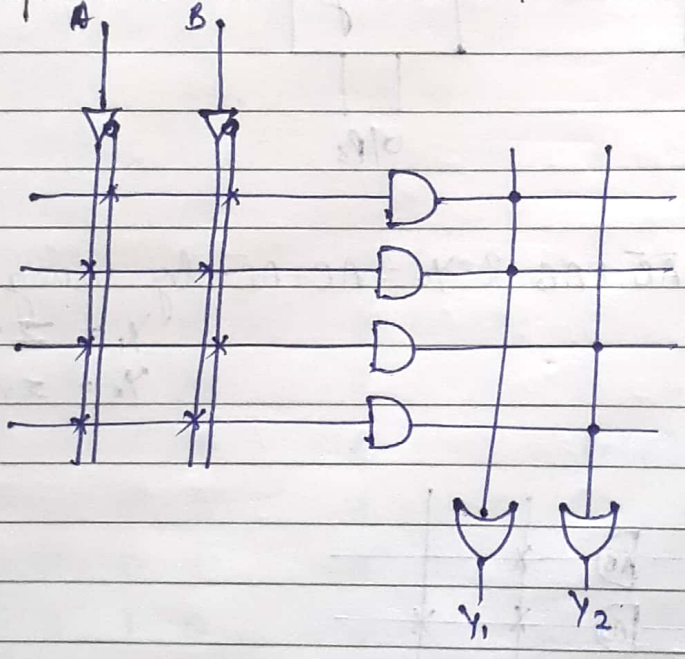
Programmable Array Logic (PAL)

Programmable AND array is followed by fixed OR array.



Q: Implement $Y_1 = \bar{A}\bar{B} + AB$ & $Y_2 = AB + A\bar{B}$ by using PAL.

Ans-

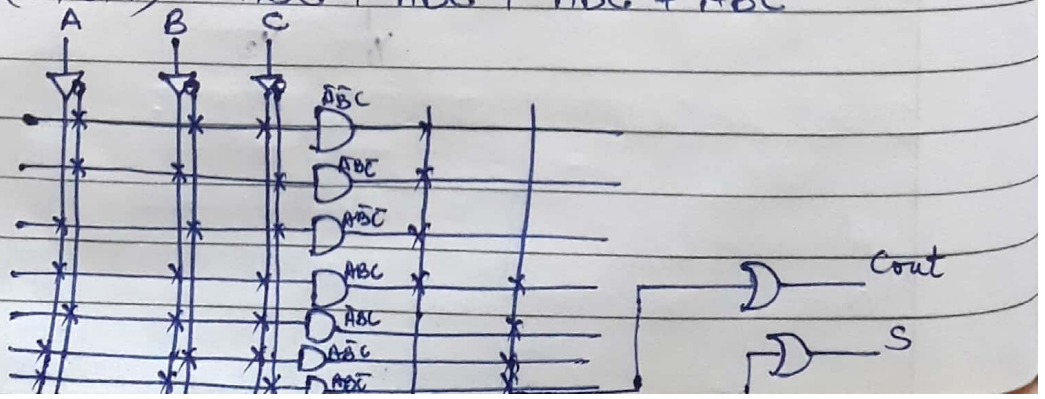


Q: Implement full adder by using PLA & PAL.

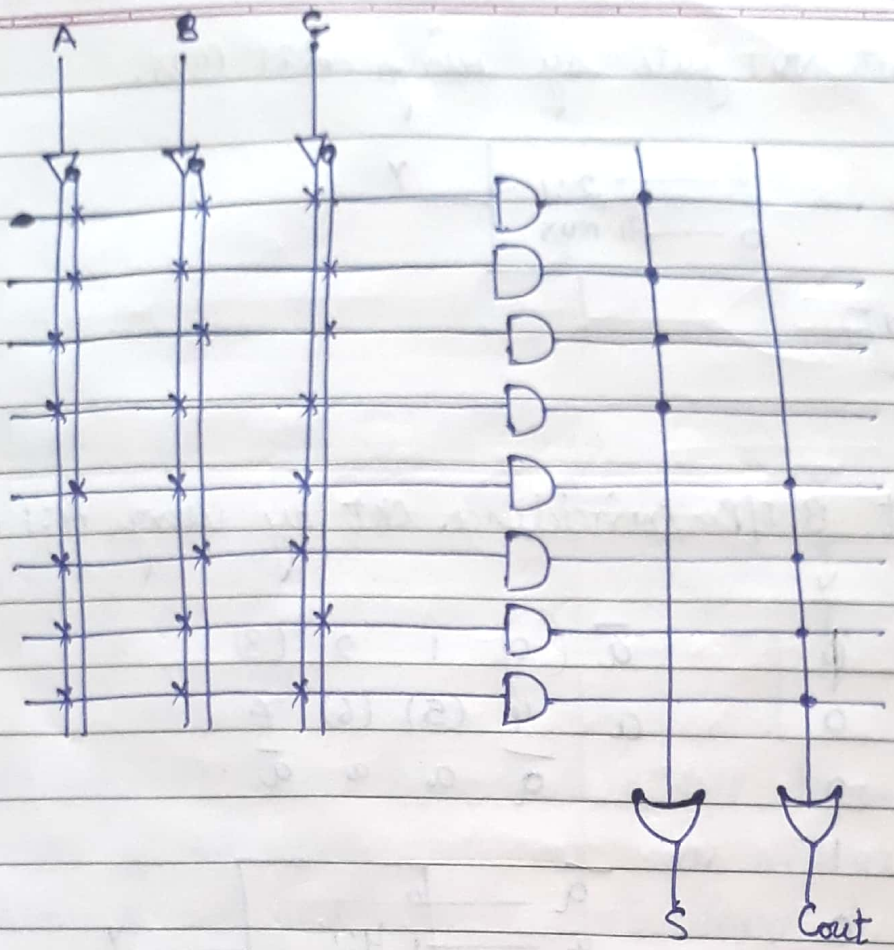
Ans- $S = \Sigma m(1,2,4,7) = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

$Cout = \Sigma m(3,5,6,7) = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

By PLA →

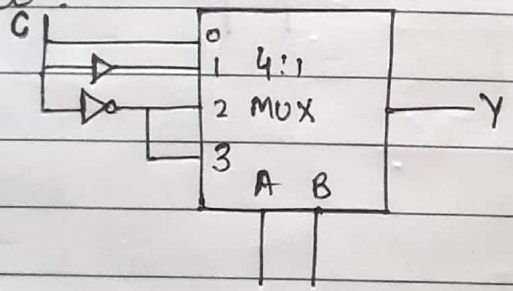


By PAL →



Revision...

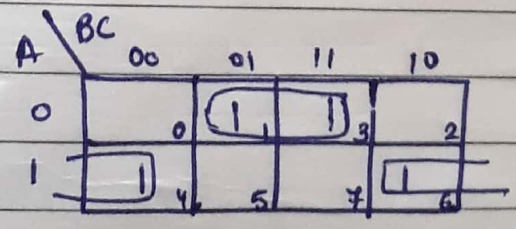
Q: What will be value of Y in the following multiplexer circuit?



Ans- ~~$Y = C+C+\bar{C}+\bar{C}$~~ $Y = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}\bar{C} + ABC$
(1) (3) (4) (6)

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

~~$= 1$~~



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

$= A \oplus C$

OR

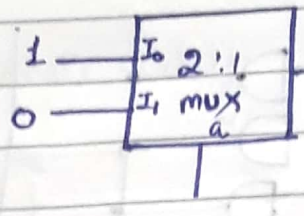
$\bar{C}$	0	2	(4)	(6)
C	(1)	(3)	5	7
	$\bar{C}$	$\bar{C}$	$\bar{C}$	$\bar{C}$
	↑	↑	↑	↑
	I_0	I_1	I_2	I_3

Page No : 78
14 | 9 | 18

Q. Implement NOT gate by using 2:1 MUX.

Ans-

X	Y
0	1
1	0



$Y = \bar{a} I_0 + a I_1$

(1 ↙ I_0, 0 ↙ I_1)

Q. Implement 3 I/P coincidence ckt by using 4:1 MUX.

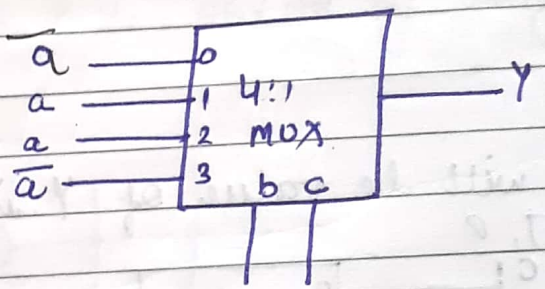
Ans-

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

$\bar{a}$ (0) 1 2 (3)

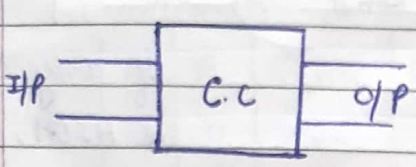
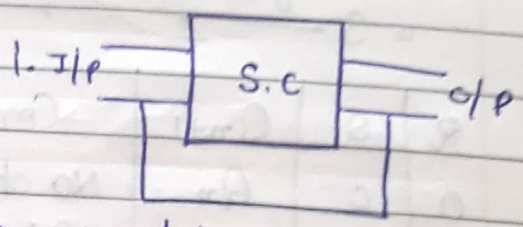
a 4 (5) (6) 7

$\bar{a}$ a a $\bar{a}$



UNIT-3

Difference b/w Combinational & Sequential Circuit

Combinational Ckt	Sequential Ckt
	
2. O/P depends upon present values of I/Ps only.	2. O/P depends upon present I/Ps as well as past values of O/Ps
3. There is no feedback path b/w I/Ps & O/Ps (means there is no memory element)	3. There is a feedback path b/w I/Ps & O/Ps (means there is memory element present in it)
4. Easy to design.	4. Difficult to design.
5. Less hardware is required.	5. More hardware is required.
6. Speed is more.	6. Speed is less.
Eg- MUX, DMUX, Decoder	Eg - Flip-flop

Flip-Flop

It is the smallest unit of memory & it stores one bit at a time and it has two stable states, that's why it is also called Bistable multivibrator.

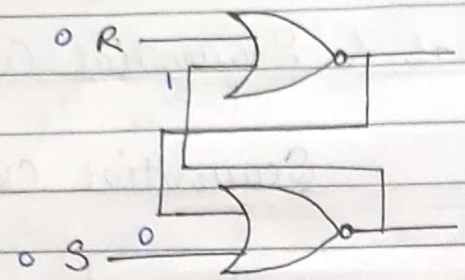
* Flip-flop has 2 O/Ps - Q_n & $\overline{Q_n}$ and one O/P is complement of other O/P.

If $Q_{n+1} = 1$ & $\overline{Q}_{n+1} = 0$, this condition is called Set Cond^n

(Q_{n+1} ↓ (Next state))

If $Q_{n+1}=0$ & $\overline{Q_{n+1}}=1$, this cond^n is called Reset Cond^n

R-S Latch (NOR based)



Q_n Q_{n+1} ← (Next value)

0 0

$\overline{Q_n}$ $\overline{Q_{n+1}}$

1

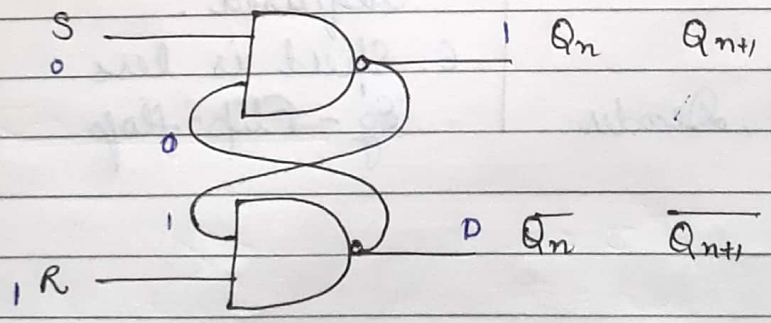
a	b	y
0	0	1
0	1	0
1	0	0
1	1	0

If any one I/P is HIGH, O/P is LOW.

R	S	Q_{n+1}	Comment
0	0	Q_n	No change (memory retained)
0	1	1	Set cond^n
1	0	0	Reset cond^n
1	1	X	Forbidden (Invalid or Indeteseminate)

If $Q_n=0$, $Q_{n+1}=0$ } $Q_{n+1}=1$
If $Q_n=1$, $Q_{n+1}=1$ } if $R=S=0$

NAND based S-R Latch



Q_n Q_{n+1}

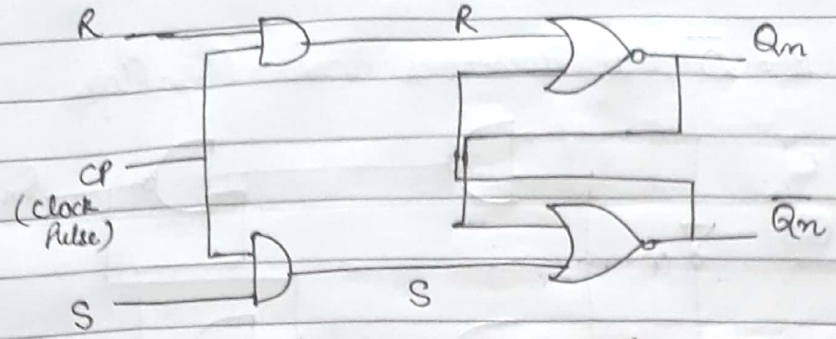
$\overline{Q_n}$ $\overline{Q_{n+1}}$

a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

If any one I/P is LOW, O/P is HIGH.

S	R	Q_{n+1}	Comment
0	0	X	forbidden
0	1	1	Reset
1	0	0	Set
1	1	Q_n	No change

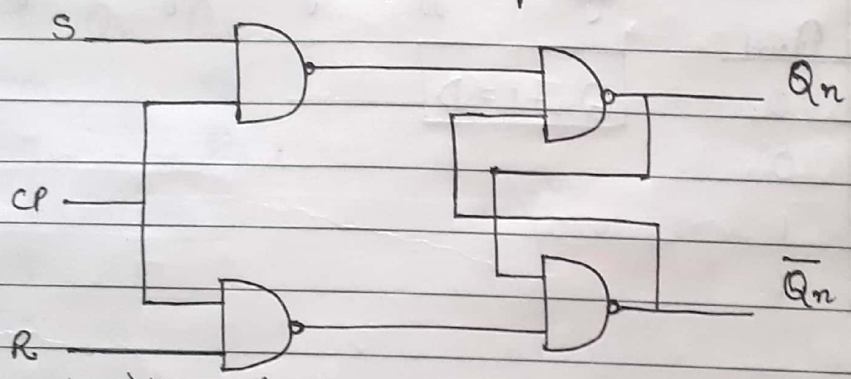
NOR Based R-S Flip-Flop



Characteristic Table of R-S Flip-Flop →

CP	S	R	Q_{n+1}	Comment
1	0	0	Q_n	No change
1	0	1	0	Reset Condⁿ
1	1	0	1	Set Condⁿ
1	1	1	X	Forbidden

NAND Based S-R Flip-Flop



Characteristics Table of S-R Flip-Flop (NAND/NOR) & Characteristics Eqⁿ of S-R Flip-Flop →

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

Char. Eqⁿ → $Q_{n+1} = f$ (ff i/ps, Q_n)

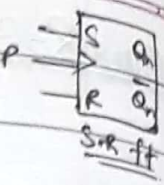
$Q_{n+1} = \Sigma m(1,4,5) + \Sigma d(6,7)$

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

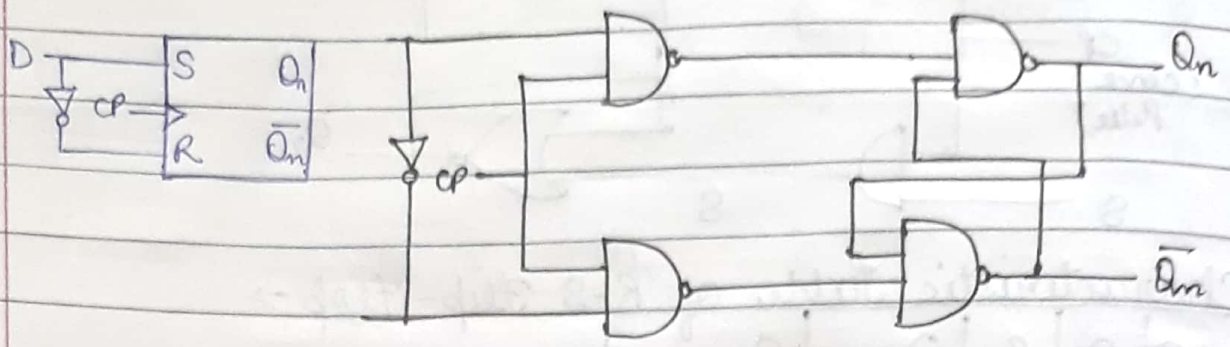
S \ RQn	00	01	11	10
0	0	1 (1)	3	2
1	1 (4)	1 (5)	d (7)	d (6)

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

D-Flip-Flop



1) $S = D$, $R = \bar{D}$, it becomes a D-Flip-Flop



Characteristic Eq^n →

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

∵ We're getting O/P after some delay, it is called Delay f-f or Transparent Latch.

Eg - Traffic Light Signal System

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

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

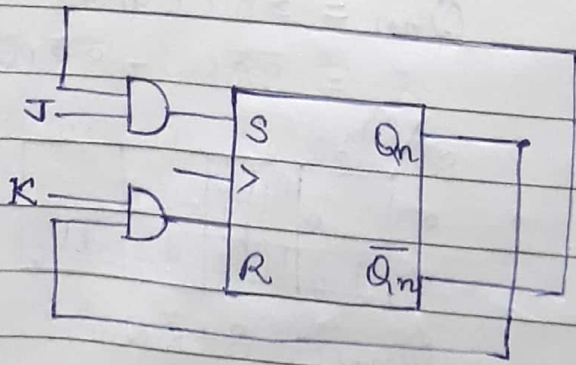
J-K Flip-Flop

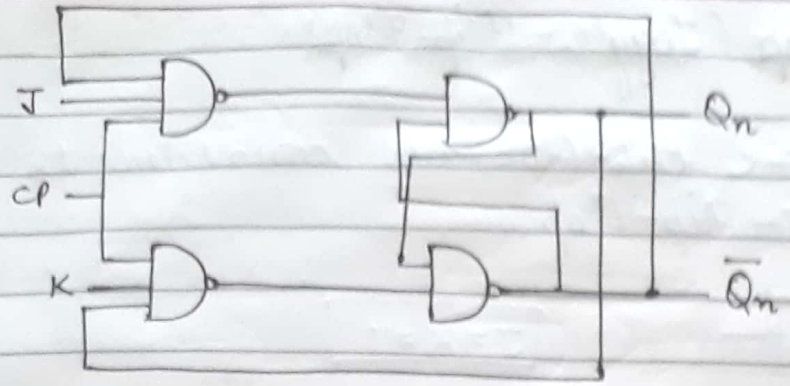
S J → Q_n

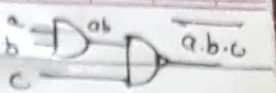
R K → $\bar{Q}_n$

$S = J\bar{Q}_n$

$R = KQ_n$







Characteristic Table of J-K Flip-Flop →

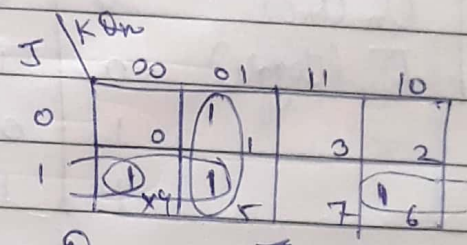
CP	(S) J	(R) K	Q_{n+1}	Comment
↑	0	0	Q_n	N.C
↑	0	1	0	R
↑	1	0	1	Set
↑	1	1	$\bar{Q}_n$	Toggle (Complement)

If $Q_n = 0, S = 1, R = 0 \Rightarrow Q_{n+1} = 1$
$Q_n = 1, S = 0, R = 1 \Rightarrow Q_{n+1} = 0$
$Q_{n+1} = \bar{Q}_n$

Characteristic Eq →

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	$\bar{Q}_n$
1	1	1	$\bar{Q}_n$

$Q_{n+1} = \bar{J}K$ $\Sigma m(1, 4, 5, 6)$



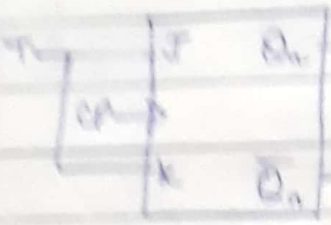
$Q_{n+1} = \bar{J}K + \bar{K}Q_n + J\bar{Q}_n$

(or)

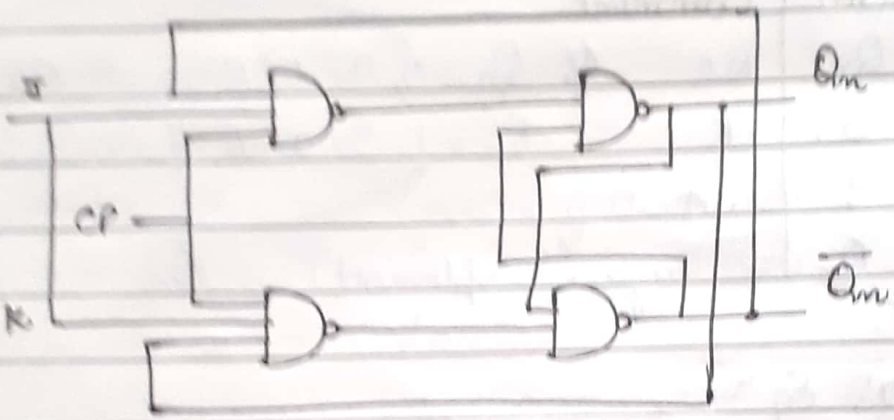
$Q_{n+1} = S + \bar{R}Q_n$
$= J\bar{Q}_n + \overline{KQ_n}\,Q_n$
$= J\bar{Q}_n + (\bar{K} + \bar{Q}_n)Q_n$
$= J\bar{Q}_n + \bar{K}Q_n$

T - Flip-Flop (Toggle - Flip-Flop)

If I/Ps of J-K flip-flop are connected to T - I/P, it becomes T-flip-flop.



$$J = K = T$$



Characteristics Eq^{n} →

clk	T	Q_{n+1}
↓	0	Q_n (N.C)
↓	1	$\overline{Q_n}$ (Toggle)

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

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

or

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

Excitation Table of Flip-Flop

In this, with the help of Q_n & Q_{n+1}, we determine value of Flip-Flop I/Ps.

Q_n	Q_{n+1}	D	T	S	R	J	K
0	0	0	0	0	d	0	d
0	1	1	1	1	0	1	d
1	0	0	1	0	1	d	1
1	1	1	0	d	0	d	0

J K	S R	Q_n	Q_{n+1}
0 0	0 0	0	0
0 0	0 0	1	1
0 1	0 0	0	0
0 1	1 0	1	0
1 0	0 1	0	1
1 0	0 1	1	1
1 1	1 0	0	1
1 1	1 0	1	0

Conversion of Flip-Flops

Steps → 1. Determine Characteristic table of required Flip-Flop.
2. Determine Excitation Table of available Flip Flop & combine both above two tables.
3. Now, with the help of K-Map or Boolean Algebra, determine I/P of available Flip-Flop which is function of required f-f I/Ps & Q_n.
4. Implement required f-f with the help of given (available) f-f & suitable logic gate.

D to T

(available) to (required)
Ext. Table Char. Table

Q1. Convert D-f-f into T-f-f.

Ans

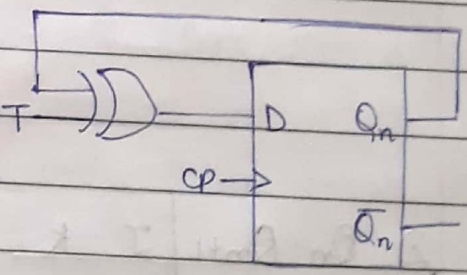
C.T of T-ff →

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

← ET of D ff

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

$= T \oplus Q_n$



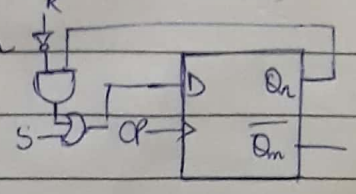
Q2. (i) Convert D-f-f to S-R f-f
(ii) Convert JK f-f to D-f-f.
(iii) Convert T to J-K f-f.

Ans - (i)

S	R	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	x	x
1	1	1	x	x

$D = \Sigma m(1,4,5) + \Sigma d(6,7)$

$= S + \bar{R} Q_n$

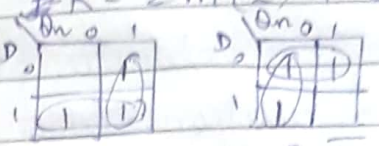


(ii)

Q_n	Q_{n+1}	D	J	K
0	0	0	0	d
0	1	1	1	d
1	0	0	d	1
1	1	1	d	0

$J = \Sigma m(1) + \Sigma d(2,3)$

$K = \Sigma m(2) + \Sigma d(0,1)$

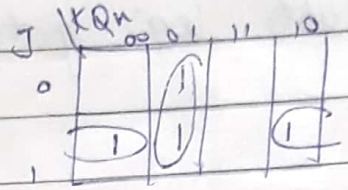


$J = D + Q_n$ $\quad K = \bar{D} + \bar{Q_n}$

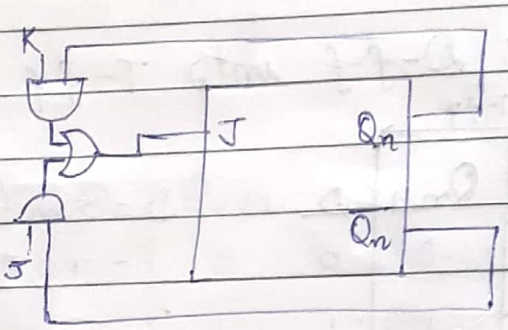
(iii) $T \rightarrow (Q_n \oplus Q_{n+1})$

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

$T = \Sigma m(3,4,6,7)$



$T = J\bar{Q}_n + KQ_n$

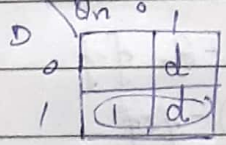


(i)

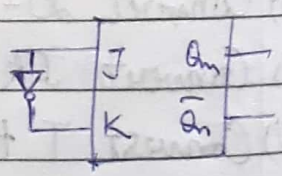
D	Q_n	Q_{n+1}	J	K
0	0	0	0	d
0	1	0	d	1
1	0	1	1	d
1	1	1	d	0

$J = \Sigma m(2) + \Sigma d(1,3)$

$K = \Sigma m(1) + \Sigma d(0,2)$

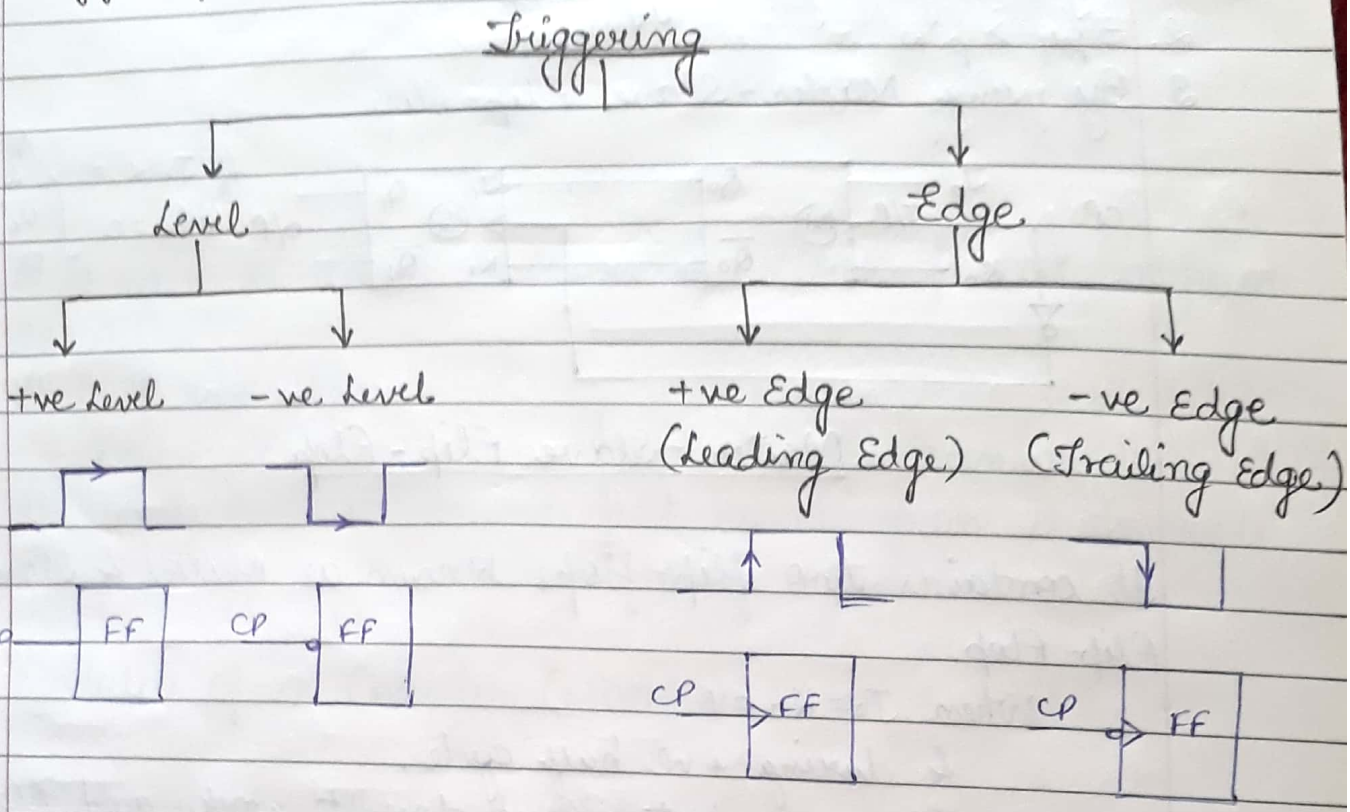


$J = D, \quad K = \bar{D}$



Before CP is applied, value is 0 (Reset)

Triggering of Flip-Flop



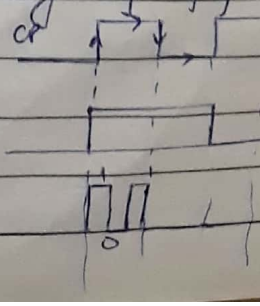
In level triggering, O/P may change more than one time in a single clock pulse and this condition arises race around condition.
In Edge triggering, O/P changes only once in a single clock pulse.

Race Around Condition →

In Race Around Condition, O/P changes many times in a single clock pulse. The following regions are race around condition :-

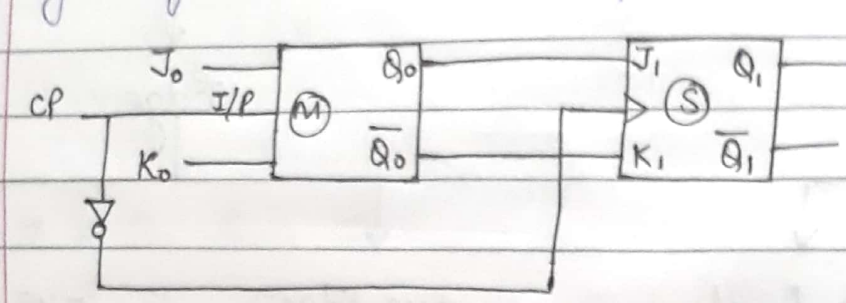
1. When we use level triggering.
2. When J=K=1 and propagation delay of ff is less than pulse width of clock pulse.

$t_{pdf} < p.w.$



Q. How race around condition can be avoided?

Ans- 1. By using edge triggering.
2. $t_{pd} > p.w$
3. By using Master-Slave Flip-Flop



Master-Slave Flip-Flop

It contains two flip-flops known as Master & Slave Flip-Flop.

When $J_0 = K_0 = 1$

& during +ve half cycle,
Master FF works but Slave doesn't work and Master gives O/P,

& during -ve half cycle,
Master FF doesn't work, but Slave FF works and it copies instructions given by Master & gives final O/P.

Hence, in single CP, O/P changes only once, so, it avoids race around condition.

UNIT-4

Counter

Counter

It is used to count number of clock pulses applied to it.
It is of two types :-

Down Counter	Up Counter
1. It counts from N-1 towards 0.	1. It counts from 0 towards N-1.

Modulus of a Counter (MOD-N)

Total number of used states is called Modulus of a counter.
For eg, in MOD-5, there are total 5 used states and these are shown as below :-

$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ (with a return arrow from 4 to 0)

* $N \leq 2^n$ → (No. of ff required) $= \log_2 N \leq n$

(N: Total used states (OR modulus of a Counter))

Eg- $5 \leq 2^3$

$\therefore n = 3$ FF required.

Again, counter is divided into 2 types :-
a) Asynchronous Counter
b) Synchronous Counter

Asynchronous Counter

This is also called Serial Counter / Ripple Counter / Frequency Divider.

Q: Design MOD-8 async. counter (UP).

Ans - 0 → 1 → 2 → 3 → 4 → 5 → 6 → 7

↑ ___ ↓

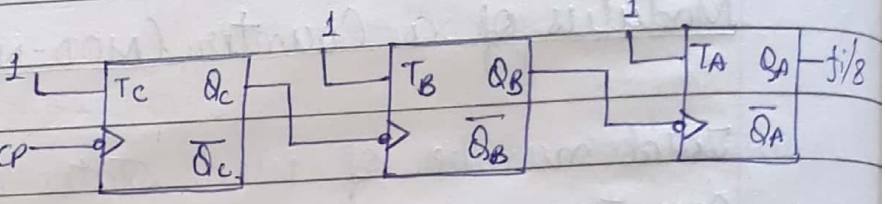
N-1 → 8-1 ⇒ 7 (111) (OR) N ≤ 2^n ⇒ 8 ≤ 2^3 → 3 F.Fs req.

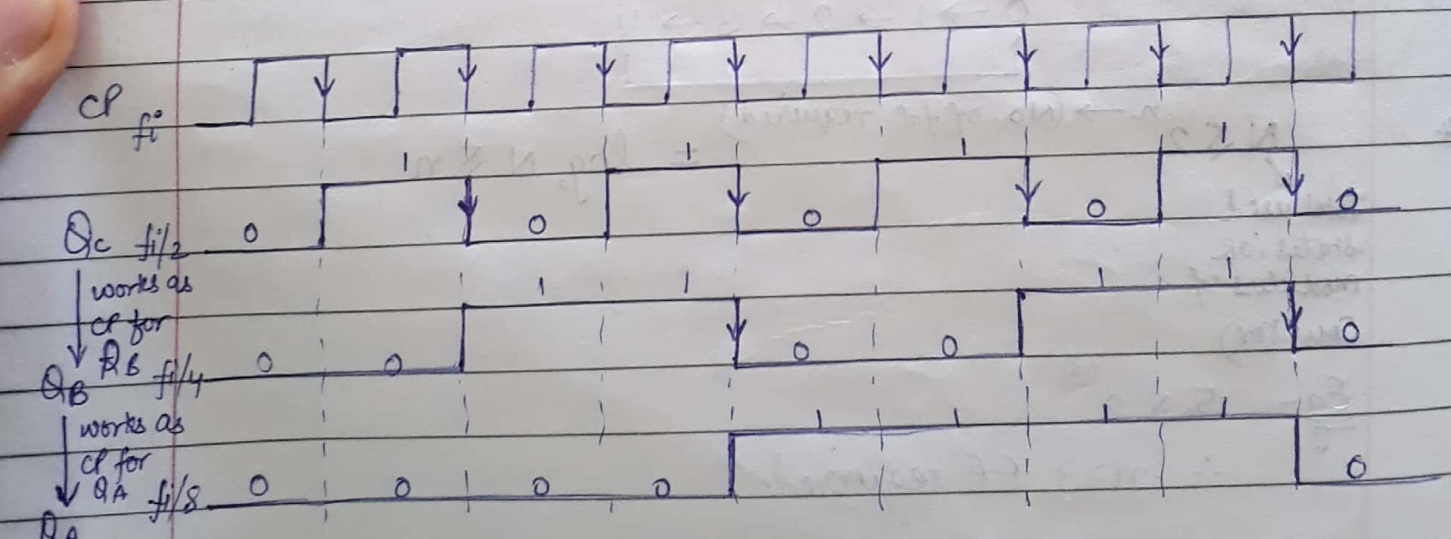
3 F.Fs are required.

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

Always from 1 to 0, then Q_B is toggling & Q_C is acting as clk for Q_B.

$Q_{n+1} = \overline{Q_n}$





$$\text{o/p freq.} = \frac{fi}{\text{No. of used states}}$$

Flip flop is MOD-2 Counter.

If $N = 2^n$, it is called n-bit binary counter.
Eg- MOD-8 can be called 3-bit binary counter.

* $T_{clock} \geq n\, t_{pdf}$
(Propagation delay of ff)
(No. of f-fs)

f_i → [MOD M Counter] → [MOD N Counter] → f_i/MN ⇒ [MOD MN Counter]

* Q. What will be O/P frequency in the following circuit if I/P frequency is 1 MHz?

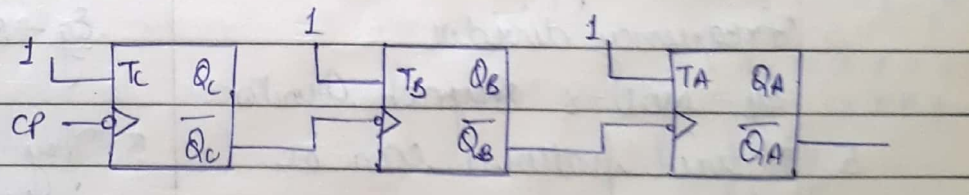
$f_i = 1$ MHz → [MOD 10 Counter] → [MOD 5 counter] → $f_o = ?$

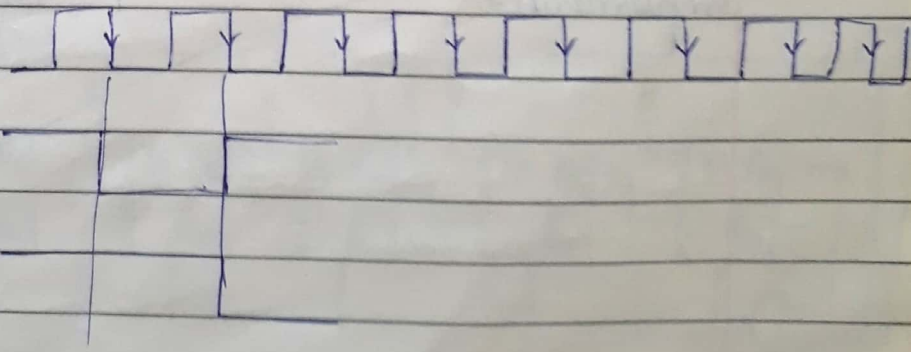
Ans- $f_o = \frac{1\ MHz}{10 \times 5} = 0.02\ MHz = 20\ KHz$

Q. Design MOD-8 down ripple counter.

Ans- $7 \to 6 \to 5 \to 4 \to 3 \to 2 \to 1 \to 0$

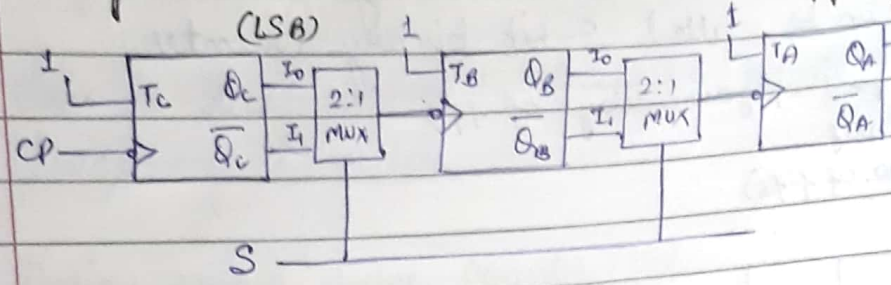
$\overline{Q_A}$	$\overline{Q_B}$	$\overline{Q_C}$
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0





CP | o/p | Type of ff
-ve | $\bar{Q}$ | D
-ve | Q | DOWN
+ve | $\bar{Q}$ | DOWN
+ve | Q | UP

Q. Design MOD-8 (3-bit) UP-DOWN Ripple Counter



If S = 0, it works as UP Counter
& S = 1, it works as DOWN Counter.

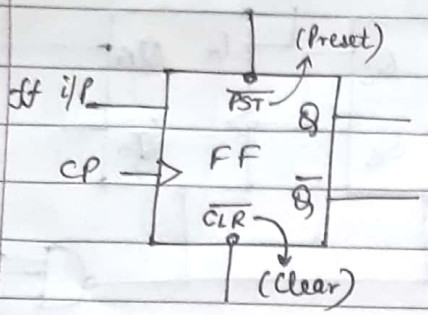
Difference b/w Asynchronous & Synchronous Counter

Asynchronous Counter	Synchronous Counter
1. External clock pulse is applied to 1st flip-flop (LSB) and o/p of this flip flop is connected to CP of next f-f & so on.	1. A common CP is applied simultaneously (parallely) to all the flip-flops.
2. Speed is low.	2. Speed is fast.
3. Easy to design.	3. Difficult to design.
4. Other names are Serial Counter, Ripple Counter, Frequency divider. Eg - MOD-8 Asynch. Counter.	4. Other name is parallel counter. Eg - Ring Counter, Register
5. A fixed pattern can be implemented.	5. Any pattern can be implemented.

Design of Non-Binary Asynchronous Counter ($N \neq 2^n$)

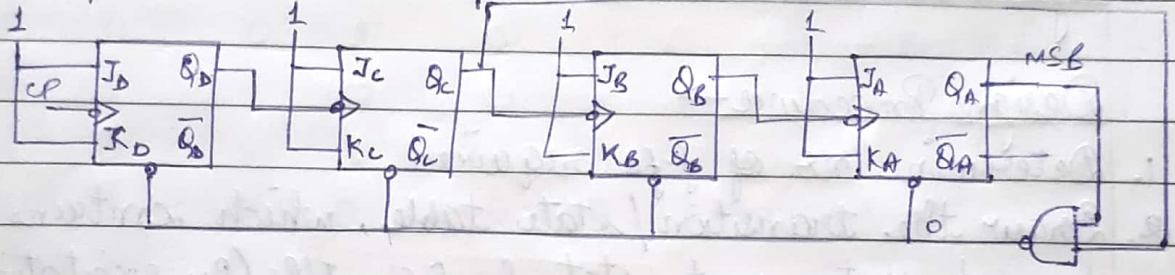
Q. Design MOD-10 (Decade/Decimal/BCD) Asynch. Counter

Ans- $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$



$\overline{PST}$	$\overline{CLR}$	Q_{n+1}
0	1	$1 \rightarrow PST$
1	0	$0 \rightarrow CLR$
1	1	FF works acc^n to I/Ps (Normally)
0	0	Invalid

We require 4 bits to rep 9



Q. Design MOD-N Asynch. Counter, If N is 9's Complement of your roll no., & if your roll no. is 2-digit no.

Ans- 99 $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$

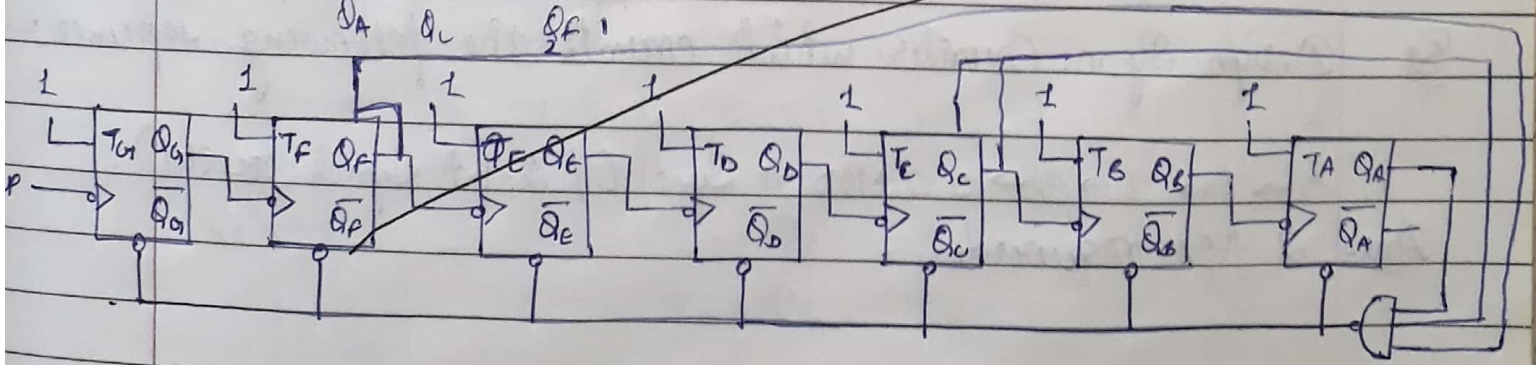
-16

83

We require 7-bits.

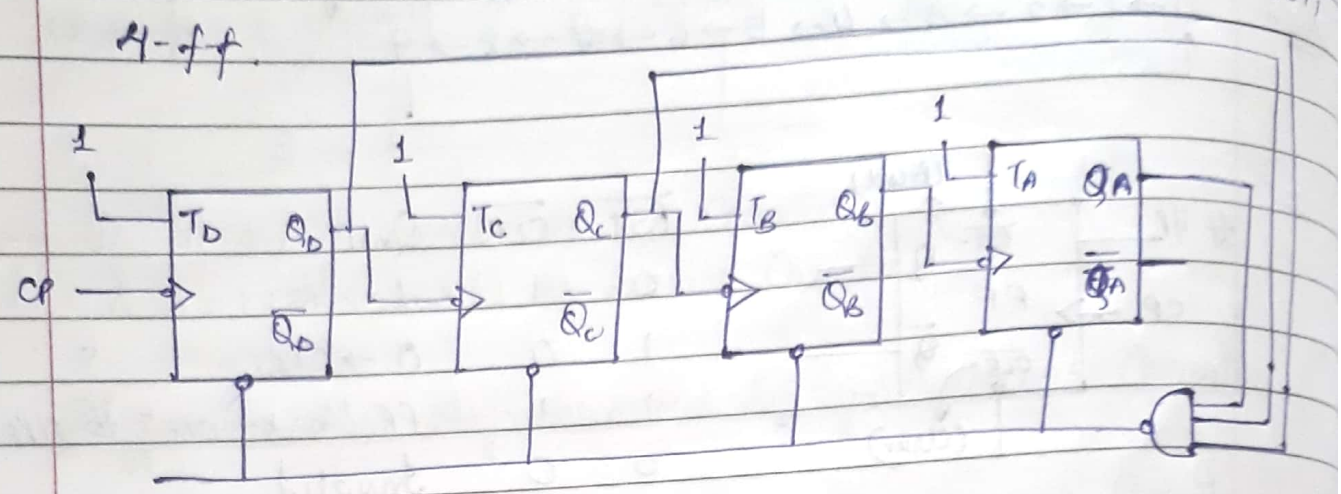
$82 \rightarrow 1\,0\,1\,0\,0\,1\,0$

$Q_A\ Q_C\ Q_F$



Q Design MOD-12 Asynch. Counter by T-flip-flop.

Ans- 0 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 0



Synchronous Counter

Design Procedure →

1. Determine no. of f-fs required.
2. Draw the transition / state table, which contains f-f's present state, next state & f-f's I/Ps (By excitation table)
3. Now, determine value of f-f I/Ps with the help of K-Map or Boolean algebra. (which is f^n of present state only).
4. Now, with the help of f-fs & suitable logic gates, circuit is implemented by applying a common clock pulse to all the f-fs.

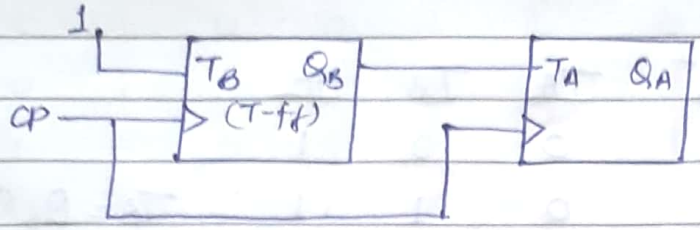
Q Design Sync. Counter which counts the following sequence:

1. 0 → 1 → 2 → 3 → 0 (MOD-4 Sync. C / 2-bit Synch. counter).

Ans- 2 f/fs required.

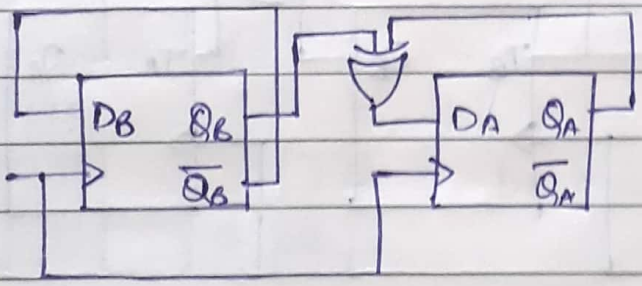
By T-ff →

Present State Q_A	Q_B	Next State Q_A^+	Q_B^+	$(Q_A \oplus Q_A^+)$ T_A	$(Q_B \oplus Q_B^+)$ T_B	
0	0	0	1 (0→1)	0	1	$T_A = Q_B$
0	1	1	0 (1→2)	1	1	$T_B = 1$
1	0	1	1 (2→3)	0	1	
1	1	0	0 (3→0)	1	1	



By D-ff →

PS Q_A	Q_B	NS Q_A^+	Q_B^+	$(=Q_A^+)$ D_A	$(=Q_B^+)$ D_B	
0	0	0	1	0	1	$D_A = Q_A \oplus Q_B$
0	1	1	0	1	0	$D_B = \overline{Q_B}$
1	0	1	1	1	1	
1	1	0	0	0	0	



By JK ff →

Exc. Table →

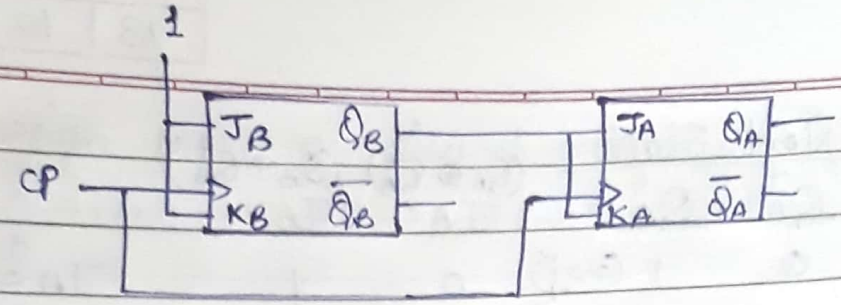
Q	Q^+	I	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

Q_A	Q_B	Q_A^+	Q_B^+	J_A	K_A	J_B	K_B
0	0	0	1	0	d	1	d
0	1	1	0	1	d	d	1
1	0	1	1	d	0	1	d
1	1	0	0	d	1	d	1

Q_A \ Q_B	0	1
0		1
1	d	d

$J_A = Q_B$ $J_B = 1$

$K_A = Q_B$ $K_B = 1$



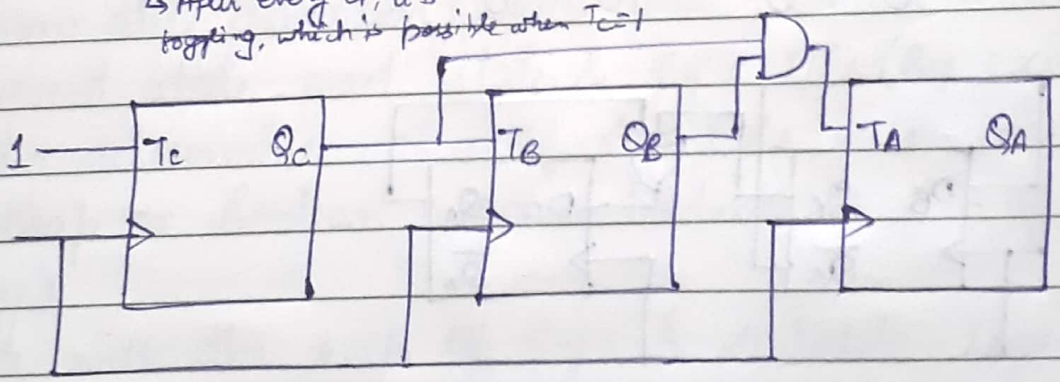
Q. Design MOD-8 Sync. Counter (3-bit Binary Synch. Counter) by T ff. 0→1→2→3→4→5→6→7

Ans-

PS			NS					
Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

$T_A = Q_B Q_C$
$T_B = Q_C$
$T_C = 1$

↳ After every cp, it's toggling, which is possible when $T_C = 1$



Similarly, for MOD-16, $T_D = 1$

$T_C = Q_D$

$T_B = Q_C Q_D$

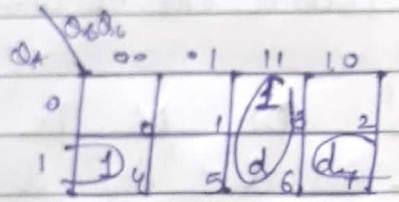
$T_A = Q_B Q_C Q_D$

→ $1 \oplus d = d$
→ $0 \oplus d = d$
→ $d \oplus d = d$

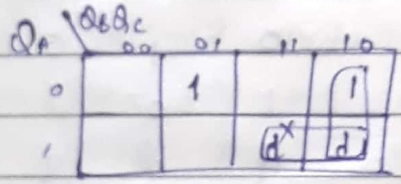
Q. Design MOD-6 Synch Counter.

Ans- $0 \to 1 \to 2 \to 3 \to 4 \to 5$.

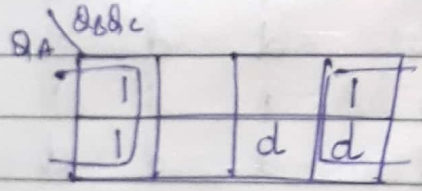
Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	D_A	D_B	D_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	0	0	0	0	0	0
1	1	0	d	d	d	d	d	d
1	1	1	d	d	d	d	d	d



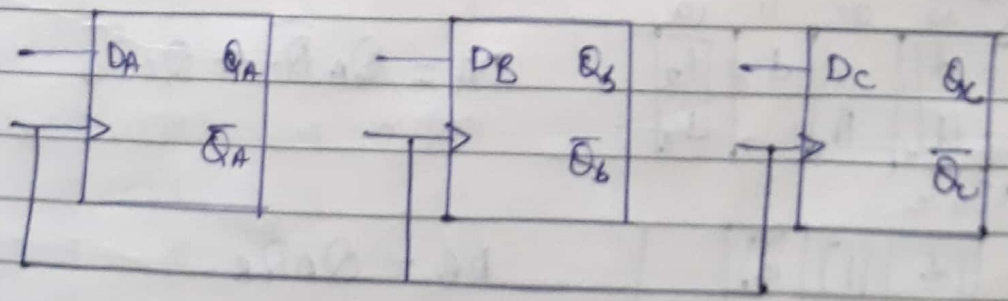
$$D_A = Q_B Q_C + Q_A \bar{Q}_C$$

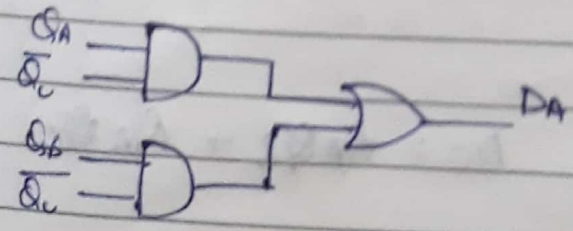


$$D_B = Q_B \bar{Q}_C + \cancel{Q_A Q_B} + \bar{Q}_A \bar{Q}_B Q_C$$



$$D_C = \bar{Q}_C$$





Q. Design MOD-6 Grey Synch. Counter.

Ans-

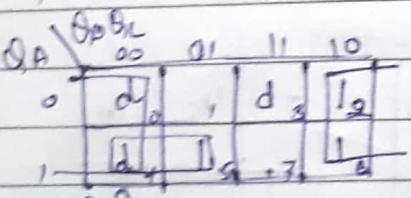
PS B_2	B_1	B_0	NS A_2	A_1	A_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	
0	0	0	0	0	1	(0→1)
0	0	1	0	1	1	(1→3)
0	1	0	0	1	1	(2→3)
0	1	1	0	1	0	(3→2)
1	0	0	d	d	d	(4→6)
1	0	1	d	d	d	(5→7)
1	1	0	1	1	1	(6→7)
1	1	1	0	0	0	(7→0)

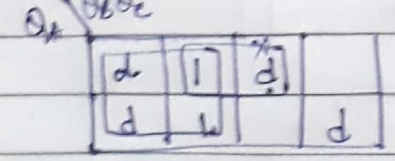
Q. Design Synch. Counter which counts the following seq:-
$1 \to 2 \to 5 \to 7 \to 1$ (By D-ff)

Ans-

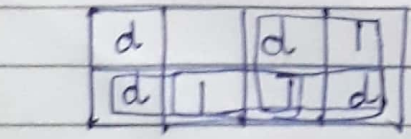
Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	D_A	D_B	D_C
0	0	0	d	d	d	d	d	d
0	0	1	0	1	0	0	1	0
0	1	0	1	0	1	1	0	1
0	1	1	d	d	d	d	d	d
1	0	0	d	d	d	d	d	d
1	0	1	1	1	1	1	1	1
1	1	0	d	d	d	d	d	d
1	1	1	0	0	1	0	0	1



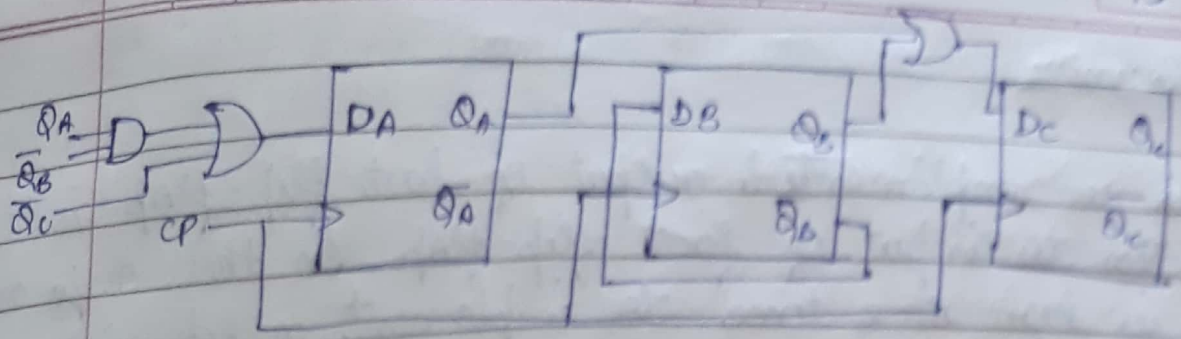
$D_A = Q_A \bar{Q}_C + Q_A \bar{Q}_B$



$D_B = Q_A \bar{Q}_B$



$D_C = Q_A Q_B + Q_A \bar{Q}_C$



Lock-Out Condition

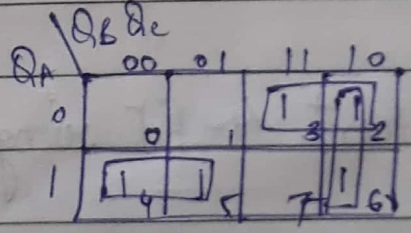
When counter goes from one used state to any unused state and again, it goes to another unused state. From that unused state, counter will never arrive in used state and this condition is called LOCK-OUT condition. To avoid lock-out condition, we write down used states in front of unused states.

Q. Design Synd. Counter which counts following sequence & avoid lock-out cond^n :-

1 → 2 → 5 → 7 → 1 ← Used States

Ans 6 3 4 6 ← Unused States

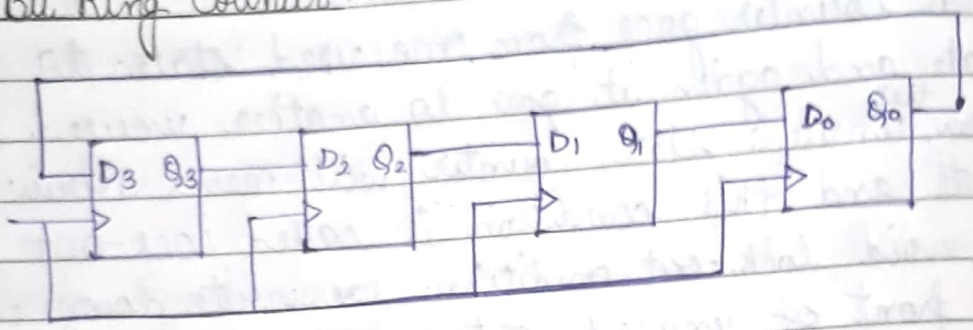
Q_A	Q_B	Q_C	$Q_A^+ = DA$	$Q_B^+ = DB$	$Q_C^+ = DC$
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	0	0	1



Ring Counter

In this, uncomplimented output of last flip-flop is connected to input of first flip-flop. In this, only one input is high at a time. To implement n-bit ring counter, we require N-flip-flops.

4-Bit Ring Counter →

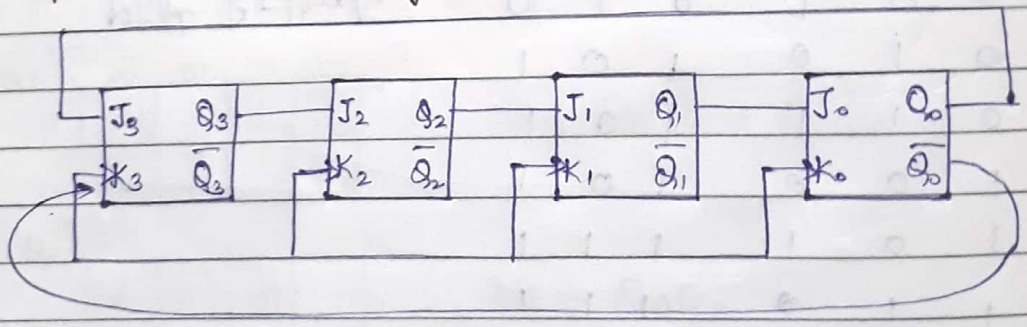


CP	Q_3	Q_2	Q_1	Q_0
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0

n Clock pulse required.
Total no. of used states = n
Unused states = $2^n - n$
$f_0 = \frac{f_i}{\text{No. of used states}}$

Q. Implement 4-bit ring counter by using J-K flip-flop.

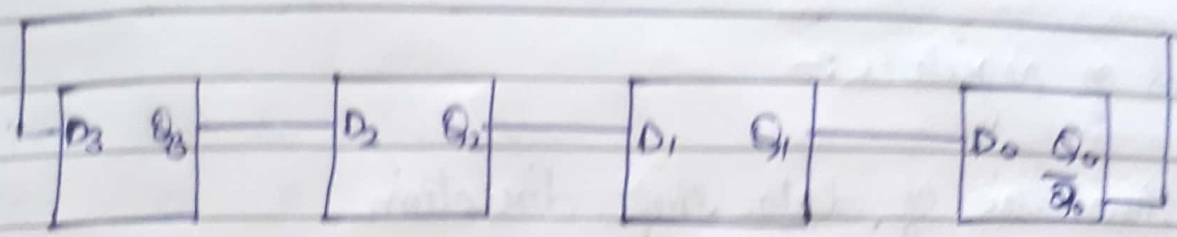
Ans -



Page No: 161
20 | 10 | 18

Twisted Ring Counter / Johnson / Creeping / Walking / Mobius / Switch Tail Counter

In this, complemented output of last flip-flop is connected to input of first flip-flop



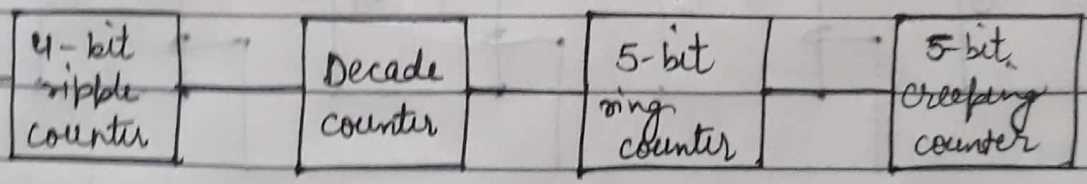
CP	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

Used $= 2^n$

Unused $= 2^n - 2n$

$f_0 = \frac{f_i}{2n}$

What will be the o/p frequency in following circuit :-



Register

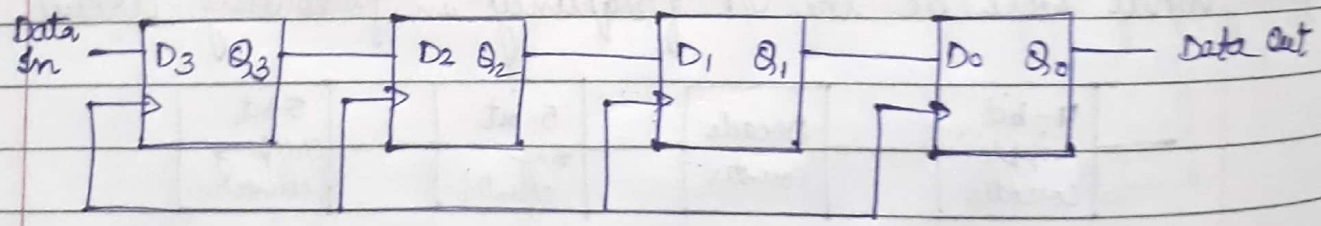
It is a group of flip-flops which is used to store temporary data.
To store n-bit data, we require N-flip flops.

Types of Registers :-

a) On the basis of data shift direction
(i) Right shift register
(ii) Left shift register

b) On the basis of data applied at I/P & O/P
(i) SISO - Serial In Serial Out
(ii) SIPO - Serial In Parallel Out
(iii) PISO - Parallel In Serial Out
(iv) PIPO - Parallel In Parallel Out

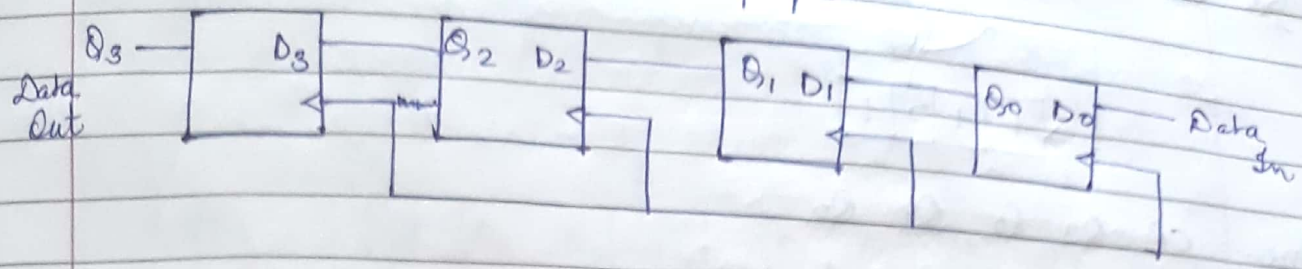
SISO - Right shift register
In this, we apply input serially & we also take output serially.



CP	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	0	1	0
4	1	1	0	1

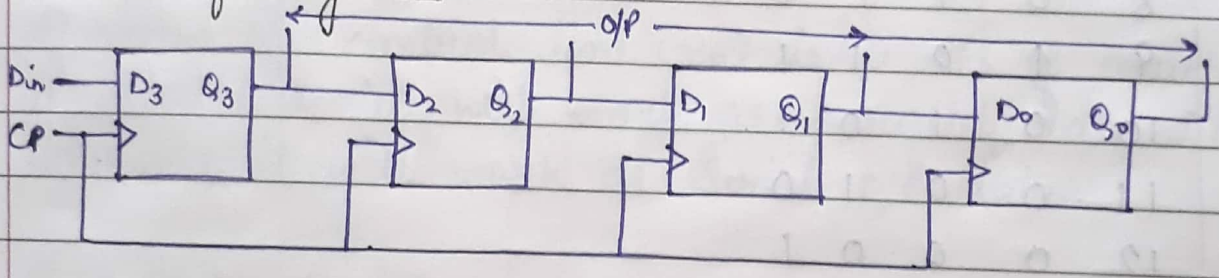
Delay = $n * T_{clock}$

where, n = No. of flip-flops

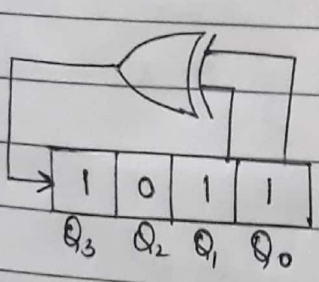


CP	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	0
4	1	1	0	1

SIPO Shift Register



In this, I/P data is given serially but O/P data is taken parallely (at same time).

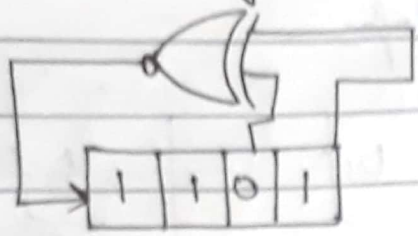
*Q. 

In the following ckt, if initial value of Q_3, Q_2, Q_1, Q_0 are 1011, what will be value after 5 CP?

Ans-

CP	Q_3	Q_2	Q_1	Q_0
0	1	0	1	1
1	0	1	0	1
2	1	0	1	0
3	1	1	0	1
4	1	1	1	0
5	1	1	1	1

Q. After how many cp data will be same?

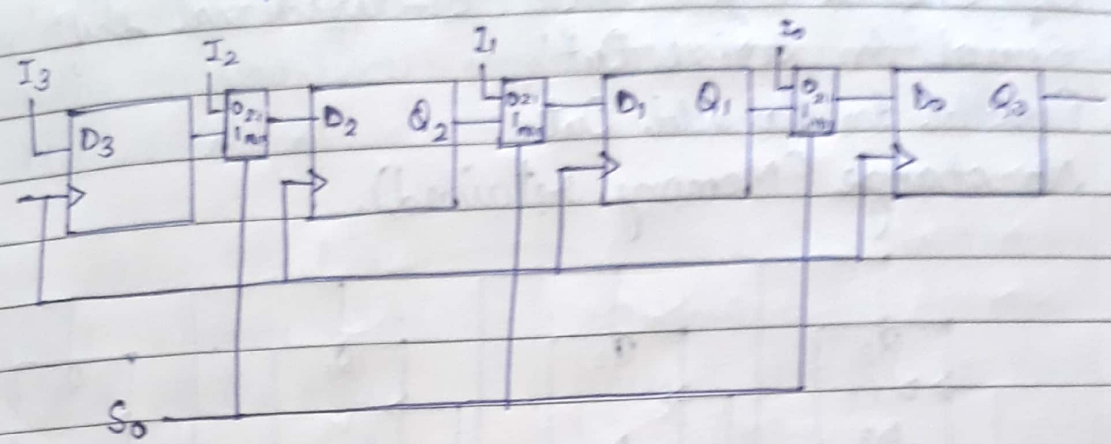


Ans-

cp	Q_3	Q_2	Q_1	Q_0
0	1	1	0	1
1	0	1	1	0
2	1	0	1	1
3	0	1	0	1
4	0	0	1	0
5	0	0	0	1
6	1	0	0	0
7	1	1	0	0
8	0	1	1	0
9	1	0	1	1
10	0	1	0	1
11	0	0	1	0
12	0	0	0	1
13	1	0	0	0
14	1	1	0	0
15	0	1	1	0

PISO Register

In this, data is given parallely but o/p data is taken serially.



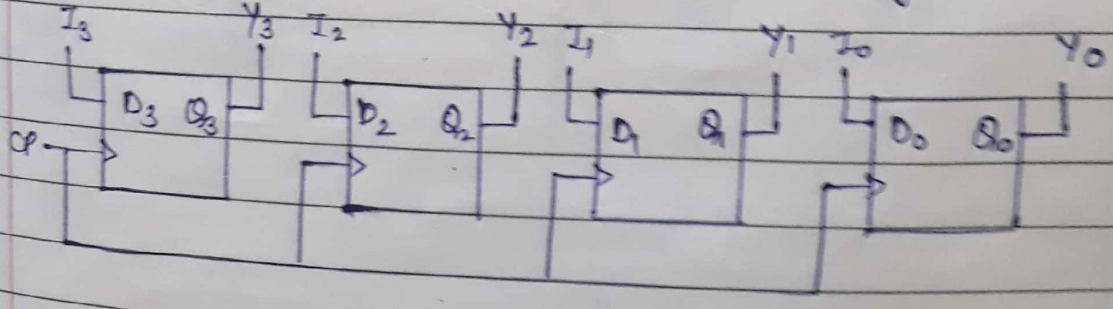
In Serial,

$D_2 = Q_3$, $D_1 = Q_2$, $D_0 = Q_1$.

If value of control line (S_0) is 0, I/p of multiplexer is selected & it will work as parallel data I/p. Otherwise it will work as serial o/p data.

PIPO Shift Register

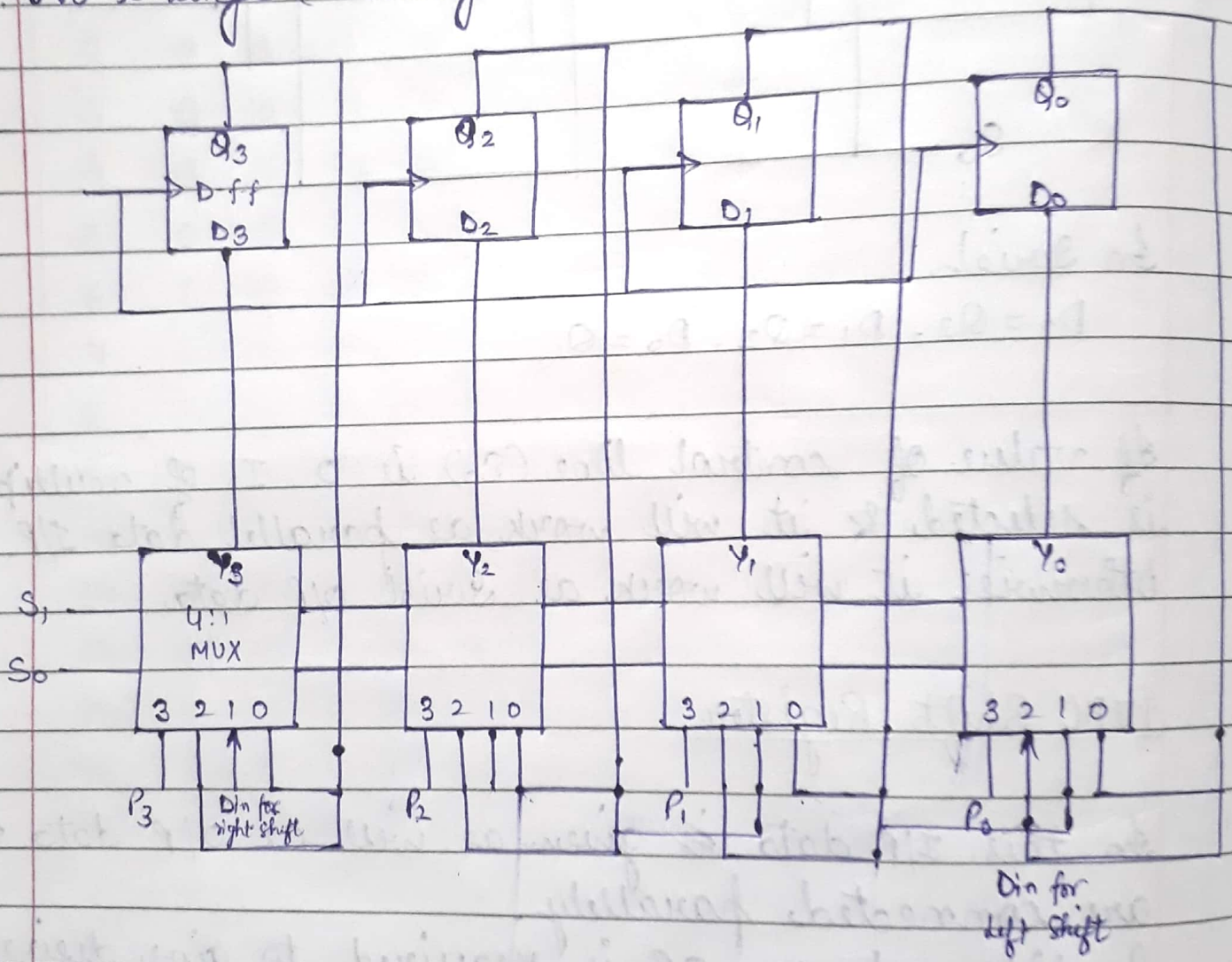
In this, I/p data ~~is~~ given as well as o/p data taken are connected parallely.
In this only one CP is required to give desired o/p.



Universal Shift Register

It performs the following operations :-

1. Left Shift & Right Shift (Bi-directional shift register)
2. Serial In Serial Out
3. Parallel In Parallel Out
4. No change (Memory retained).



UNIT-5

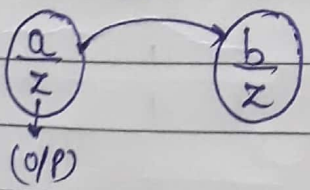
Synchronous Sequential Clocked Circuit Design

This can be designed or analysed by using two models (or machines) and these are Melay Machine & Moore Machine.

Moore Machine

1. Output depends upon present states only.
   $z = f(Q)$
2. If input changes, output does not change.
3. More number of states are required.
4. More hardware required.

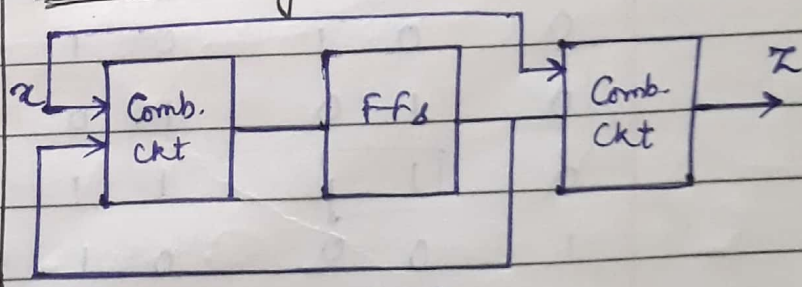
Block Diagram →

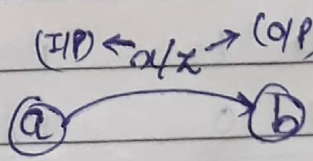
5. 

Melay Machine

1. Output depends upon present states as well as input (x).
   $z = f(Q, x)$
2. If input changes, output also changes.
3. Less number of states are required.
4. Less hardware required.

Block Diagram →

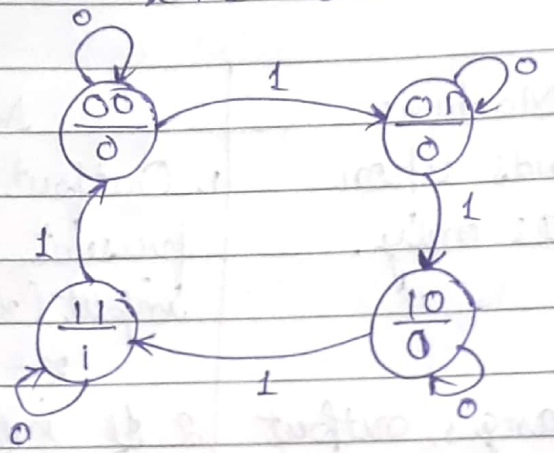


5. 

Q-1 In a sequential clocked circuit, there are two T-f/f whose o/ps are A & B, and there is one I/P (control line) x. If $x=1$, it works as up-counter and if $x=0$, it remains in same state & when its value is max^m. (3,7) o/p is high, otherwise low. a) Draw the state table & state diagram. b) Draw the logical diagram.

Ans- a → 0 0
b → 0 1
c → 1 0
d → 1 1

$x \to 1$: 00 → 01 → 10 → 11 → 00



State Diagram

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

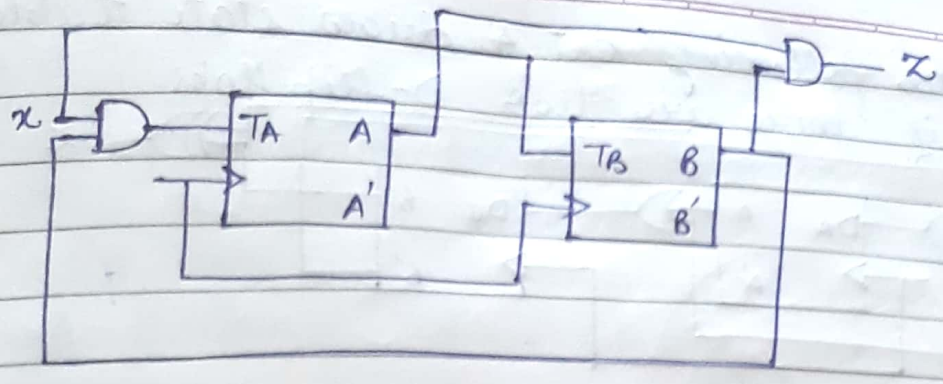
I/P	P.S		N.S		O/P	T_A	T_B
x	A	B	A^+	B^+	Z	$(A \oplus A^+)$	$(B \oplus B^+)$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	0	0
1	0	0	0	1	0	0	1
1	0	1	1	0	0	1	1
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	1

$Z = \Sigma m(3,7) = A.B$

$T_A = \Sigma m(5,7) = xB$

$T_B = x$

∵ Z does not depend on x, it's an example of Moore M/C.



Q.2 In the above sequential synchronous circuit, analyse the circuit & determine whether this is example of Mealy M/C or Moore M/C and draw the state table & state diagram.

Ans- $Z = A \cdot B$

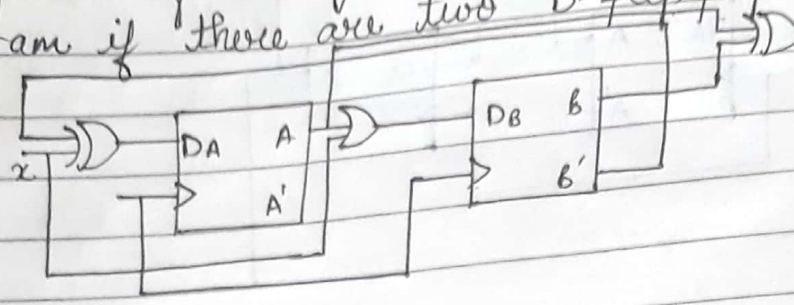
$TA = xB$

$TB = x$

∵ Z does not depend upon x, so it's example of Moore M/C.

I/P	P.S.		(xB)		TA⊕A N.S		(AB)
			‖		↑		‖
x	A	B	TA	TB	A	B	Z
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	0	0	1	1	1
1	0	0	0	1	0	1	0
1	0	1	1	1	1	0	0
1	1	0	0	1	1	1	0
1	1	1	1	1	0	0	1

Q-3 Analyse the following ckt & draw state table & state diagram if there are two D-flip-flops.

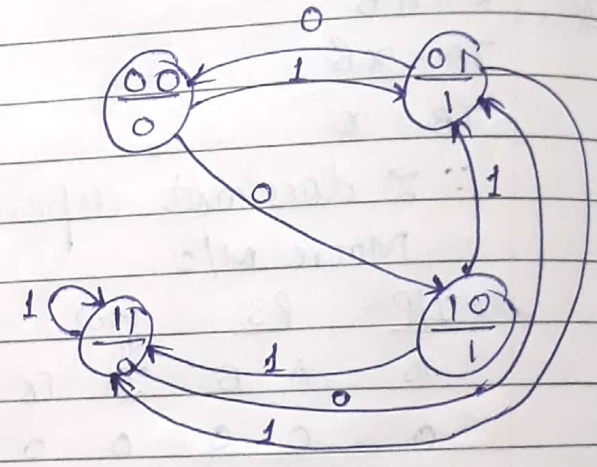


Ans- $D_A = x \oplus B' = x \odot B$

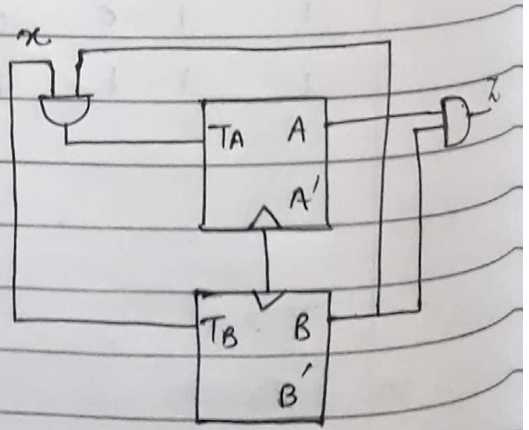
$D_B = A + x$

$z = A \oplus B$ ⇒ (Moore M/C)

I/P	P.S		N.S		O/P
x	A	B	A^+	B^+	z
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	1	1	1
0	1	1	0	1	0
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	0	1	1
1	1	1	1	1	0



Q-4 Analyse the following logical diagram of synchronous sequential circuit and draw the state table, state diagram and ASM Chart.
(Algorithmic State Machine)



Ans- ∵ Clock pulse is simultaneous,
∴ it's a synch. seq. ckt.
It's an example of Moore M/c

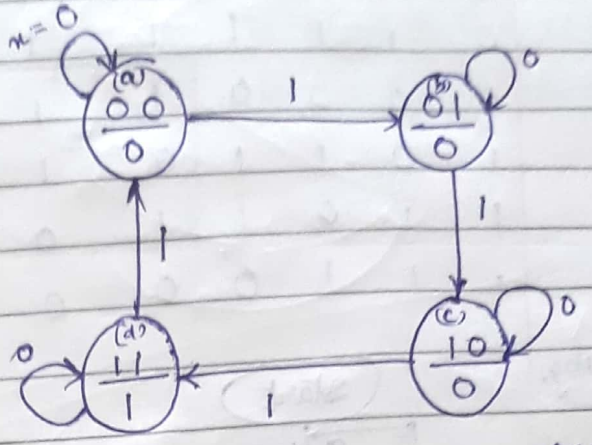
$T_A = Bx$

$T_B = x$

$Z = A \cdot B$ (Moore M/C – FSM)

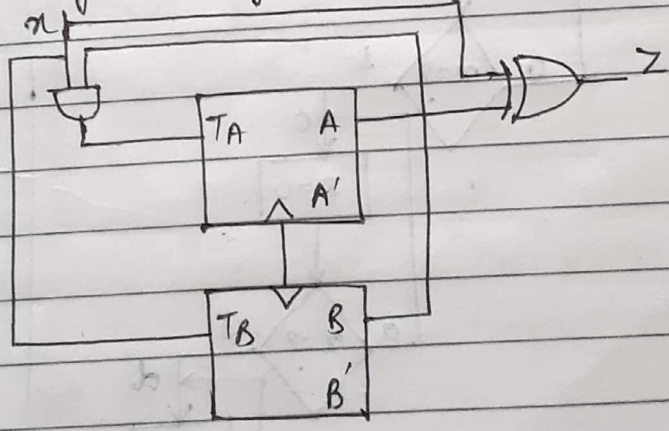
a → 00
b → 01
c → 10
d → 11

I/P	P.S	N.S			
x	A	B	A^+ (T_A = Bx)	B^+ (T_B = x)	Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	0	0	1



ASM Diag.

Analyse the following clocked synch. seq. ckt & determine whether this is example of Mealy M/c or Moore M/c. Draw the state diagram by drawing state table & ASM chart



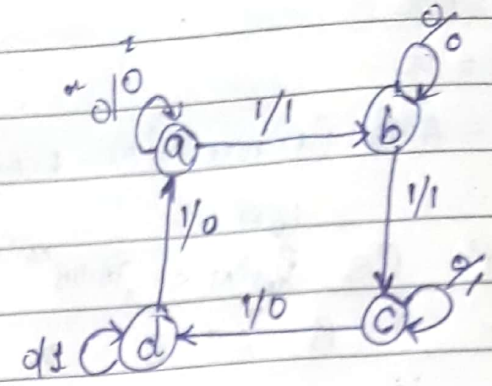
$T_A = Bx$

$T_B = x$

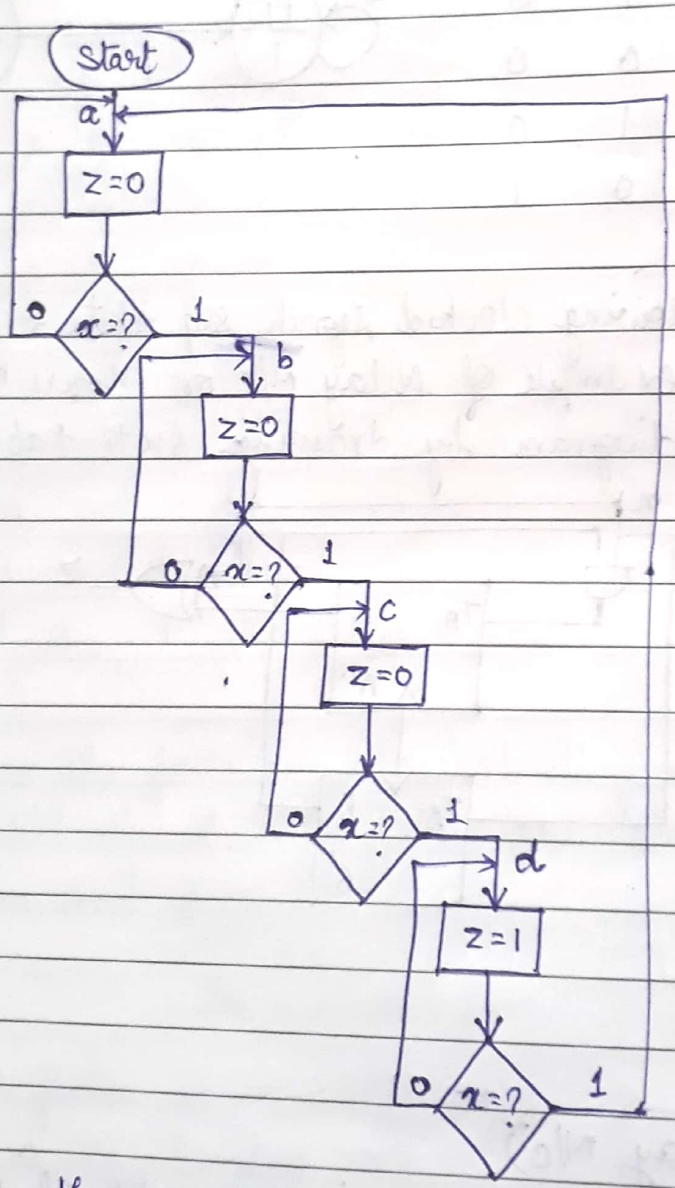
$Z = A \odot x$ (Mealy M/c)

∵ Z depends upon x, so it's example of Mealy M/c.

I/P	P.S		N.S		
x	A	B	A^+	B^+	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	0	0	0



ASM Diag.



ASM is the symbolic representation of a FSM.

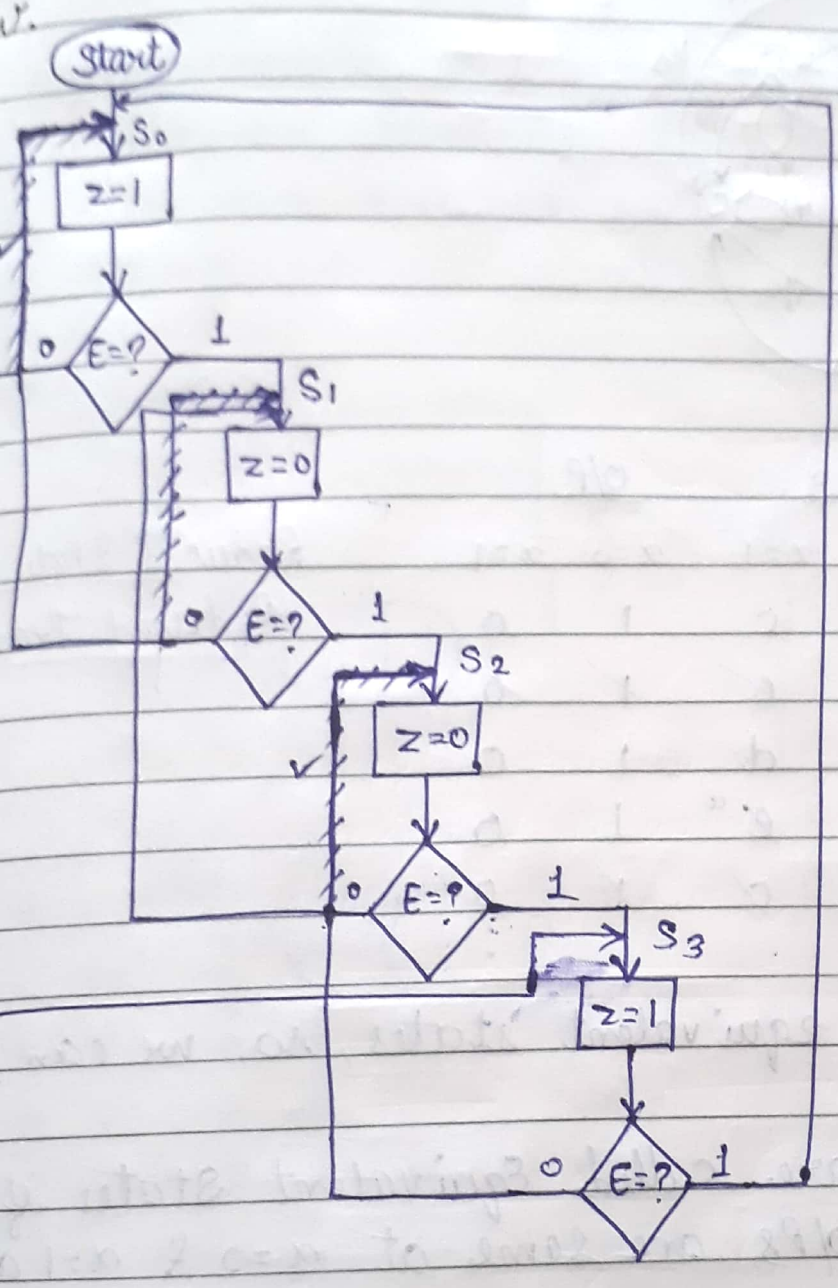
⬭ → Start

▭ → Statement

◇ → Decision

Q1 Draw the ASM chart if a logical ckt has one enable line E. If its value is 1, it works as 2-bit up-counter & if its value is 0, it works as down-counter and whenever it reaches min^m or max^m value, o/p becomes high, otherwise low.

Ans-



E→1 → $S_0 \to S_1 \to S_2 \to S_3 \to S$

E→0 → $S_0 \to S_3 \to S_2 \to S_1 \to S_0$

Q2

(a) (b)

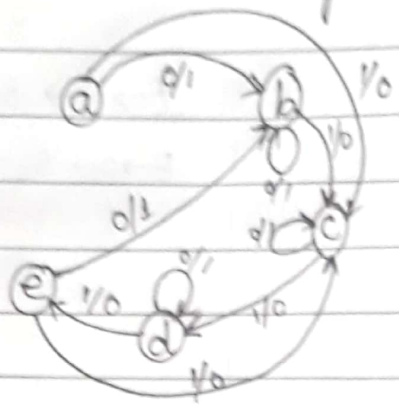
(c)

(e)

(d)

State Reduction Table

Q. From given state diagram, draw state reduction table & reduced state diagram.



Ans- (i)

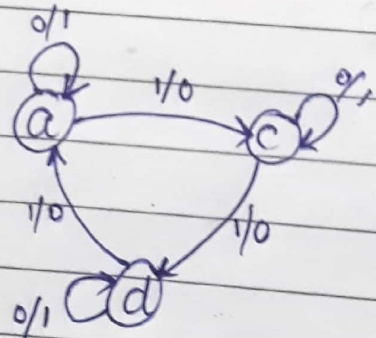
P.S	N.S		O/P	
	x=0	x=1	x=0	x=1
a	b[a]	c	1	0
b	b	e	1	0
c	c	d	1	0
d	d	e[a]	1	0
e	b	c	1	0

Reduced State Table/ Reduced Transition Table

a, b & e are equivalent states, so, we can eliminate them.

→ Two states are called Equivalent States if next states and o/ps are same at x=0 & x=1 and we can eliminate one state.

(ii) Again, a & e are equivalent, so eliminate e & replace it by a.



State Diagram

Asynchronous Sequential Circuit Design

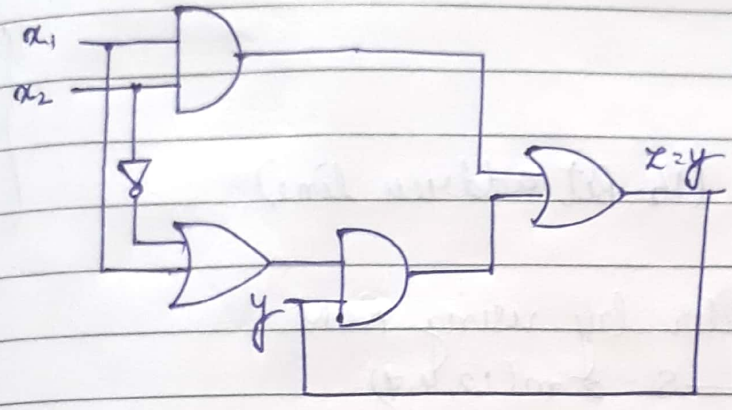
In asynchronous sequential circuit, there is no clock pulse like in synchronous sequential clocked circuit.

Q. An asynchronous seq. ckt is described by following eq^ns. Draw logical diagram, transition table & output map.

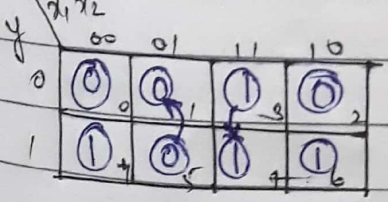
$$Y = x_1 x_2 + (x_1 + \bar{x}_2)\, y$$

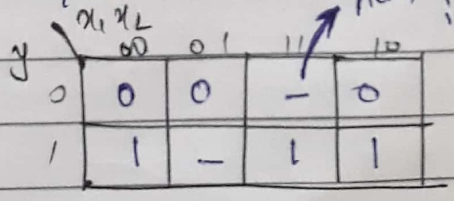
$$Z = y$$

Ans-



P.S			(By eqn) N.S			Stable	O/P (Z)
y	x_1	x_2	y	x_1	x_2		
0	0	0	0	0	0	Yes	0
0	0	1	0	0	1	Yes	0
0	1	0	0	1	0	Yes	0
0	1	1	1	1	1	No	1
1	0	0	1	0	0	Yes	1
1	0	1	0	0	1	~~Yes~~ No	0
1	1	0	1	1	0	Yes	1
1	1	1	1	1	1	Yes	1

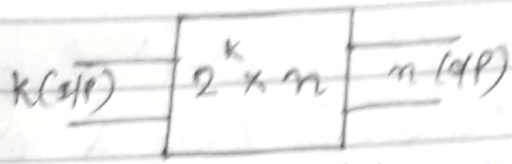




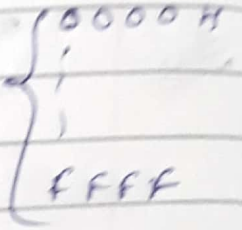
Here, transition is taking place.

Read Only Memory (ROM)

It is used to store permanent data & it has fixed AND array followed by programmable OR array.



K = No. of address lines (Input)
n = Size of word (Output).



⇒ 64K x 8 ROM

$2^6 \times 2^{10} = 2^A$

A = 16 Bit (16-Bit address line).

Q. Design full adder by using ROM.

Ans-