New L J Institute of Engineering and Technology, Bodakdev
Branch: CSE (AIML)/ IT

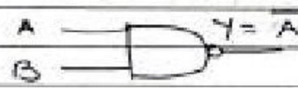
Semester: III (D to D)
Subject: Digital Fundamentals (3130704)

1 **Explain how NAND and NOR gates can be utilized as universal gates to implement all the basic gates <u>OR</u> Explain NAND and NOR as universal gates.**

→ NAND and NOR gates are universal Because, we can derive every other Gate from these two only (NAND and NOR separately)

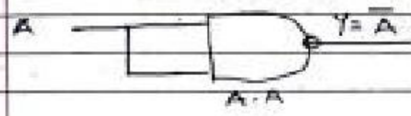
★ Universality of NAND gates:

NAND



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

(1) NOT gate using NAND: Inverter

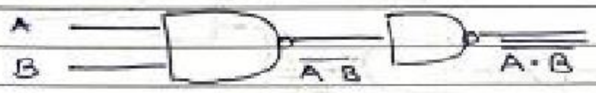


$\boxed{A \cdot A = A}$

$Y = \overline{A}$

(2) AND gate using NAND:

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

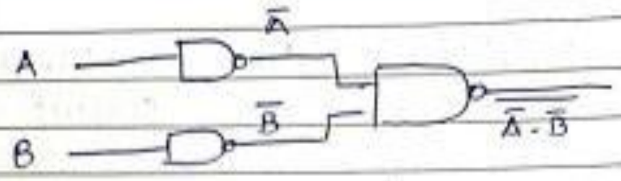


(3) OR gate using NAND:

OR → $A + B$

use De-Morgan's theorem.

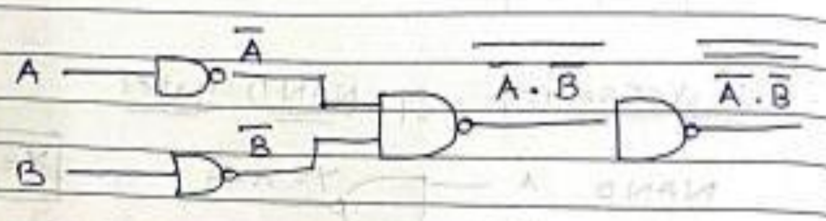
$A + B = \overline{\overline{A + B}} = \overline{\overline{A} \cdot \overline{B}}$



(4) NOR-Gate using NAND:

NOR $\rightarrow \overline{A+B}$

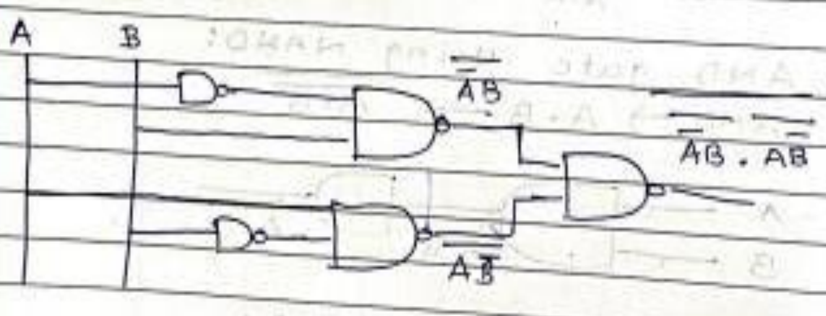
$= \overline{\overline{\overline{A+B}}} = \overline{\bar{A}\cdot\bar{B}} = \overline{\overline{\overline{\bar{A}\cdot\bar{B}}}}$



(5) Ex-OR using NAND:

Ex-OR: $A \oplus B \rightarrow \bar{A}B + A\bar{B} = \overline{\overline{\bar{A}B + A\bar{B}}}$

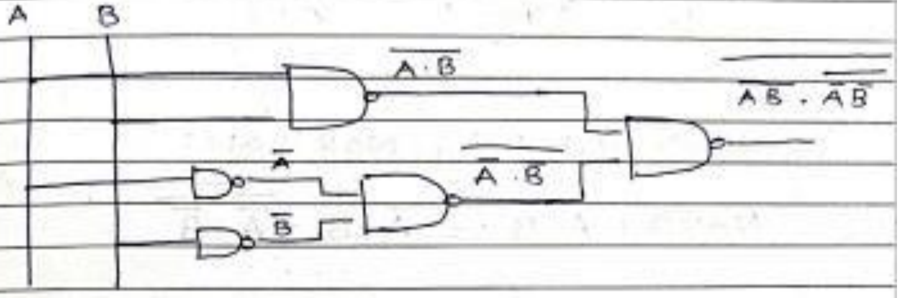
$= \overline{\overline{\bar{A}B}\cdot\overline{A\bar{B}}}$



* Ex-NOR Using NAND:

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

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



* Universality of NOR Gate:

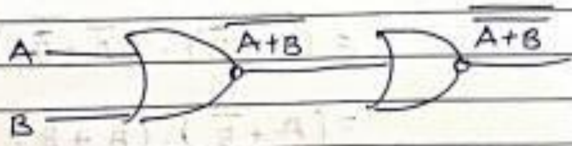
NOR: $Y = \overline{A+B}$

(1) NOT Gate using NOR:

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

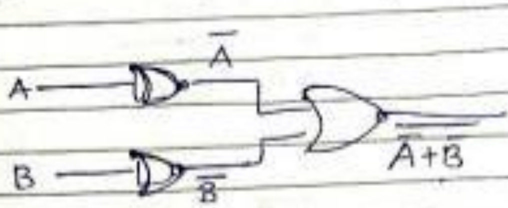
(2) OR Gate using NOR:

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



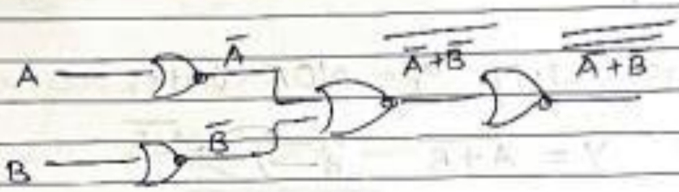
③ AND gate using NOR gate:

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



④ NAND gate Using NOR gate:

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



⑤ Ex-OR using NOR gate:

$$Ex\text{-}OR \rightarrow \bar{A}B + A\bar{B} = \overline{\overline{\bar{A}B + A\bar{B}}}$$

$$= \cancel{\overline{\bar{A}B}\,\overline{A\bar{B}}}$$

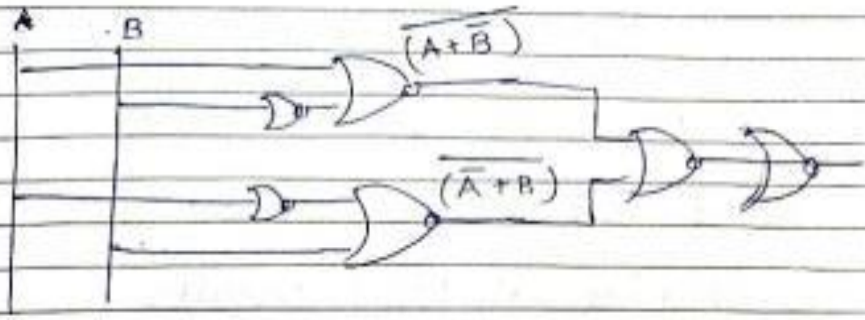
$$= \overline{(\overline{\bar{A}B}) \cdot (\overline{A\bar{B}})}$$

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

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

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

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



(6) Ex-NOR Using NOR

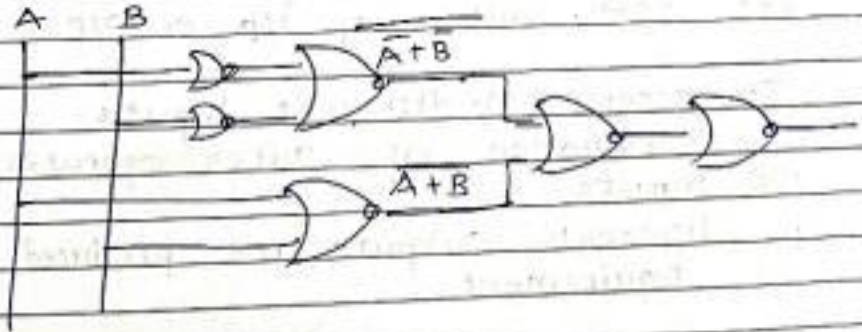
Ex-NOR =

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

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

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

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

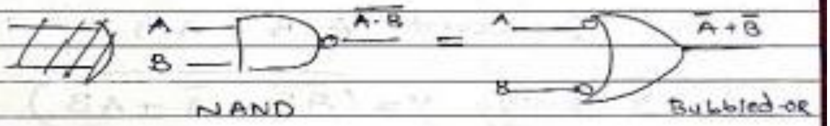


2 State and Prove D'Morgan Theorem using truth-tables.

* De-Morgan's theorems:

Theorem 1: $\overline{AB} = \bar{A} + \bar{B}$

The complement of product is equal to sum of the complement



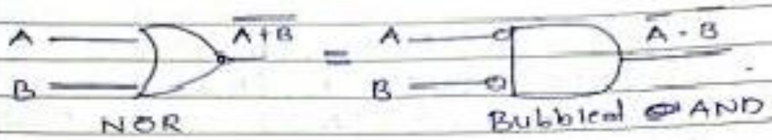
Proof:

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

Hence prooved, $\overline{A \cdot B} = \bar{A} + \bar{B}$

(2) Theorem 2: $\overline{A + B} = \bar{A} \cdot \bar{B}$

The complement of sum is equal to the product of the complement



proof:

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

Hence, prooved $\overline{A + B} = \bar{A} \cdot \bar{B}$

3 Convert the following numbers:

- **(i)** $(52)_{10} = (\ )_2$
- **(ii)** $(436)_8 = (\ )_{16}$
- **(iii)** $(5C7)_{16} = (\ )_{10}$
- **(iv)** $(11011.101)_2 = (\ )_{10}$
- **(v)** $(11111)_2 = (\ ?\ )_{Gray}$

*

(i) $(52)_{10} = (\ )_2$

2	52	
2	26	0
2	13	0
2	6	1
2	3	0
	1	1

$(52)_{10} = (110100)_2$

(ii) $(436)_8 = (\ )_{16}$

$\underbrace{100}_{4}\ \underbrace{011}_{3}\ \underbrace{110}_{6} = (100011110)_2$

$(100011110)_2 = (11E)$

(iii) $(5C7)_{16} = (\ )_{10}$

5 C 7

$5\times16^2 \quad C\times16^1 \quad 7\times16^0$

$= 5\times16^2 + 12\times16 + 7\times1$

$= 1280 + 192 + 7$

$= 1479$

$(5C7)_{16} = (1479)_{10}$

A → 10
B → 11
C → 12
D → 13
E → 14
F → 15

→ $(11011.101)_2$

$$\begin{array}{ccccccccc} 1 & 1 & 0 & 1 & 1 & . & 1 & 0 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\ 1\times 2^4 & 1\times 2^3 & 0\times 2^2 & 1\times 2^1 & 1\times 2^0 & & 1\times 2^{-1} & 0\times 2^{-2} & 1\times 2^{-3} \end{array}$$

$16 + 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125$

$= (27.625)_{10}$

→ $(11111)_2 = (?)_{gray}$

$$\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \\ \downarrow & & & & \\ \hline 1 & 0 & 0 & 0 & 0 \end{array}$$

$(11111)_2 = (10000)_{gray}$

4 Construct Hamming code for BCD 0110. Use even parity.

→

* No of parity bits

$2^P \geq x + P + 1$ $\quad x = 4$

No of given bits (0110)

so, $P = 3$

$8 \geq 8$

so, $P = 3$ satisfies the equation.

* Total code bits = 4 + 3 = 7

	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Data bits	0	1	1		0		
Parity bits				0		1	1

* Now determine parity bits. (total no of 1's must be in even parity)

for $P_1 \Rightarrow$ D_3 D_5 D_7

0 1 0 $\qquad$ 1

for $P_2 \Rightarrow$ D_3 D_6 D_7

0 1 0 $\qquad$ 1

for $P_4 \Rightarrow$ D_5 D_6 D_7

1 1 0 $\qquad$ 0

Ans $\Rightarrow$ (0110011)

5 Convert 1000 0110 (BCD) to decimal, binary & octal.

Convert 1000 0110 (BCD) to decimal, bin
and octal.

$\underbrace{1000}\ \underbrace{0110}$ (BCD) ⇒ decimal is 86.

Conversion of $(86)_{10}$ to Binary

2	86	
2	43	0
2	21	1
2	10	1
2	5	0
2	2	1
	1	0

$(\cancel{1010110})_2$

$(1010110)_2$

$(1010110)_2 = (?)_8$

$\underbrace{001}_{1}\ \underbrace{010}_{2}\ \underbrace{110}_{6} = (126)_8$

6 Convert F(A, B, C) = BC + A into standard minterm form.

6.

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

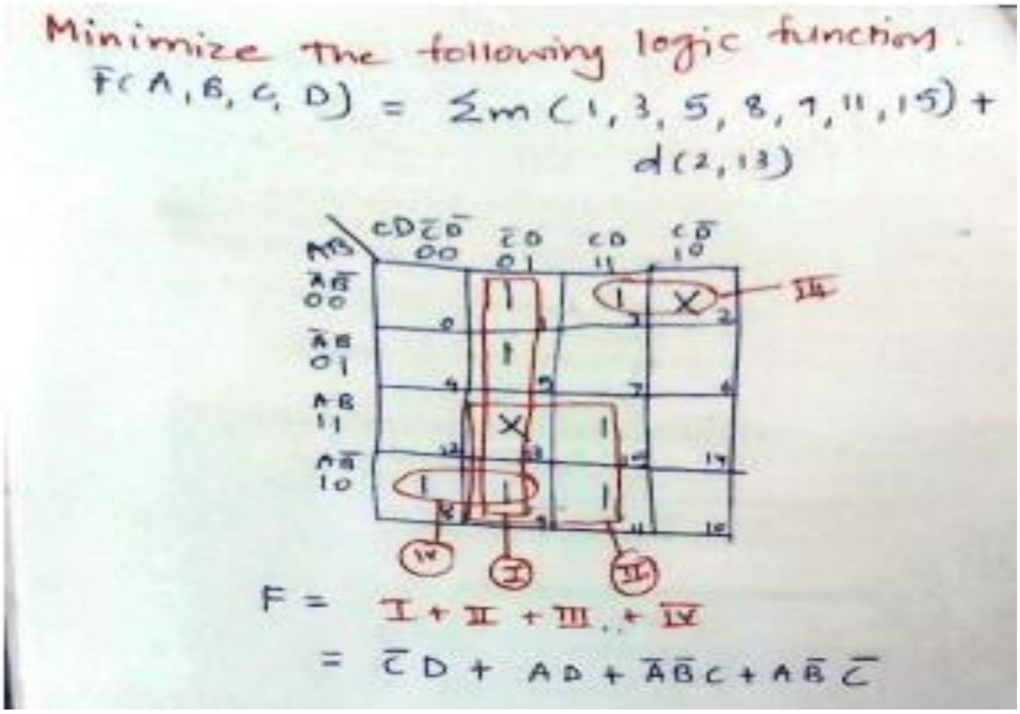
$$= (A+\bar{A})\,BC + A(B+\bar{B})(C+\bar{C})$$

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

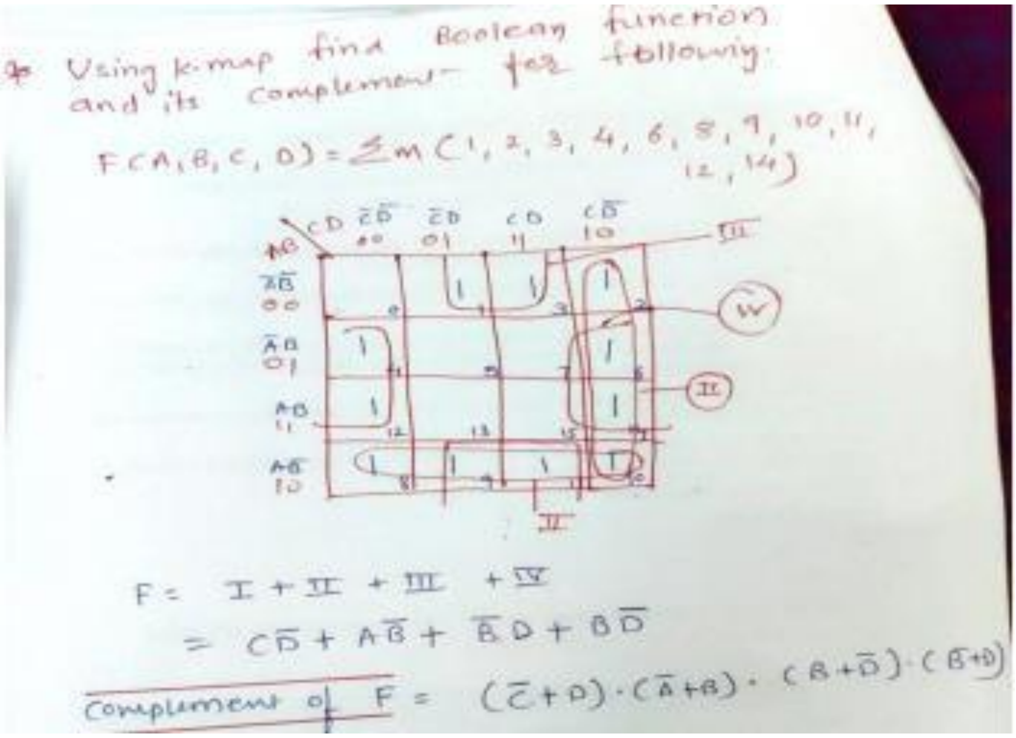
$$= ABC + \bar{A}BC + ABC + A\bar{B}\bar{C} + AB\bar{C} + A\bar{B}C$$

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

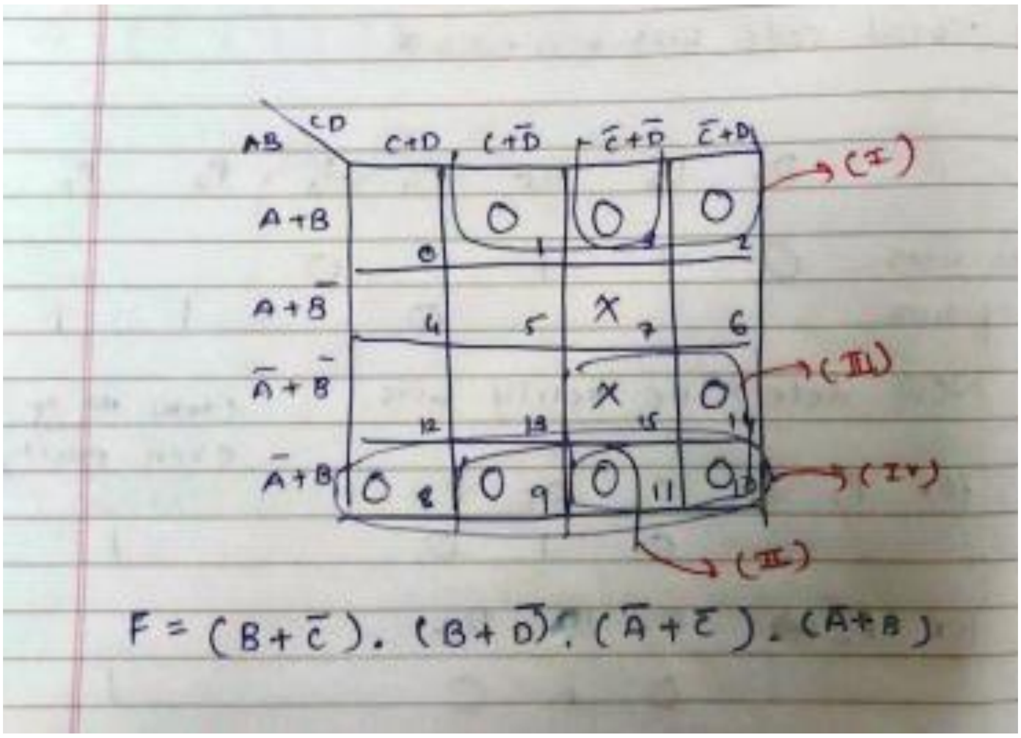
7 **Minimize the following logic function using K-map:**
F(A,B,C,D) = Σ m(1, 3, 5, 8, 9, 11, 15) + d(2,13)



8 **Using K-map find the Boolean function and its complement for the following: F(A,B,C,D) = ∑(1,2,3,4,6,8,9,10,11,12,14)**



9 **Minimize the logic function**
F (A,B,C,D) = π M (1, 2, 3, 8, 9, 10, 11, 14) . d (7, 15) Use Karnaugh map.



10 Simplify following Boolean function by using the tabulation method
F = Σ(0,1,3,7,8,9,11,15).

$f(A,B,C,D) = \sum m(0,1,3,7,8,9,11,15)$

	A	B	C	D	No of 1's
0	0	0	0	0	0
1	0	0	0	1	1
3	0	0	1	1	2
7	0	1	1	1	3
8	1	0	0	0	1
9	1	0	0	1	2
11	1	0	1	1	3
15	1	1	1	1	4

Step 1: Make group by counting No of 1's

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

No PI in this table.

Step 2: Compare adjacent groups and find 1 bit change.

→ Mark changed bit as –.
→ Mark the minterm which is grouped with another minterm.

group	Pair of m_i	A	B	C	D	
0	$m_0 - m_1$	0	0	0	—	✓
	$m_0 - m_8$	—	0	0	0	✓
1	$m_1 - m_3$	0	0	—	1	✓
	$m_1 - m_9$	—	0	0	1	✓
	$m_8 - m_9$	1	0	0	—	✓
2	$m_3 - m_7$	0	—	1	1	✓
	$m_3 - m_{11}$	—	0	1	1	✓
	$m_9 - m_{11}$	1	0	—	1	✓
3	$m_7 - m_{15}$	—	1	1	1	✓
	$m_{11} - m_{15}$	1	—	1	1	✓

NO PI in the table

group	quad of m_i	A	B	C	D	
0	$m_0 - m_1 - m_8 - m_9$	—	0	0	—	PI
	$m_0 - m_8 - m_1 - m_9$	—	0	0	—	
1	$m_1 - m_3 - m_9 - m_{11}$	—	0	—	1	PI
	$m_1 - m_9 - m_3 - m_{11}$	—	0	—	1	
2	$m_3 - m_7 - m_{11} - m_{15}$	—	—	1	1	PI
	$m_3 - m_{11} - m_7 - m_{15}$	—	—	1	1	

Step 3: make PI table

EPI

	group	0	1	3	7	8	9	11	15
✓ $\bar{B}\bar{C}$	0,1,8,9	x	x			x	x		
$\bar{B}D$	1,3,9,11		x	x			x	x	
✓ CD	3,7,11,15			x	x			x	x

step 4: Consider columnwise → Mark the Single x column

write PI of ⊗ mark

$\boxed{0, 7, 8, 15}$

→ Check all the minterms are covered. If Not, find PI that includes remaining terms.

Ans $y = \bar{B}\bar{C} + CD$ [covers Every minterm]

→ ✓̊ marked terms are EPI

11 What is meant by decoder? Explain 3-to-8 line decoder with diagram and truth table.

* Decoder:-

↳ Decoder takes n inputs & generates upto 2^n outputs

↳ Decoder does reverse operation of Encoder

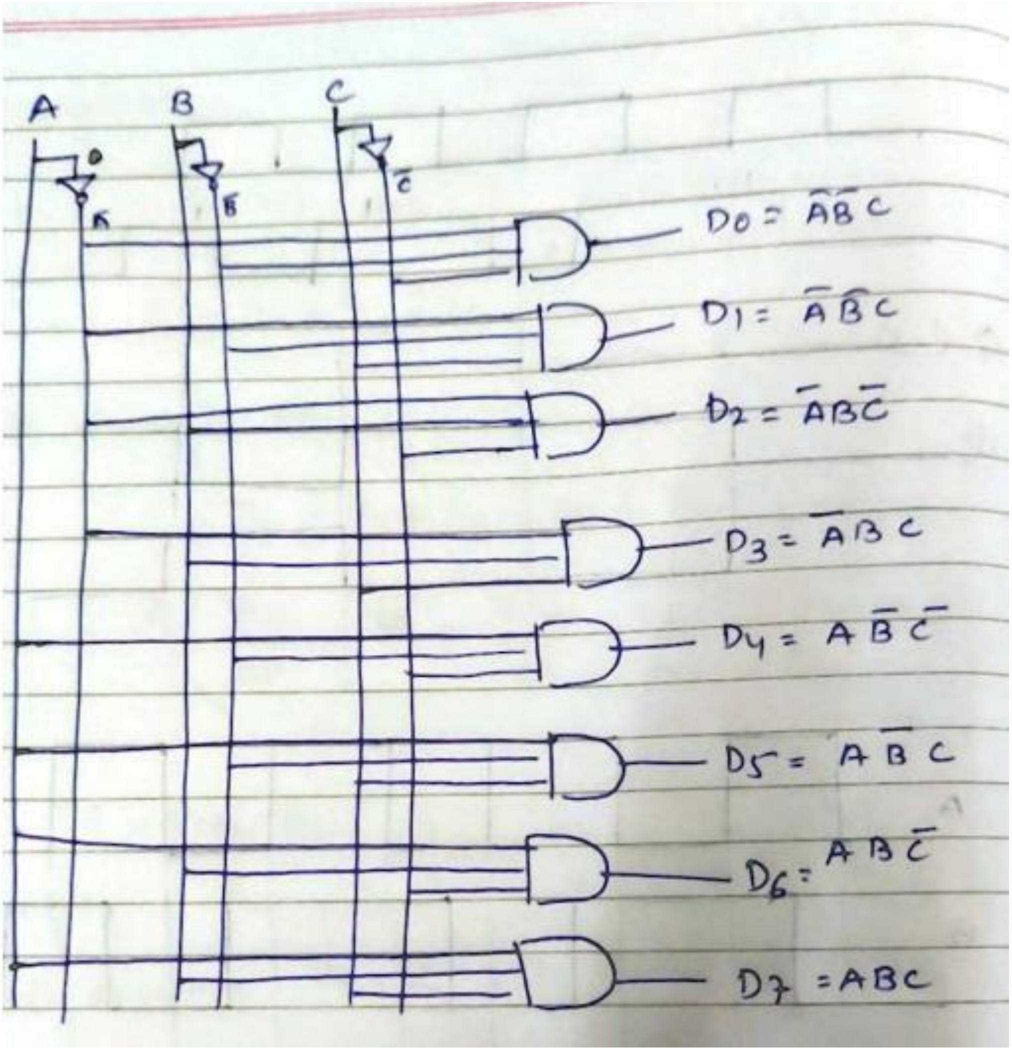
* 2 to 4 Line Decoder:

→ It take 2 inputs & gives four outputs

\# 3:8 line Decoder

↳ Takes 3 input & gives 8 outputs

Inputs			Outputs								
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
0	0	0	1	0	0	0	0	0	0	0	$D_0 : \bar{A}\,\overline{BC}$
0	0	1	0	1	0	0	0	0	0	0	$D_1 = \bar{A}\bar{B}C$
0	1	0	0	0	1	0	0	0	0	0	$D_2 = \bar{A}B\bar{C}$
0	1	1	0	0	0	1	0	0	0	0	$D_3 = \bar{A}BC$
1	0	0	0	0	0	0	1	0	0	0	$D_4 = A\bar{B}\bar{C}$
1	0	1	0	0	0	0	0	1	0	0	$D_5 = A\bar{B}C$
1	1	0	0	0	0	0	0	0	1	0	$D_6 = AB\bar{C}$
1	1	1	0	0	0	0	0	0	0	1	$D_7 = ABC$



A
B
C
D0 = $\bar{A}\bar{B}C$
D1 = $\bar{A}\bar{B}C$
D2 = $\bar{A}B\bar{C}$
D3 = $\bar{A}BC$
D4 = $A\bar{B}\bar{C}$
D5 = $A\bar{B}C$
D6 = $AB\bar{C}$
D7 = ABC

12 Design BCD to Excess-3 code convertor circuit.

→ BCD to Excess-3 code converter:

* For this conversion, 4 bit-BCD code is an input to the circuit.
* The output is 4 bit Excess-3 code
* Since the 0 to 9 is valid inputs ~~so does~~ for the BCD codes, we will consider only 0 to 9 inputs and the output is 0 to 9 Excess 3 code.
* Excess 3 code is generated by adding 3 to the BCD codes
* Here, O/P 10 to 15 will be considered as don't care (x) to construct K-Maps.

Decimal Number	BCD code D_3	D_2	D_1	D_0	Excess-3 code E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

→ For Each output we will obtain K-Map

① for O/P E3

$D_3D_2 \backslash D_1D_0$	00	01	11	10
00	(0) 0	(1) 0	(3) 0	(2) 0
01	(4) 0	(5) 1	(7) 1	(6) 1
11	(12) X	(13) X	(15) X	(14) X
10	(8) 1	(9) 1	(11) X	(10) X

$$E3 = D_3 + D_2 D_0 + D_2 D_1$$

② For E_2 Output

$D_3D_2 \backslash D_1D_0$	00	01	11	10
00	(0) 0	(1) 1	(3) 1	(2) 1
01	(4) 1	(5) 0	(7) 0	(6) 0
11	(12) X	(13) X	(15) X	(14) X
10	(8) 0	(9) 1	(11) X	(10) X

$$E_2 = D_2 \overline{D_1}\,\overline{D_0} + \overline{D_2} D_0 + \overline{D_2} D_1$$

(3) For E_1 output

D_3D_2 \ D_1D_0	00	01	11	10
00	1^{0}	0^{1}	1^{3}	0^{2}
01	1^{4}	0^{5}	1^{7}	0^{6}
11	1^{12}	$\times^{13}$	$\times^{15}$	$\times^{14}$
10	1^{8}	0^{9}	$\times^{11}$	$\times^{10}$

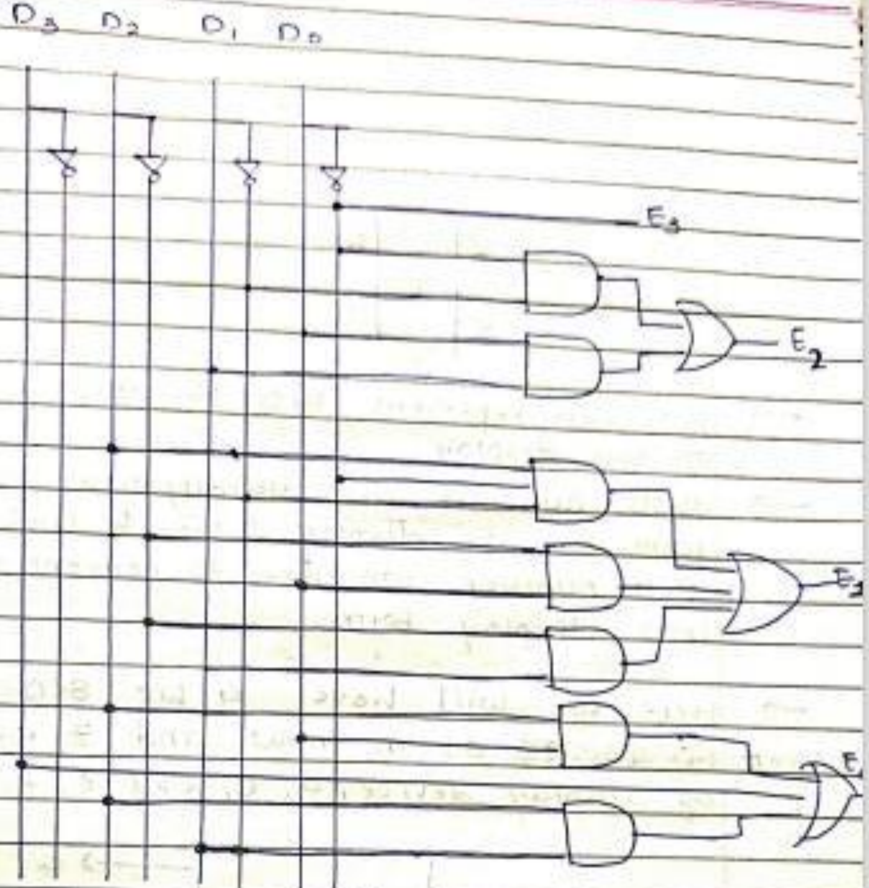
$$E_1 = \overline{D_1}\,\overline{D_0} + D_1 D_0$$

(4) For E_0 Output

D_3D_2 \ D_1D_0	00	01	11	10
00	1^{0}	0^{1}	0^{3}	1^{2}
01	1^{4}	0^{5}	0^{7}	1^{6}
11	$\times^{13}$	$\times^{14}$	$\times^{12}$	$\times^{15}$
10	1^{8}	0^{9}	$\times^{12}$	$\times^{10}$

$$E_0 = \overline{D_0}$$

Logic Diagram:



For $E_1 = \overline{D_1}\,\overline{D_0} + D_1 D_0$.

Ex-NOR gate can be directly used.

13 Design 2-bit magnitude comparator.

* Digital Comparators:-

→ The Digital comparators compares two binary Numbers.

→ It takes two binary Numbers, A and B as an input and gives the output based on A>B, A=B or A<B.

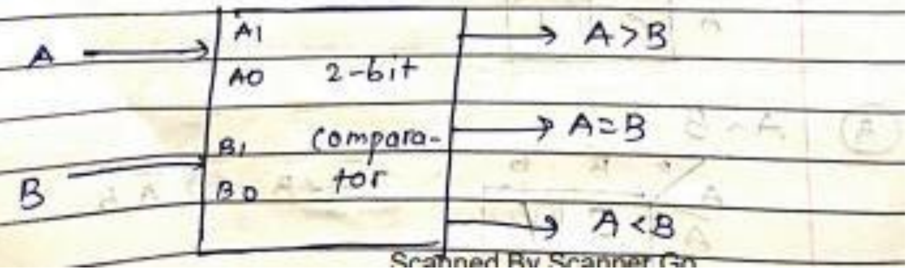
* 2-bit comparator / 2-bit Magnitude comparator.

→ It compares two 2-bit binary Numbers.

$$\left.\begin{array}{l} A \rightarrow 2 \text{ bits } (A_1\ A_0) \\ B \rightarrow 2 \text{ bits } (B_1\ B_0) \end{array}\right\} \text{Inputs}$$

$$\left.\begin{array}{l} A = B \\ A > B \\ A < B \end{array}\right\} \text{outputs.}$$

* Block Diagram.



* Rule:- compare A_1 & B_1

if $A_1 > B_1$ $\longrightarrow$ $A > B$

if $A_1 < B_1$ $\longrightarrow$ $A < B$

if $A_1 = B_1$ $\longrightarrow$ check A_0, B_0

if $A_1 = B_1$ compare A_0 & B_0

if $A_0 > B_0$ $\longrightarrow$ $A > B$

$A_0 < B_0$ $\longrightarrow$ $A < B$

$A_0 = B_0$ $\longrightarrow$ $A = B$.

* Truth Table

Inputs				outputs		
A_1	A_0	B_1	B_0	$A>B$	$A=B$	$A<B$
0	0	0	0	0	1	0
0	0	0	1	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

* equation:- 4 inputs → 4 var K-Map

(1) $A > B$

$A_1A_0 \backslash B_1B_0$	$\overline{B_1}\,\overline{B_0}$	$\overline{B_1}B_0$	B_1B_0	$B_1\overline{B_0}$
$\overline{A_1}\,\overline{A_0}$	0	0	0	0
$\overline{A_1}A_0$	1	0	0	0
A_1A_0	1	1	0	1
$A_1\overline{A_0}$	1	1	0	0

$A>B = A_0\overline{B_1}B_0 + A_1\overline{B_1} + A_1A_0\overline{B_0}$

(2) $A = B$

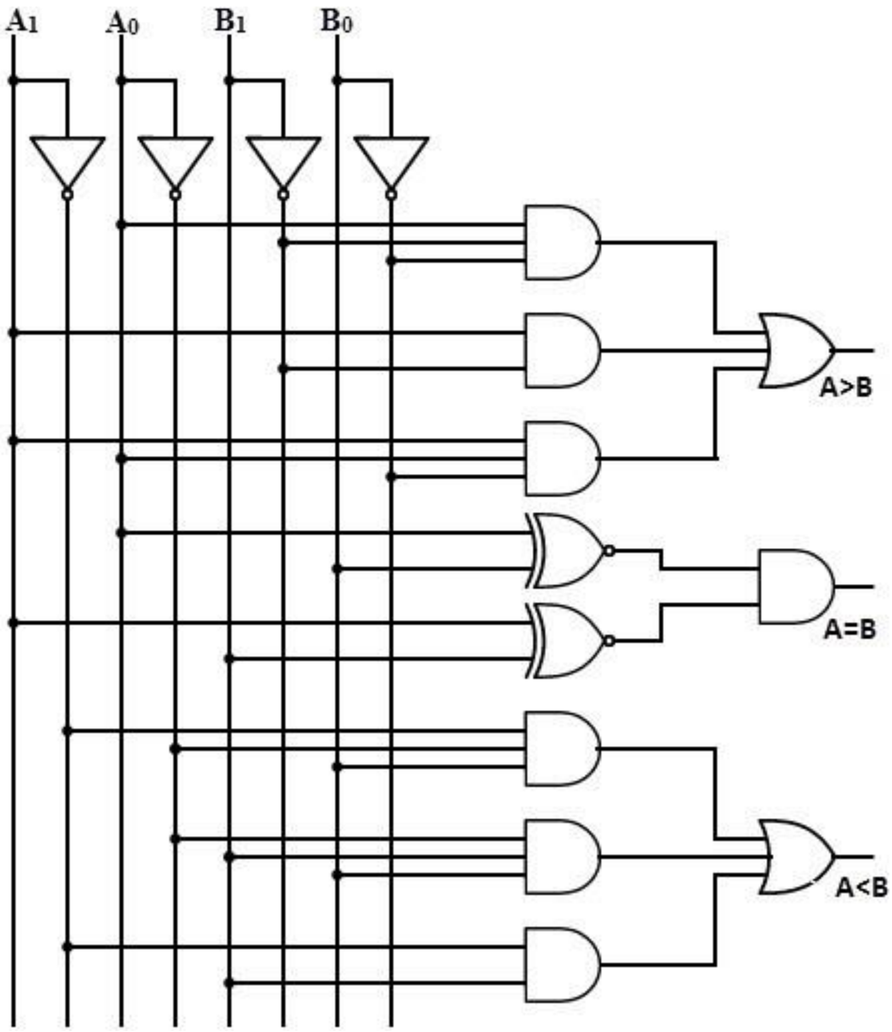
$A_1A_0 \backslash B_1B_0$	$\overline{B_1}\,\overline{B_0}$	$\overline{B_1}B_0$	B_1B_0	$B_1\overline{B_0}$
$\overline{A_1}\,\overline{A_0}$	1	0	0	0
$\overline{A_1}A_0$	0	1	0	0
A_1A_0	0	0	1	0
$A_1\overline{A_0}$	0	0	0	1

$A=B = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}A_0\overline{B_1}B_0 + A_1A_0A_1B_0 +$
$A_1\overline{A_0}B_1\overline{B_0}$
$= \overline{A_1}\,\overline{B_1}(\overline{A_0}\,\overline{B_0} + A_0B_0) + A_1B_1(A_0B_0 + \overline{A_0}\,\overline{B_0})$
$= (A_0 \odot B_0)(A_1 \odot B_1)$

(3) A<B

A_1A_0 \ B_1B_0	$\bar{B_1}\bar{B_0}$	$\bar{B_1}B_0$	B_1B_0	$B_1\bar{B_0}$
$\bar{A_1}\bar{A_0}$	0	1	1	1
$\bar{A_1}A_0$	0	0	1	1
A_1A_0	0	0	0	0
$A_1\bar{A_0}$	0	0	1	0

$$A<B = \bar{A_1}\bar{A_0}B_0 + \bar{A_0}B_1B_0 + \bar{A_1}B_1$$



14 Explain Full Adder in detail.

→ Full Adder:

→ A half adder can add only two bits.

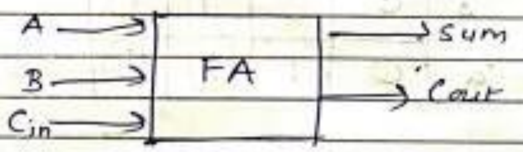
→ There can be possibility that 3rd bit carry can arrive from previous bit addition.

→ so, Addition will become of 3 bits. In that case a full Adder is needed

$$\begin{array}{rl} 1 & \rightarrow C_{in} \text{ (previous carry)} \\ + \; 0 & \rightarrow A \\ 1 & \rightarrow B \\ \hline 1\,0 & \end{array}$$

C_{out} ← 1, 0 → sum.

\# Block Diagram:



\# Truth Table

Inputs			Outputs	
A	B	Cin	Sum	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

Equation:-

3 inputs → 3 variable K-Map

Sum:

$A \backslash BC_{in}$	$\bar{B}\bar{C}_{in}$	$\bar{B}C_{in}$	BC_{in}	$B\bar{C}_{in}$
A	0	1	0	1
$\bar{A}$	1	0	1	0

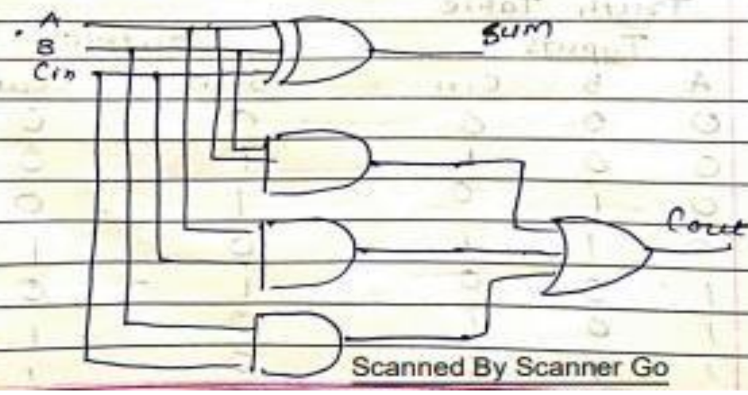
Sum: $\bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$

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

Carry (C_{out}):-

$A \backslash BC_{in}$	$\bar{B}\bar{C}_{in}$	$\bar{B}C_{in}$	BC_{in}	$B\bar{C}_{in}$
A	0	0	1	0
$\bar{A}$	0	1	1	1

$C_{out} = AB + AC_{in} + BC_{in}$

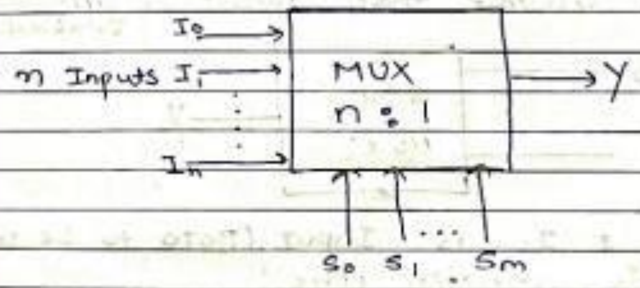


15 Design and explain 4x1 Multiplexer.

★ Multiplexer:

→ Multiplexer / MUX is used when user wants to send mutiple signals over single channel

→ Multiplexer takes n inputs and gives 1 output i.e it selects any 1 o/p from the give n inputs. That is why it is also called Data selector



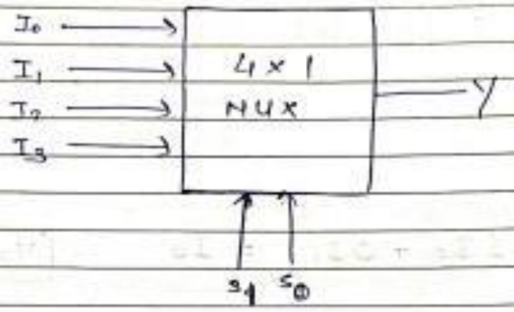
→ n inputs, 1 output, m selection lines.

* 4×1 MUX

→ $n = 4$ (inputs). so, $n = 2^m$ $4 = 2^2 \Rightarrow m = 2$

→ Therefore, there will be 2 selection Lines.

* Block Diagram:



* Truth Table:

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

equation:

$$Y = \bar{S_1}\,\bar{S_0}\,I_0 + \bar{S_1}\,S_0\,I_1 + S_1\,\bar{S_0}\,I_2 + S_1\,S_2\,I_3$$

* proof:

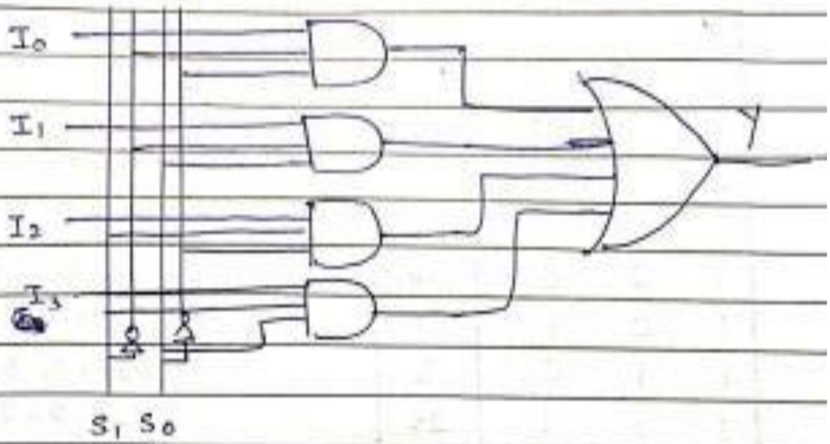
case: 10 $S_1 = 1$ $S_0 = 0$

$$Y = 0.1\,I_0 + 0.0\,I_1 + 1.1\,I_2 + 1.0\,I_3$$

$$= 0 + 0 + I_2 + 0$$

$$\boxed{Y = I_2}$$

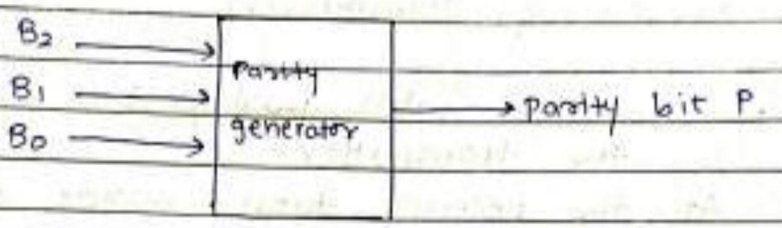
Logic Diagram



16 Design 3-bit even parity generator circuit.

Parity Generator:

→ parity generator is a logic circuit which generates the parity bits for even parity or odd parity.



(2) 3 bit even parity generator.

Truth Table :

Input			Output
B_2	B_1	B_0	Even parity bit (P)
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

→ K-Map :

B_2 \ B_1B_0	00	01	11	10
0	0 [0]	1 [1]	0 [3]	1 [2]
1	1 [4]	0 [5]	1 [7]	0 [6]

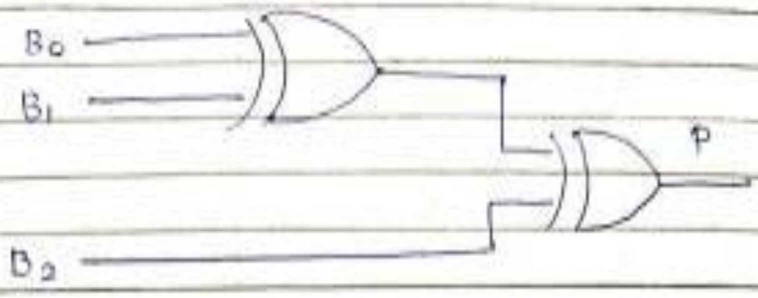
$$P = \overline{B_2}\,\overline{B_1}\,B_0 + \overline{B_2}\,B_1\,\overline{B_0} + B_2\,\overline{B_1}\,\overline{B_0} + B_2 B_1 B_0$$

$$= \overline{B_2}\,(\overline{B_1}\,B_0 + B_1\overline{B_0}) + B_2\,(\overline{B_1}\,\overline{B_0} + B_1 B_0)$$

$$= \overline{B_2}\,(B_1 \oplus B_0) + B_2\,(\overline{B_1 \oplus B_0})$$

$$= B_2 \oplus (B_1 \oplus B_0)$$

Logic diagram:



Here, for ~~pø~~ even parity two X-OR gates will be used.

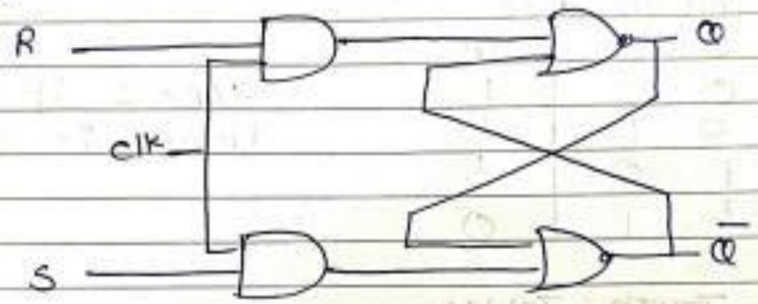
17 Draw and explain working of following flip-flops.
[1] Clocked RS [2] JK

Clocked RS flipflop:- (NOR)

→ In clocked RS flip flop 2 and gates & 2 NOR gates are connected

→ NOR gates are cross coupled & 2 AND gates will be given input R & S with clock signal.

→ Diagram:-



→ logic Diagram

→ graphic symbol

Inputs			Outputs
Q	S	R	Q(t+1)
0	0	0	0 (No change)
0	0	1	0 (Reset)
0	1	0	1 (set)
0	1	1	x (Indeterminate)
1	0	0	1 (No change)
1	0	1	0 (Reset)
1	1	0	1 (set)
1	1	1	x (Indeterminate)

→ Q is the binary state of the flip-flop at a given time named as present state

→ Q(t+1) is the state of the flip-flop after occurrance of flip flop referred as next state.

→ Here, when S=0 R=0, gives NC state
S=0 R=1, gives Reset state
S=1 R=0, gives set state
S=1 R=1, gives Invalid state.

→ The triangle symbol in the graphic symbol of clock pulse. & responds to an input clock transition from low level (0) to high level (1).

→

Q \ SR				
0	0	0	x	1
1	1	0	x	1

$Q(t+1) = S + \overline{R}Q$

$SR = 0$

(means S & R can not be 1 together)

J - set
K - Reset

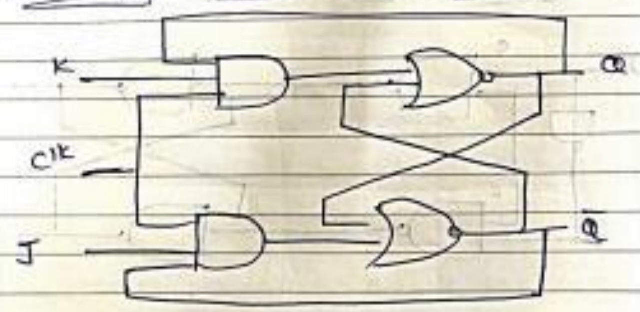
* JK flip-flop

→ Works same as SR flip-flop.
→ The only difference between JK flipflop and SR flipflop is that when both inputs of SR is set to ~~0~~ 1, the circuit produces the invalid states.
→ But in JK flip-flop, there are NO invalid states even if both 'J' & 'K' flip-flop are set to 1.

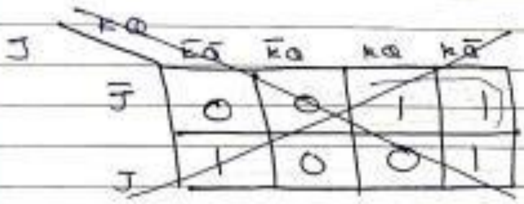
→ 11 input will produce Toggle condition

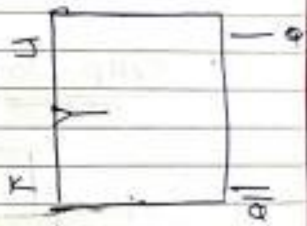
for toggle { so, when $Q = 0$ (prev state) $Q(t+1) = 1$
$Q = 1$ (prev state) $Q(t+1) = 0$

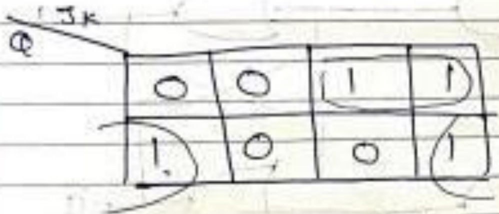
* JK flip flop Using NOR gates



Q	J	K	Q(t+1)	
0	0	0	0	NC
0	0	1	0	Reset
0	1	0	1	Set
0	1	1	1	Toggle
1	0	0	1	NC
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	0	toggle







$Q(t+1) = \bar{Q}J + K\bar{Q}$

When J = 0 K = 0 State is NC
J = 0 K = 1 " Reset
J = 1 K = 0 " set
J = 1 K = 1 " toggle

18 Construct D FF using SR FF. Write truth table of D FF.

3) D flip-flop

Characteristics Tables:- / Truth Table

D	Q(t)	Q(t+1)	
0	0	0	1
0	1	0	3
1	0	1	2
1	1	1	4

→ Excitation Table :-

	Q(t)	Q(t+1)	S	R
1	0	0	0	x
2	0	1	1	0
3	1	0	0	1
4	1	1	x	0

Step 1 Available FF → SR (ET)
Required FF → D (CT)

Step 2 & 3 :-

D	Q(t)	Q(t+1)	S	R
0	0	0	0	x
0	1	0	0	1
1	0	1	1	0
1	1	1	x	0

Step 4 :-

For S

D \ Q(t)	0	1
0	0	0
1	1	x

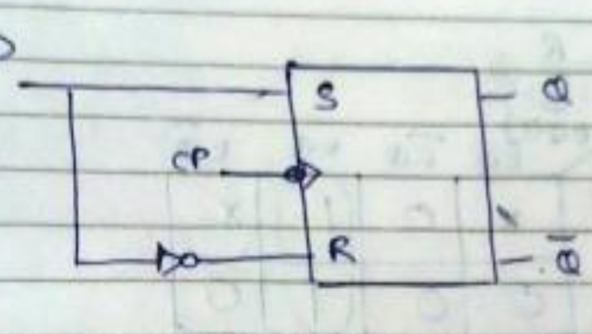
$\boxed{S = D}$

For R

D \ Qn	0	1
0	x	1
1	0	0

$\boxed{R = \overline{D}}$

Step 5 :-

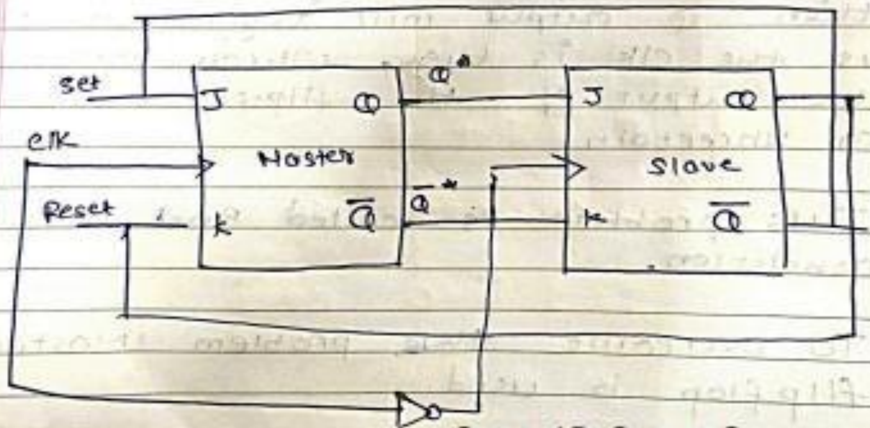


19 Explain Master Slave JK flip-flop with truth table and circuit diagram.

- Master-slave Flip-flop

↳ A Master-slave flip flop is constructed from two separate flipflops.

↳ One circuit serves as a master and the other as a slave. so, the overall circuit is referred to as master-slave flip-flop.



→ In addition of these two flip-flops, the Circuit also includes an inverter.

→ Here, Inverter is connected with clk pulse of slave flipflop.

→ So, when Clk = 1 for master flipflop,
Clk = 0 for slave flipflop &
vice versa.

Working:

→ When the Clk = 1, Master FF will work & Slave FF will be isolated.

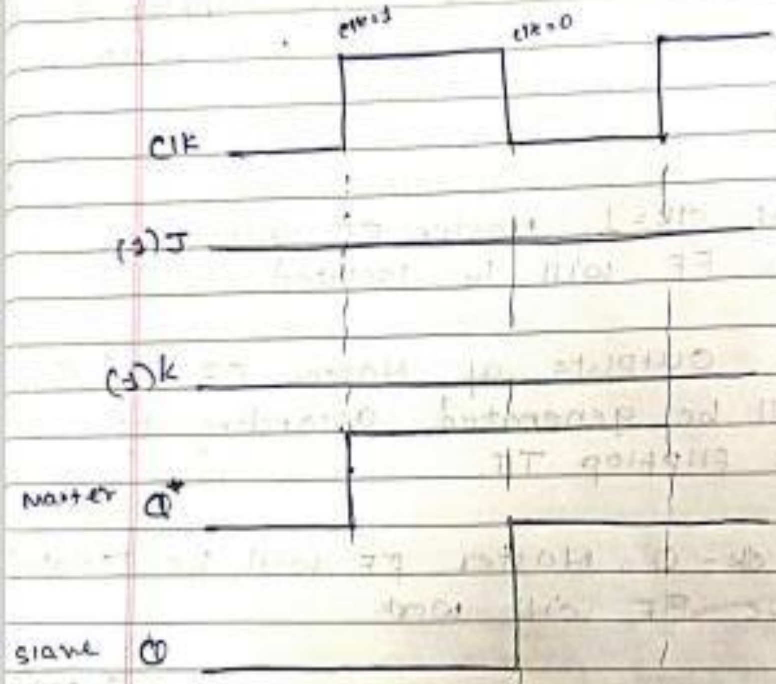
→ So, the outputs of Master FF (Q^* & $\overline{Q}^*$) will be generated according to the JK flipflop TT.

→ When clk = 0, Master FF will be isolated & slave FF will work.

→ Now, the outputs of Master FF (Q^* & $\overline{Q}^*$) will be input to the slave FF & outputs will be produced according to JK ~~ff~~ flip-flop.

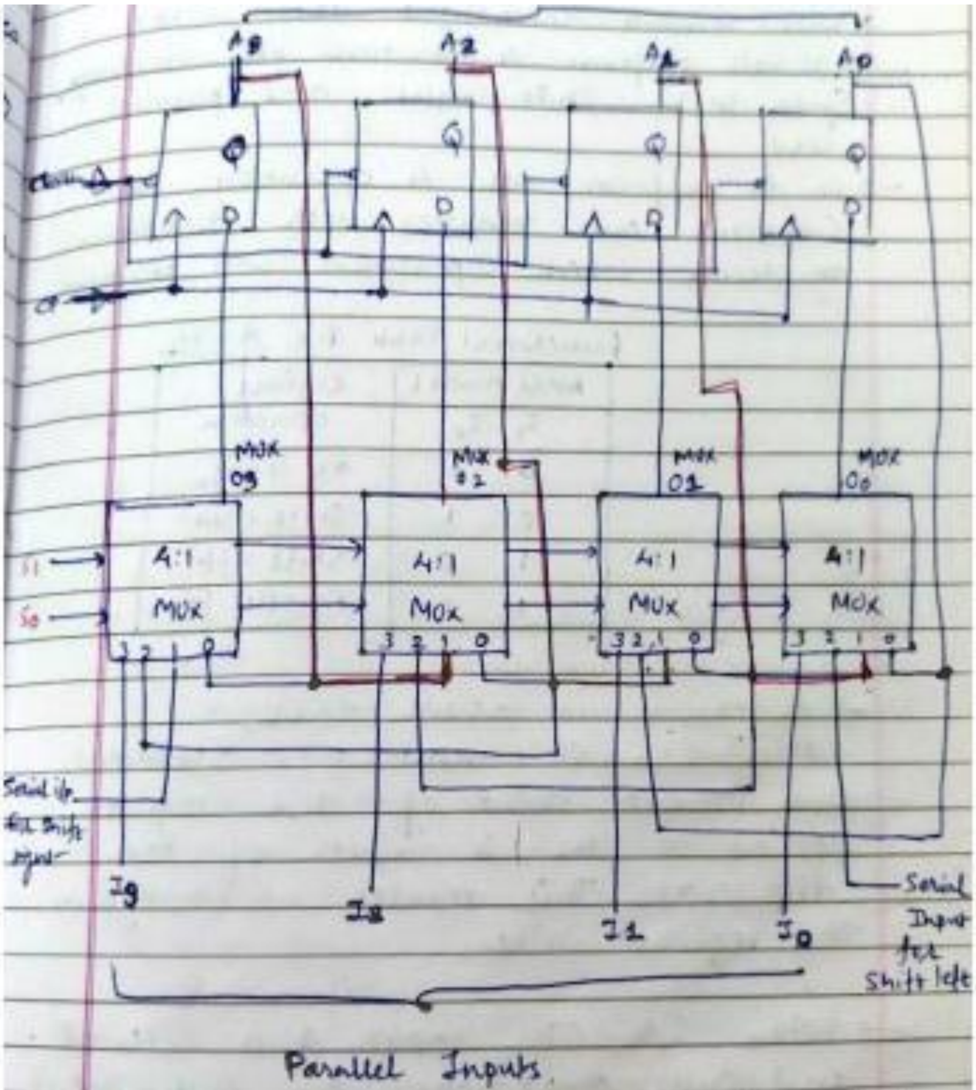
→ Since, the second flipflop follows the first one, it is referred as the slave & first one as the master.

→ In master-slave JK flip-flop state Change occurs when flip-flop goes through both positive transition (First half) of clock and Negative transition of the clock (second half). Thus, race-around condition does not exist in Master slave JK flip flop.



Timing Diagram of Master-slave J-K flip flop.

20 Draw and explain the block diagram of 4-bit bidirectional shift register with Parallel load.



→ Universal shift register can perform both the operation at same time (shift left and shift right)

→ Apart from this we can perform

no change operation means before any operation FF content is like 1011, so after performing any operation content will remain unchanged that is 1011

→ It will perform 4 operations like no change, Shift left, Shift right and parallel load.

→ So to perform all 4 operation, we connected our register with multiplexer to decide which operation will perform.

Functional Table for register

Mode control S_1 S_0		Register operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

→ As shown in above table,
when $S_1S_0 = 00$, input = 0 is selected and present value of the register is applied to the D inputs of the flip-flops. This results no change in the register value.

→ When $S_1S_0 = 01$, input 1 is selected and circuit connections are such that it operates as a right shift register.

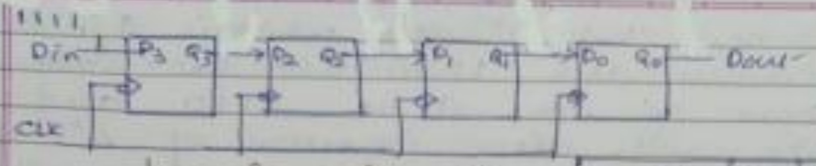
→ When $S_1S_0 = 10$, input 2 is selected and circuit connections are such that it operates as a left shift register.

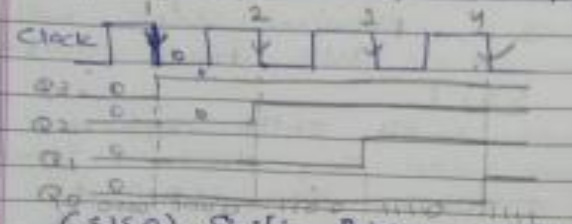
→ Finally when $S_1S_0 = 11$, the binary information on the parallel input lines is transferred into the register simultaneously and it is a parallel load operation.

21 Draw and explain 4-bit serial-in serial-out shift register using D FFs.

Serial Input Serial Output
(4 bit)

Page No.
Date :





CLK	Q_3	Q_2	Q_1	Q_0
Initial	0	0	0	0
1 ↓	1	0	0	0
2 ↓	1	1	0	0
3 ↓	1	1	1	0
4 ↓	1	1	1	1

(SISO) Shift Right register)

→ In shift Right right register we have Serial Input and serial output.
→ We have 3 terminal input, output, Clock signals and D FFs.
→ It is 4 bit shift register
→ First of all we want to transfer data 1111 serialy
→ Initailly all data of Q_1 Q_2 Q_3 Q_4 are 0 as shown in table.
→ As per basic working of D flipflop as we give clock I/P data will go at O/P side.
→ So data at O/P Q_3 will be shifted to Q_2, data of Q_2 will shifted to Q_1 & Q_1 data get shifted to Q_0
→ So at 2^{nd} clock signal I/P after inserting data 1 data of Q_3 which is 1 is shifted to Q_2, Q_2 data will be shifted to Q_1 & Q_1 data will be shifted to Q_0 so the 2^{nd} O/P will be 1100.

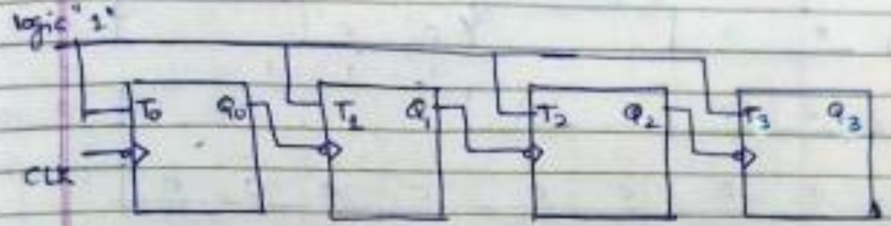
→ In next clock 3^{rd} data which is 1, so Previous data 1100 will get shifted right so O/P will be 1110 and like will after insertion of 4^{th} bit which is 1 previous data will be shifted right 4' but final output will be 1111
→ So here how 4-bit shift register works so we can say that to store data n-no of clocks are required.

22 **Explain working of 4-bit binary ripple counter** **<u>OR</u>** **Explain the working of 4 bit asynchronous counter.**

– 4-bit Asynchronous ripple counter

A 4 bit ripple counter can count up to 2^4 states.

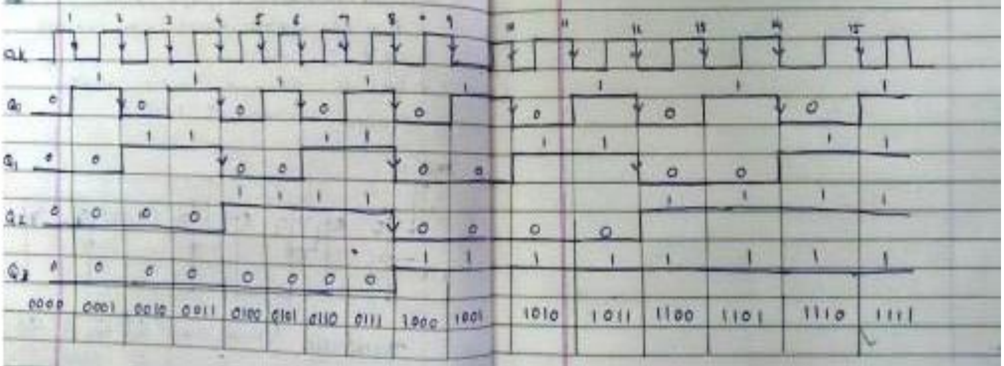
→ We will implement 4-bit ripple counter using T FlipFlop.



Truth Table

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

Timing Diagram



23 Design a counter to generate the repetitive sequence 0, 1, 2,4,3,6.

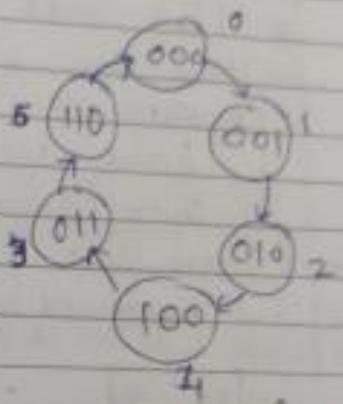
Design a counter with repetitive sequence 0, 1, 2, 4, 3, 6, 0 Using T flipflop

Step 1: decide No of Flipflop required $n=3$ FFs

Step 2: State diagram

Excitation Table of T

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



State diagram

tate table

	Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	T_A	T_B	T_C
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
2	0	1	0	1	0	0	1	1	0
3	0	1	1	1	1	0	1	0	1
4	1	0	0	0	1	1	1	1	1
5	1	0	1	x	x	x	x	x	x
6	1	1	0	0	0	0	1	1	0
7	1	1	1	x	x	x	x	x	x

Page No.
Date :

T_A

A \ BC	00	01	11	10
$\bar{A}$ 0	0	0	1	1
A 1	1	X	X	1

T_B

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

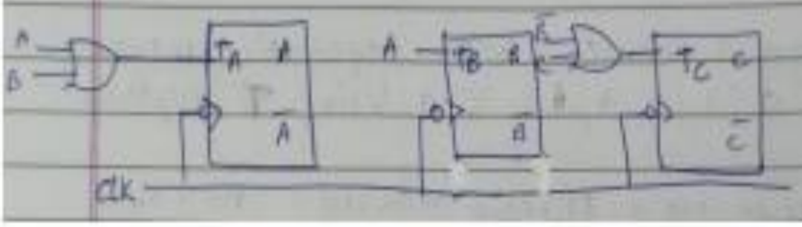
T_C

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

$T_A = A + B$

$T_B = A$

$T_C = \bar{B} + C$

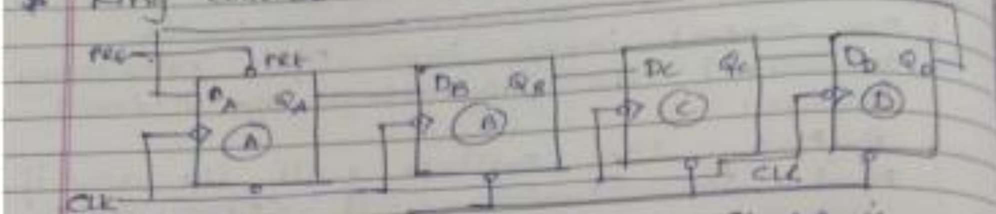


24 Draw and explain Ring counter.

Date :

Shift Register counters

* Ring Counter: (kind of SISO shift register)



→ Here Q output of each stage is connected to the D input of the next stage and output of last FF is given to the first FF.

→ PRE (Preset) makes output $Q_A = 1$ and $Q_B, Q_C, Q_D = 0$

→ Here in Ring Counter High bit (1) is rotated so firstly $Q_A Q_B Q_C Q_D = 1000$. In next clk $Q_A Q_B Q_C Q_D$ will be 0100, next clk $Q_A Q_B Q_C Q_D$ will be 0010 & in last clock pulse $Q_A Q_B Q_C Q_D$ will be 0001 & then again after giving clk pulses it becomes $Q_A Q_B Q_C Q_D = 1000$. So that is why it is said to be a ring counter.

Clk	Q_A	Q_B	Q_C	Q_D
Initially	1	0	0	0
↓	0	1	0	0
↓	0	0	1	0
↓	0	0	0	1
↓	1	0	0	0

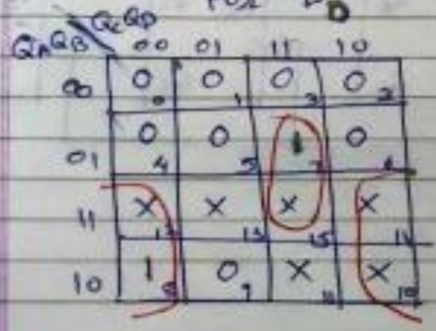
25 Design BCD/MOD-10 synchronous Counter.

Step 1: MOD-10 counter has ten states 0 to 9. So, 4 Flip Flops will be required.

Step 2: Types of FF : D

Present State				Next State				Input to FF			
Q_D	Q_C	Q_B	Q_A	Q_D^+	Q_C^+	Q_B^+	Q_A^+	D_D	D_C	D_B	D_A
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	1	0	0	1
1	0	0	1	0	0	0	0	0	0	0	0

K-map Simplification



$D_D = Q_A\bar{Q}_D + Q_BQ_CQ_D$.

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

For D_B

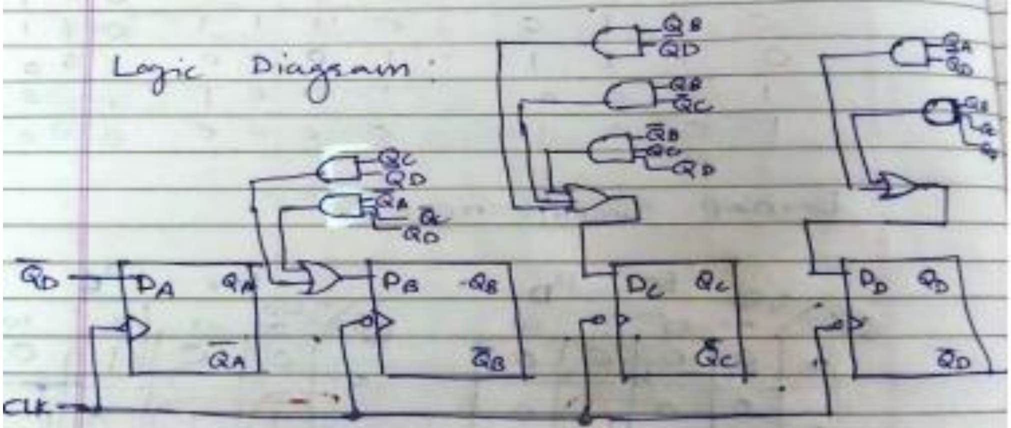
$Q_AQ_B \backslash Q_CQ_D$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	X	X	X	X
10	0	0	X	X

For D_A

$Q_AQ_B \backslash Q_CQ_D$	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$D_B = Q_C\bar{Q}_D + \bar{Q}_A\bar{Q}_C Q_D$ $\qquad$ $D_A = \bar{Q}_D$

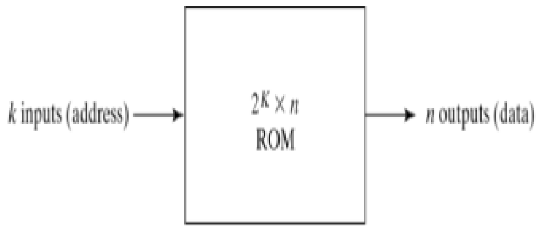
Logic Diagram:



26 Write a short note on ROM & its types.

Read Only Memory (ROM)

- ✓ ROM is a type of Primary Memory. As the name suggests its contents can be read only but cannot write on it. It is a non-volatile memory and so the data is retained even when the power is switched off. A ROM is essentially a memory device in which permanent binary information is stored .The data that is required to be stored inside ROM is written during manufacturing phase. It stores such programs that are essential for the booting process of the computer. It generally cannot be altered. However, technologies are available to program these types of ROM.
- ✓ Read-only memory is a non-volatile storage solution. This is because you cannot erase or modify it when the computer system is turned off. Computer manufacturers write codes on the ROM chip, and users cannot alter or interfere with it. But there are modern types of ROMs which can actually be deleted or modified despite the fact that they are non-volatile.
- ✓ A block diagram of ROM is shown in the Figure below. It consists of k inputs and n outputs. The inputs provide the address for the memory and the outputs give the data bits of the stored word which is selected by the address. The number of words in a ROM is determined from the fact that k address input lines are needed to specify 2 k words.



Classification of ROM

- Mask Read-Only Memory (MROM)
- Programmable Read-Only Memory (PROM)
- Erasable Programmable Read-Only Memory (EPROM)
- Electrically Erasable Programmable Read-Only Memory (EEPROM)

PROM (Programmable Read-Only Memory)

- PROM stands for Programmable Read Only Memory. PROM is manufactured as a blank memory. And as its name suggests Programmable, it is programmed after manufacturing. The user buys a blank memory and enters the desired contents using a PROM program.
- PROM consists of fixed AND array and programmable OR array. It has n inputs and m outputs. For n input variables, there are 2^n distinct addresses. In simple words, we can say that the fixed AND array acts as a decoder (n: 2^n).

EPROM (Erasable Programmable Read-Only Memory)

- EPROM stands for Erasable Programmable Read-Only Memory. It is a non-volatile memory i.e. it can retain data even if the power supply is cut off. The basic limitation being encountered in PROM is that once it is programmed, it cannot be changed or altered. This limitation has been overcame by EPROM.
- EPROM can be erased by exposing it to ultra violet light for a particular length of time using an EPROM eraser. After exposing, the chip returns to its initial state and can be reprogrammed.
- This procedure can be carried out many times but repeated erasing and rewriting can eventually render the chip useless. Once written, data can be retained for about 10 years.

EEPROM (Electrically Erasable Programmable Read-Only Memory)

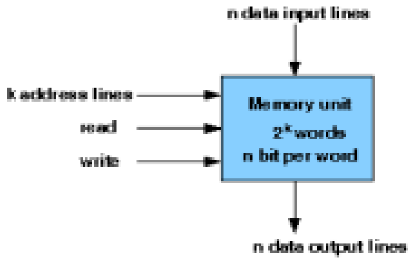
- EEPROM is the short form for Electrically Erasable Programmable Read Only Memory. It is similar to EPROM and thus developed to overcome the drawbacks of EPROMs. It is erased and programmed electrically i.e. it uses electrical signals instead of ultra violet rays.
- The erasing and programming of data takes 4 to 10 milliseconds. Any byte can be erased at a time instead of the entire chip. The chip can be erased and re programmed for around ten thousand times, though the process is flexible but slow.

27 Explain memory unit.

Semiconductor memory is a digital electronic semiconductor device used for digital data storage, such as computer memory. It typically refers to MOS memory, where data is stored within metal–oxide–semiconductor (MOS) memory cells on a silicon integrated circuit memory chip. Memory is the storage space in computer where data is to be processed and instructions required for processing are stored. It is the most essential component for the normal functioning of any system – to store data, to perform calculations, to do complex operations, etc.

The memory is divided into large number of small parts. Each part is called a cell. Each location or cell has a unique address which varies from zero to memory size minus one.

A memory unit stores binary information in groups of bits called words. A word in memory is an entity of bits that move in and out of storage as a unit. A memory word is a group of 1's and 0's and may represent a number, an instruction, one or more alphanumeric characters, or any other binary-coded information. A group of eight bits is called a byte. Most computer memories use words that are multiples of 8 bits in length. Thus, a 16-bit word contains two bytes, and a 32-bit word is made up of four bytes. The capacity of a memory unit is usually stated as the total number of bytes that it can store.



Each word in memory is assigned an identification number, called an address, starting from 0 up to 2^k -1, where k is the number of address lines.

Write and Read operations:

Transferring a new word to be stored into memory:

1. Apply the binary address of the desired word to the address lines.
2. Apply the data bits that must be stored in memory to the data input lines.
3. Activate the write input.

Transferring a stored word out of memory:

1. Apply the binary address of the desired word to the address lines.
2. Activate the read input.

For example if computer has 64k words, then this memory unit has 64 * 1024 = 65536 memory location. The address of these locations varies from 0 to 65535.

28 Write short note on Programmable Logic Arrays.

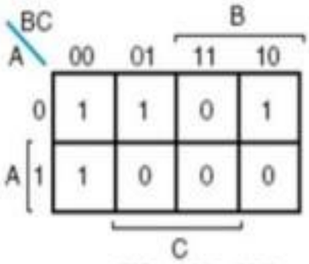
Programmable Logic Array (PLA)

- The PLA is similar to PROM in concept except that PLA does not provide full decoding of the variable and does not generate all the minterms. The decoder is replaced by an array of AND gates that can be programmed to generate any product term of the input variables. The product terms are then connected to OR gates to provide the sum of products for the required Boolean functions.
- Since PLA has m-outputs, the number of OR gates is m. The output of each OR gate goes to an XOR gate, where the other input has two sets of links, one connected to logic 0 and other to logic 1. It allows the output function to be generated either in the true form or in the complement form

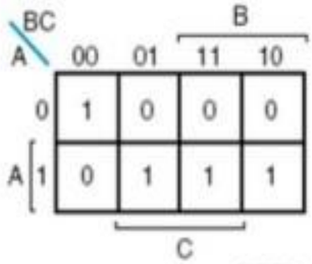
Implement the combinational circuit having the shown truth table, using PLA.

A	*B*	*C*	F_1	F_2
0	0	0	1	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

Each product term in the expression requires an AND gate. To minimize the cost, it is necessary to simplify the function to a minimum number of product terms.



$$F_1 = \overline{A}\,\overline{B} + \overline{A}\,\overline{C} + \overline{B}\,\overline{C}$$
$$\overline{F}_1 = AB + AC + BC$$



$$F_2 = AB + AC + \overline{A}\,\overline{B}\,\overline{C}$$
$$\overline{F}_2 = \overline{A}C + \overline{A}B + A\overline{B}\,\overline{C}$$

<u>Designing using a PLA, a careful investigation must be taken in order to reduce the distinct product terms.</u> Both the true and complement forms of each function should be simplified to see which one can be expressed with fewer product terms and which one provides product terms that are common to other functions.

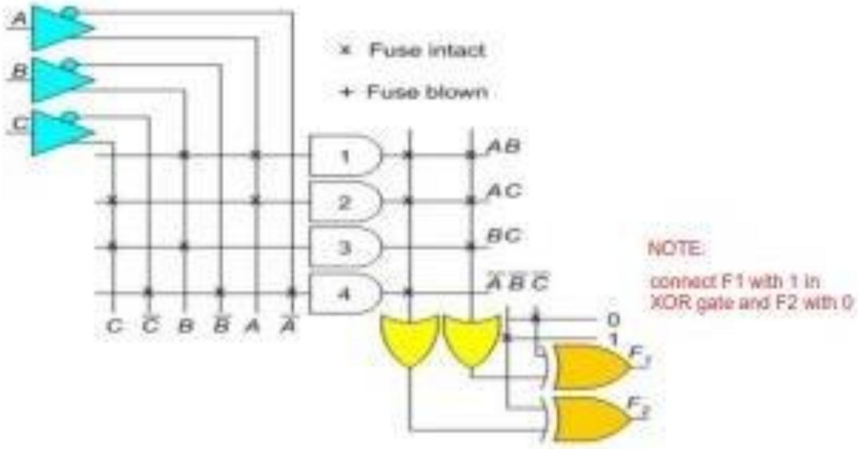
The combination that gives a minimum number of product terms is,

F1' = AB + AC + BC, F2 = AB + AC + A'B'C'

This gives only 4 distinct product terms: AB, AC, BC, and A'B'C'. So the PLA table will be as follows, PLA programming table

	Product term	Inputs A B C	Outputs (C) F_1	Outputs (T) F_2
AB	1	1 1 –	1	1
AC	2	1 – 1	1	1
BC	3	– 1 1	1	–
$\overline{A}\overline{B}\overline{C}$	4	0 0 0	–	1

- **For each product term, the inputs are marked with 1, 0, or – (dash). If a variable in the product term appears in its normal form (unprimed), the corresponding input variable is marked with a 1.**
- **A 0 in the Inputs column specifies a path from the corresponding complemented input to the input of the AND gate.**
- **A dash specifies no connection.**



The appropriate fuses are blown and the ones left intact form the desired paths. It is assumed that the open terminals in the AND gate behave like a 1 input. Note that output F1 is the normal (or true) output even though a C (for complement) is marked over it. This is because F1' is generated with AND-OR circuit prior to the output XOR. The output XOR complements the function F1' to produce the true F1 output as its second input is connected to logic 1.

29 **A combinational circuit is defined by functions:**

F1(A,B,C) = ∑(3, 5, 6, 7)

F2(A,B,C) = ∑(0, 2, 4, 7)

Implement the circuit with PLA having three inputs, four product term and two outputs.

$F_1 = \Sigma(3, 5, 6, 7)$ $\quad F_2 = \Sigma(0, 2, 4, 7)$

→ step 1:-

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

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

$F_1 = BC + AC + AB$ $\quad F_2 = \bar{B}\bar{C} + \bar{A}\bar{C} + ABC$

Now find, $\overline{F_1}$ & $\overline{F_2}$

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

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

$\overline{F_1} = \bar{B}\bar{C} + \bar{A}\bar{B} + \bar{A}\bar{C}$

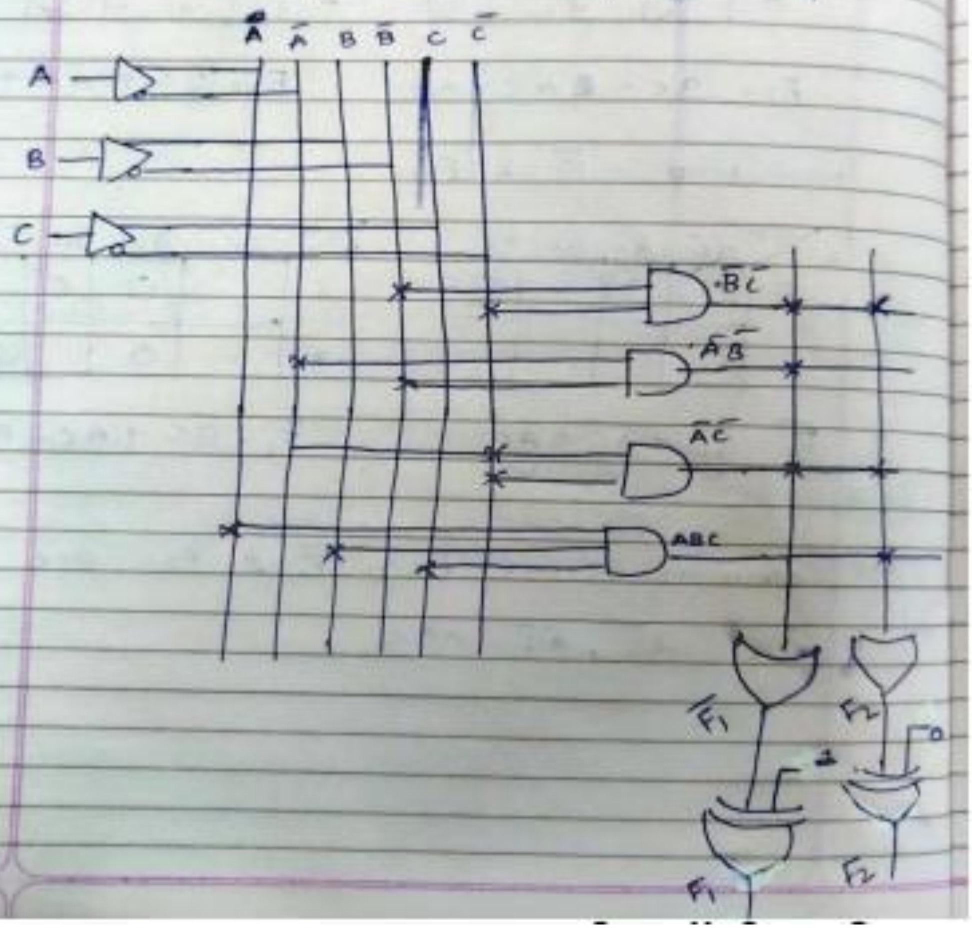
$\overline{F_2} = \bar{B}C + \bar{A}C + AB\bar{C}$

Common terms from $\overline{F_1}$ & F_2 are

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

→

Product Terms	A	B	C	F_1	F_2
	Inputs			outputs	
$\bar{B}\bar{C}$	–	0	0	1	1
$\bar{A}\bar{B}$	0	0	–	1	–
$\bar{A}\bar{C}$	0	–	0	1	1
ABC	1	1	1	–	1
				C	T



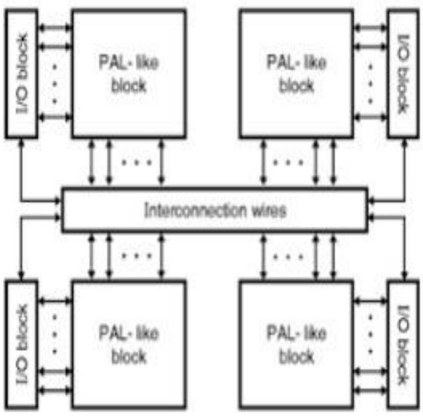
30 Draw and explain in brief block diagram of CPLD. Also compare CPLD with FPGA.

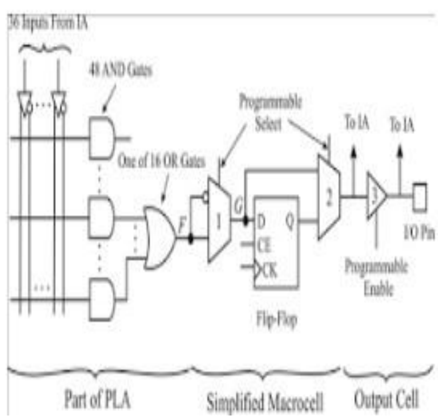
COMPLEX PROGRAMMABLE LOGIC DEVICES (CPLD)

- A CPLD contains a bunch of PLD blocks whose inputs and outputs are connected together by a global interconnection wires.
- A CPLD is an arrangement of many SPLD-like blocks on a single chip. These circuit blocks might be either PAL-like or PLA-like blocks.
- Thus a CPLD has two levels of programmability: each PLD block can be programmed, and then the interconnections between the PLDs can be programmed.
- A PAL like block in CPLD usually consists of 16 macrocells.The macrocell in CPLD consist of AND-OR configurations.
- The EX-OR gate provides the output of OR-gate in inverted or non-inverted form.
- A D-Flipflop stores the output of EX-OR gate.
- A multiplexer select either the output of the D FF or the output of EX-OR gate depending upon selection of inputs.
- The tri-state buffer is used to enable or disable the output.

Characteristics: They have a higher input to logic gate ratio. These devices are denser than SPLDs but have better functional abilities.

- CPLDs are based on EPROM or EEPROM technology. If you require a larger number of microcells for a given application, ranging anywhere between 32 to 1000 microcells, then a Complex Programmable Logic Device is the solution.
- Thus, we use CPLD in applications involving larger I/Os, but data processing is relatively low





Differences between FPGA and CPLD

FPGA	CPLD
Suited for timing circuit because they have more registers.	CPLD is suited for control circuit because they have more combinational circuit.
FPGA consist of CLB,I/O blocks, row and column interconnect	CPLD consist of PAL like blocks, I/O blocks, and programmable interconnect structures.
FPGA use memory called LUT to generate logic functions.	CPLD use AND/OR configuration to generate logic functions.
FPGA has more flexibility as well as design capacity.	CPLD has less compared to FPGA regarding design complexity
Architecture is based on "Look UP Table"	Architecture is based on "Logic function "
FPGA can operate at very high speed	CPLD has less
Cost is high	Cost is Less
Timing Analysis is complex to determine.	Timing Analysis is easier to determine.
The FPGA are volatile in many cases, that's way they need a configuration memory for working with programmed design.	CPLD devices are not volatile. They contain flash or erasable ROM memory in all of cases.
FPGA could not work until the configuration is done.	The CPLD could work immediately after power up.
FPGA is RAM base.	CPLD is ROM base.
FPGAs can contain very large digital designs	CPLDs can contain small designs only.

31. Compare the following in every aspect. RAM and ROM.

RAM	ROM
RAM is a volatile memory which could store the data as long as the power is supplied.	ROM is a non-volatile memory which could retain the data even when power is turned off.
Data stored in RAM can be retrieved and altered.	Data stored in ROM can only be read.
Used to store the data that has to be currently processed by CPU temporarily.	It stores the instructions required during bootstrap of the computer.
It is a high-speed memory.	It is much slower than the RAM
It is costlier than ROM.	It is cheaper than RAM.
Types: DRAM (Dynamic Random Access Memory), SRAM (Static Random Access Memory).	Types: PROM (programmable read-only memory), EPROM (erasable programmable read-only memory), EEPROM(electrically erasable programmable ROM), Mask ROM.
It is large in size than ROM with high capacity	It is smaller in size than ROM with less capacity

32 Design a full adder circuit using two half adders and gates.

HA | Sum = Input1 ⊕ Input2
Carry = Input1 · Input2

Full Adder using two half Adders:-

→ It can be constructed with 2 HA & 1 OR gate

For Full Adder equation,

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

$C_{out} = AB + AC_{in} + BC_{in}$

→ lets make equation of sum, we need 2 HA for that

$\boxed{C_{out} = AB + A\bar{B}C_{in} + \bar{A}BC_{in}}$ RHS

→ output of sum will be directly derived from 2 HA.

→ output Cout will be generated by ORing carry o/p of HA1 & carry o/p of HA 2.

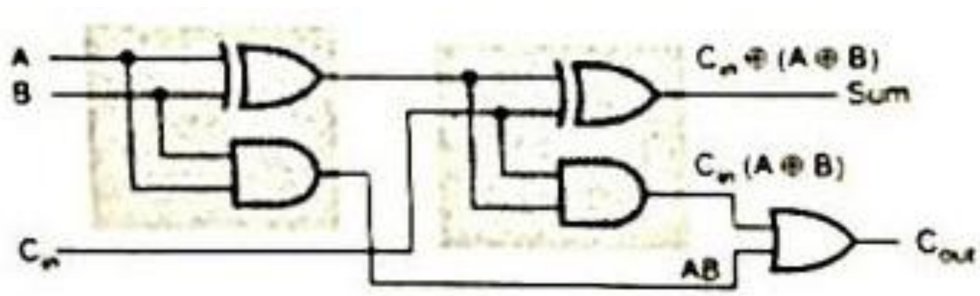
→ $C_{out} = AB + AC_{in} + BC_{in}$ (LHS)

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

$= AB + ABC_{in} + A\bar{B}C_{in} + ABC_{in} + \bar{A}BC_{in}$

$= AB(1 + C_{in} + C_{in}) + A\bar{B}C_{in} + \bar{A}BC_{in}$

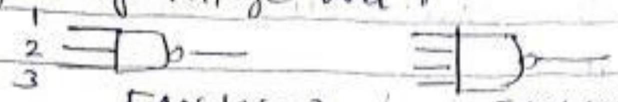
$\boxed{C_{out} = AB + A\bar{B}C_{in} + \bar{A}BC_{in}}$ Proved



33 Define: Fan in, Noise Margin, Propagation delay, Fan out, Negative Logic, Figure of merit, Power dissipation.

- **Fan-in:**

FAN-IN :- max. Number of Inputs to the single gate is called FAN-IN of the gate without degrading voltage level



FAN-IN = 3 FAN-IN = 4

- **Noise margin/Noise Immunity:**

Noise immunity is the max noise voltage added to an I/P signal of a digital ckt that doesn't cause an undesirable change in ckt O/P.

- **Propagation Delay:**

It is the average transition delay time for the signal to propagate from i/p to o/p when binary I/P signal changes in value.

- **Fan out:**

FAN-OUT: (Loading)

The FAN-OUT of a gate specifies the number of loads that can be connected to the output of gate without affecting (degrading its normal operation.

- **Negative Logic:**

In negative logic system, lower voltage represents 1 and Higher voltage represent 0.

- **Figure of Merit/Speed Power Product:**

Figure of Merit / Speed Power Product.
A common mean for measuring & comparing the overall performance of IC family is speed power product / Figure of merit.
Speed Power Product = Propagation Delay × Power Dissipation.

- **Power dissipation:** It is power used by gates when all inputs are connected to the gate.

34 Explain the specification of D/A converter.

* parameters / specifications of DAC

(1) Resolution :-

(i) The number of different analog outputs that can be provided by a DAC. for n-bit DAC

$$\text{Resolution} = 2^n$$

(ii) Resolution is also defined as the ratio of a change in o/p voltage resulting from change of 1 LSB at the digital inputs.

For an n-bit DAC it can be given as

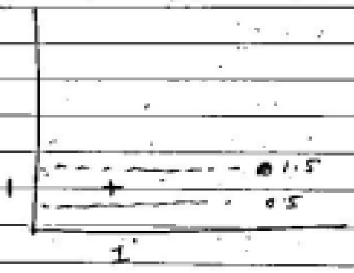
$$\text{Resolution} = \frac{V_{OFS}}{2^n - 1} \Rightarrow \frac{3}{2^2 - 1} = \frac{3}{3} = 1$$

V_{OFS} → full scale o/p voltage.

(2) Accuracy

→ It is a comparision of actual output voltage with expected output voltage.

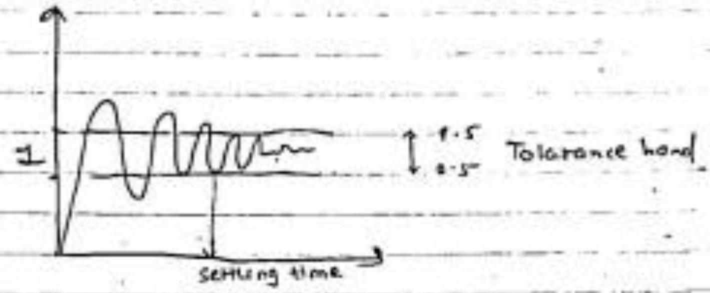
→ It must be betn ±½ of its LSB.



$$\text{Accuracy} = \frac{V_{OFS}}{(2^n - 1) \times 2} = \frac{3}{(2^3 - 1) \times 2} = \frac{3}{3 \times 2} = 0.5$$

(So, o/p must be betn ±0.5 of Desired (Required) output

③ → Settling time:-

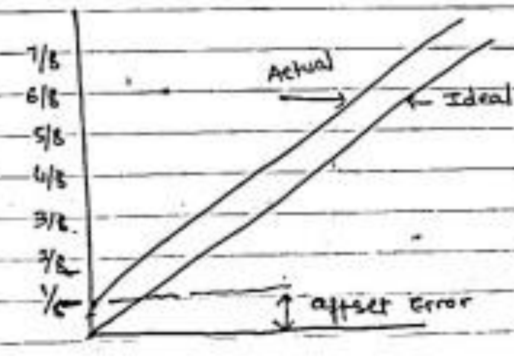


→ The time taken by the o/p to settle within a tolarance band around the steady state value is called.

④ → Dynamic Range:-

- The dynamic range of DAC is defined as the ratio of the largest output to the smallest output; excluding zero.
- It is expressed in dB.

⑤ → Offset Error.



→ The offset error is defined as the non-zero level of the output voltage when all inputs are zero.

→ It is due to current leakage.

* gain Error :-

(6)

→ Gain Error :-

- The gain error is defined as the difference betn calculated gain of the current to voltage converter and actual gain achieved.

(7)

→ Non-Linearity

→ For an ideal DAC, the output voltage would be a linear function of input code.

→ But Because of Non-exact values of resistors the DAC departs somewhat from the ideal linearity.

(8) Stability

→ The performance of converter changes with temperature, age & power supply variations. So all the relevent parameters must be specified over the full temperature & powersupply range.

35 List out various commonly used D/A converters. Draw & explain any one D/A converter.

→ Basic conversion Techniques of D/A

1. Binary weighted resistor D/A Converter
2. R/2R Ladder D/A converter.

1.

→ Binary weighted Resistor Type D/A Converter:-

↳ In this techniques, op-Amp & different values of Resistors are used.

↳ op-Amp is used as summing Amplifier (To sum up all the currents)

↳ The resistors are Binary weighted, meaning Different values of resistors are 2R, 2^2R, 2^3R, ..., 2^nR.

↳ where, n is the number of input bits.

↳ Here, $b_1, b_2, b_3, \ldots, b_n$ are input bits.

* ↳ b_1 is connected to 2R, and b_1 is MSB
↳ b_2 is connected to 4R and so on...
↳ b_n is connected to 2^nR & b_n is LSB.

* $b_1, b_2 \ldots b_n$ are connected with switch

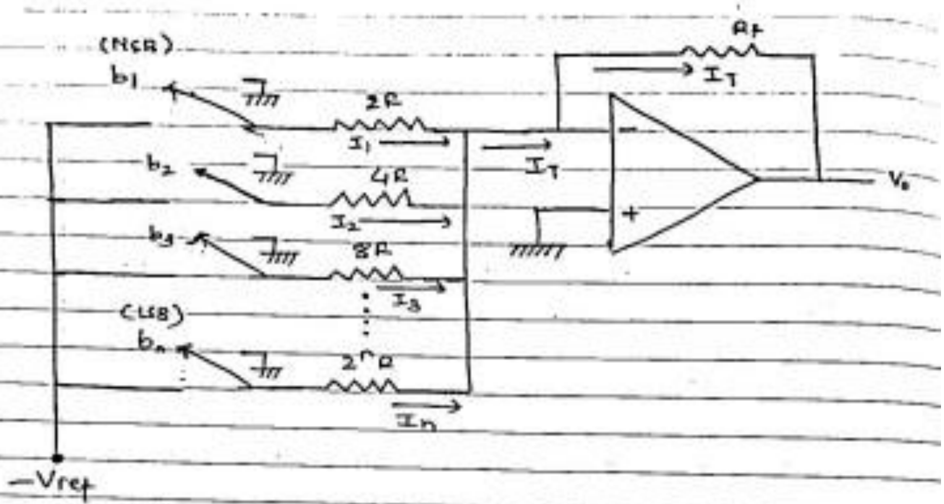
↳ So when, input bit (b = 0), switch connects to ground.

↳ input bit (b = 1), switch will be connected to Reference Voltage (Vref).

↳ I_1 current pass through 2R & so on...

Here we can take (+V ref) also.

Diagram:-



Here, for ON-switch, corresponding current $I = \frac{-V_{ref}}{R}$

for OFF-switch $I = 0$

→ Here, Due to high impedance of op-amp, summing current will flow through R_f. (i.e. No current will pass in op-Amp).

so,

$$I_T = I_1 + I_2 + I_3 + \dots\dots I_n.$$

↳ As per the O/P equation of op-Amp.

$$V_o = -I_T \, R_f$$

$$= -\left(I_1 + I_2 + I_3 + \dots + I_n\right) \cdot R_f$$

↳ we know that by ohm's law $\boxed{I = \frac{V}{R}}$

so, $$V_o = -\left(\frac{-V_{ref}}{2R} \cdot b_1 + \frac{-V_{ref}}{4R} \cdot b_2 + \dots \frac{-V_{ref}}{2^n R} \cdot b_n\right) \cdot R_f$$

↳ Here, $b_1, b_2, b_3 \dots b_n$ (Input bits) are multiplied to Indicate whether current is 0 or has some value of voltage.

↳ $$V_o = \frac{V_{ref} \cdot R_f}{R}\left(\frac{b_1}{2} + \frac{b_2}{4} + \frac{b_3}{8} + \dots + \frac{b_n}{2^n}\right).$$

↳ Assume, $R_f = R$,

$$V_o = V_{ref}\left(\frac{b_1}{2} + \frac{b_2}{4} + \frac{b_3}{8} + \dots + \frac{b_n}{2^n}\right)$$