DCCLD

Date / /
Page No.

Number system

Binary	Base	
	2	0,1
octal	8	0-7
decimal	10	0-9
exadecimal	16	0-9 a-f

★ Binary → decimal

$(100)_2 = (4)_{10}$

$$\begin{matrix} 1 & 0 & 0 \\ \times_{2} & \times_{1} & \times_{0} \\ 2^{2} & 2^{1} & 2^{0} \end{matrix} \quad + \quad + \quad = \quad 4+0+0$$

$$= 4$$

$(100)_2 = (4)_{10}$

$(10010)_2 = (18)_{10}$

$$\begin{matrix} 1 & 0 & 0 & 1 & 0 \\ \times & \times & \times & \times & \times \\ 2^{4} & 2^{3} & 2^{2} & 2^{1} & 2^{0} \end{matrix} \quad + \quad + \quad + \quad + \quad = \quad 16+0+0+2+0$$

$$= 18$$

* Decimal → Binary

$(6)_{10} = (110)_2$

2	6
2	3 - 0
	1 - 1

LCM

110

DL
BM

6

Date / /
Page No.

* $(14)_{10} = (1110)_2$

2	14
2	7 — 0
2	3 — 1
	1 — 1

check how it right

1 1 1 0

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

$= 1110$

$= 8 + 4 + 2$

$= 14$

* $(156894)_{10} = (\ )_2$

Check

$(156)_{10} = (\ )_2$

2	156
2	78 — 0
2	39 — 0
2	19 — 1
2	9 — 1
2	4 — 1
2	2 — 0
	1 — 0

$(156)_{10} = (10011100)_2$

10011100

$(10.01)_2 = (2.75)_{10}$

$2^1 + 2^0 \quad 2^{-1} \quad 2^{-2} = 2 + \frac{1}{2} + \frac{1}{4}$

$2 + 0.25 = 2.25$

Date / /
Page No.

* $(11.101)_2 = (3.625)_{10}$

$2^1 + 2^0 + 2^{-1} \cdot 2^{-2} \cdot 2^{-3}$

$2 + 1 + 0.5 + 0 + 0.125$

$= 3.625$

$0.1 = \frac{1}{2} + \frac{1}{8} = 0.5 + 0.125 = 0.625$

$2^{-1} \quad 2^{-3}$

* $(6.5)_{10} = (\quad)_2$

$2 \mid 6$

$2 \mid 3 \to 0$

$1 \to 1$

$110.$

$.5 \times 2 = 1.0 \to 1$

$\rightarrow (110.1)$

$(6.5)_{10} = (110.1)_2$

* $(0.6875)_{10} = (\quad)_2$

$.6875 \times 2 = 1.3750 = 1$

$.3750 \times 2 = 0.7500 = 0$

$.7500 \times 2 = 1.5000 = 1$

$.5000 \times 2 = 1.0000 = 1$

$(0.6875)_{10} = (0.1011)_2$

Date / /
Page No.

* $(54.54)_{10} = (\quad)_2$
* $(3/8)_{10} = (\quad)_2$
* $(111.111)_2 = (\quad)_{10}$
* $(100.001)_2 = (\quad)_{10}$

→ (i) $(54.54)_{10} = (\quad)_2$

2 | 54
2 | 27 — 0
2 | 13 — 1
2 | 6 — 1
2 | 3 — 0
1 — 1

.54 × 2 = 1.08	1
.08 × 2 = 0.16	0
.16 × 2 = 0.32	0
.32 × 2 = 0.64	0
.64 × 2 = 1.28	1
.28 × 2 = 0.56	

$(54.54)_{10} = (110110.10001)_2$

* (i) $(3/8)_{10} = (\quad)_2$

(ii) $(0.375)_{10} = (\quad)_2$

.375 × 2 = 0.75	0
.75 × 2 = 1.5	1
.5 × 2 = 1.0	1

[redacted] $= (0.011)_2$
$(0.375)_{10}$

(i) $(3/8)_{10} = (\quad)_2$

Date / /
Page No.

* $(111.111)_2 = (7.875)_{10}$

$1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \rightarrow 1 \times 2^{-1} \quad 1 \times 2^{-2} \quad 1 \times 2^{-3}$

$4 + 2 + 1 + 0.5 + 0.25 + 0.125$

$7 + 0.875 = 7.875$

* $(100.001)_2 = (4.125)_{10}$

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

$4 + 0 + 0 + 0 + 0 + 0.125$

$4 + 0.125 = 4.125$

** Binary to octal

$(110)_2 = (\quad)_8$

$8 = 2^3$ = 3 bit pairing

$1 \quad 1 \quad 0$

$2^2 + 2^1 + 2^0$

$4 + 2 + 0$

$(6)_8$

Date / /
Page No.

* $(11001)_2 = (\quad)_8$

0 1 1 0 0 1

2^2 2^1 2^0 2^1 2^0

$2+1$ $0+0+1$

3 1

$\rightarrow (31)_8$

* $(100111)_2 = (\quad)_8$

0 0 1 0 0 1 1 1 1

2^2 2^1 2^0 2^2+2^0 $2^2+2^1+2^0$

1 $0+0+1$ $4+2+1$

1 1 7

$(117)_8$

Date / /
Page No.

(a) $(11.01)_2 = (\quad)_8$

(011.010)

0 1 1 . 0 1 0

$2^2 \; 2^1 \; 2^0 \quad 2^2 \; 2^1 \; 2^0$

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

$(3.2)_8$

(b) Octal to Binary (3 bit representation for every digit)

$(64)_8 = (\quad)_2$

6 4 → 110 100 $(110100)_2$

4 3 2 1 0

2 2 2 2 2

8 4 2 1

1 0 0 → 4

4+2=6 // 1 1 0 → 6

(c) $(431)_8 = (\quad)_2$ → $(\quad)_2$

(100011001)

8 7 6 5 4 3 2 1 0

$2^8 \; 2^7 \; 2^6 \; 2^5 \; 2^4 \; 2^3 \; 2^2 \; 2^1 \; 2^0$

4 2 1

001 → 1
011 → 3
100 → 4

Date / /
Page No.

(d) $(84)_8 = (\quad)_2$

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

$8\ 4\ 2\ 1$

$1\ 0\ 0$

$1\ 0\ 0\ 0$

$\therefore (1000100)_2$

wrong Question because 8 cannot in octal only (0-7)

* octal to decimal

$(74)_8 = (\quad)_{10} = (60)_{10}$

$(111100)_2$

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0 = 60$

Date / /
Page No.

Decimal to octal

$(94)_{10} = (\ )_8$

8	94
8	11 → 6
	1 → 3

$(\ )_2 = (\ )_8$

$(136)_8$

Binary to Hexa

$(1001100)_2 = (\ )_{16}$

$16 = 2^4 =$ 4 bit pairing

0 0 1 0 — $x_3\ x_2\ x_1\ x_0$ — $2^3 + 2 + 2^1 + 2^0$

$0 + 0 + 2 + 0$

2

0 1 0 0 — $x_3\ x_2\ x_1\ x_0$ — $2 + 2 + 2^1 + 2$

$0 + 4 + 0 + 0$

4

$(24)_{16}$

Date / /
Page No.

* $(1110.1111)_2 = (\quad)_{16}$

1 1 1 0 . 1 1 1 1

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

$8+4+2+0 \qquad 8+4+2+1$

$14 \qquad 15$

$(0-9) \rightarrow 10 \quad 11 \quad 12 \quad 13 \quad 14 \quad 15$

$A \quad B \quad C \quad D \quad E \quad F$

$(EF)_{16}$

* $(11000001)_2 = (\quad)_{16}$

1 1 0 0 0 0 0 1

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

$8+4+0+0 \qquad 0+0+0+1$

$12 \rightarrow C \qquad 1$

$(C1)_{16}$

* Hexa → binary

$(E39)_{16} = (\quad)_2$

4 bit 4 bit 4 bit rep

Date / /
Page No.

32 16 8 4 2 1

1 1 0 → 14

0 0 1 1 → 3

1 0 0 1 → 9

$(111000111001)_2$

	D	B	0	H
	0	0	0	0
1		1	1	1
2		10	2	2
3		11	3	3
4		100	4	4
5		101	5	5
6		110	6	6
7		111	7	7
8		1000	10	8
9		1001	11	9
10		1010	12	10 - A
11		1011	13	11 - B
12		1100	14	12 - C
13		1101	15	13 - D
14		1110	16	14 - E
15		1111	17	15 - F

Date / /
Page No.

Q1 what is the role of number system in digital electronics.

ans ⇒ Number system plays a fundamental role

1. Representation of data
2. Arithmetic operations.
3. Data storage
4. Addressing and Memory
5. Data transmission
6. logic gate.

Q2 what is the benefit of hexadecimal number system:

ans ⇒
- because it compresses the data.
- Easy to convert
- Usefulness in computing
- Human readability

Q3 What is the application areas of octal numbers.

ans ⇒ It is used in 8 bit system.
Computer Science, Networking, Data Storage, programming languages.

Date / /
Page No.

* Binary Arithmetic

2/3
1-1

Add

A	B	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

carry = 1

1 1 1 0
1 1 0 0

1 1 0 1 0

1 1 0 . 1 1 0
1 0 0 . 0 1 1
1 0 1 1 . 0 0 1

Sub

A	B	sub	carry
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

carry 0=10

borrow -1

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

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

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

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

Multiplication

A	B	*
0	0	0
0	1	0
1	0	0
1	1	1

ex

1 1 0 0
1 0 1

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

1 1 1 1 0 0

Division ÷

A	B	÷
0	0	X
0	1	0
1	0	X
1	1	1

11) 1001011 (0011001

0↓
100
11↓
0011
11↓
000
00↓
01
00↓
11
11
0

quotient → 1100 1

remainder → 0

Q eg → 10) 11100 (110

10↓
010
10↓
000
00
0

quotient → 110

Remainder → 0

Date / /
Page No.

Q 11011) 111001010000 (01
00
11100 1
11011
0111100

101) 1101010 (10101
101
0011
0
110
101
0011
0
110
101
001

remender = 001

Q = 10101

Date / /
Page No.

* $\frac{11010101}{101}$ = Q = 10101, R = 11

```
        10101
   101 ) 11010101 ( 10101
         101
         ----
          0011
          0
          ----
           110
           101
           ----
            0011
            0
            ----
             110
             101
             ----
              0011
              0
              ----
               11
```

R = 11

Q = 10101

* Signed / unfinged numbers

Unfigned numbers :- Unsigned numbers have no sign that means all unfigned binary numbers are positive numbers.

They contain only the magnitude of number

* Signed numbers : Signed numbers contain both sign and magnitude of number and the sign is placed in front of the number

If sign bit = 0 → +ve

If sign bit = 1 ⇒ -ve

↳ Sign magnitude form

↳ 2's complement form

MSB → most significant bit (Left most)
LSB → least significant bit (Right most)

(1) −6 sign magnitude form

↳ 1 110

+ve 6 → 0 110

7 → 0111

−7 → 1111

we don't use

(2) ~~↳ 2's complement form.~~

* The limitations of sign magnitude

it leads to wastage of data bits so that's why we don't use.

(2) 2's Complement form :-

(1) 1's Complement of a number is obtain by complementing all the bits.

1010 1's C → 0101

2's C → +1
0110

Date / /
Page No.

* 2' complement form :- 2' complement form is obtained by adding one to the one's complement.

** -7 2'c 7

$7 \rightarrow 0111$ (111)

1's c $\rightarrow$ 1000

2's c $\rightarrow$ 1000 (+1)

$2^3 2^2 2^1 2^0$

$-2^3+0+0+1$

$-8+1 = -7$

2 | 7

2 | 3 — 1

1 — 1

111

4 bit repr

0111

Note $\rightarrow$ 0 – 7 $\Rightarrow$ 4 bit

8 – 127 $\rightarrow$ 8 bit

128 $\rightarrow$ above $\rightarrow$ 16 bit

** -6

$\hookrightarrow$ 0110

1's c $\rightarrow$ 1001

2's c $\rightarrow$ 1001 (+1.)

1010.

$-2^3 2^2 2^1 2^0$

$-2^3+0+2+0$

$-8+2 = -6$

Date / /
Page No.

-15 = 1111

↳ 8 bit

0000 1111

1's c 1111 0000

+1

2's c 1111 0001

$-2^7+2^6+2^5+2^4+0+0+0+2^0$

$(-128+64+32+16+1) = -15$

4 bit
8 bit
16 bit

Q -21 → answer

0001 0101

1's c → 1110 1010

+1

2's c → 1110 1011

Q -108 binary no

↳ (1101100)

1's c → 0011 0100

→ 1001 0011

2's c → +1

1001 0100

2	108
2	54 – 0
2	27 – 0
2	13 – 1
2	6 – 1
2	3 – 0
	1 – 1

1101100
8 bit
01101100

Date / /
Page No.

* +8 −8

1000

1(000)

111 → 7

→ 1-7 → 4-bit

(1000)

8 → 1(000)

→ 00.001.000.0

1's → 1111 0111

+1.

2's → 1111.1.000

8 - 127 → 8 bit representation

2 | 8
2 | 4 - 0
2 | 2 - 0
1 - 0

* (-131) - B → ~~1(00.000 11)~~ 1(00000011)

↳ 00000000 1000 0011 (16-bit)

1's → 1111111101111100

+1

2's → 1111111101111101

2 | 131
2 | 65 - 1
2 | 32 - 1
2 | 16 - 0
2 | 8 - 0
2 | 4 - 0
2 | 2 - 0
1 - 0

Date / /
Page No.

[redacted]

It is also known as unit distance code

* Gray Codes: The reflected binary code (RBC) also known as gray code is an ordering of binary number systems such that two successive values differ in only one bit.

	B	G
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110

B → G: (binary के पहले bit को same लिख दो उसके बगल वाला से xor करो ((same → 0) (differ → 1))

B → 0 0 1 1

G → 0 0 1 0

G → B

0000 ← 0

0001 ← 1

0011 ← 2

0010 ← 3

0110 ← 4

* Why we use gray code → To [redacted] facilitate (i) error prevention (ii) Use in K-Map (iii) A/D

~~B → G~~

G → B

* $(1100)_G = (\quad)_2$

G = 1 1 0 0

B = 1 0 0 0 → Gray

G → (1100)

B = 1 0 0 0

B = (1000) → Gray

B = (0100)

(0110)

* G (0110)

G = 0 1 1 0

B = 0 1 0 0

B = (0100)

* BCD Code

* BCD: BCD is the system of writing number that assigns a four digit binary code to each digit 0 to 9 in decimal no system.

$(10)_{10} = (\quad)_{BCD}$

(0001 0000)

$(12)_{10} \rightarrow (\quad)_{BCD}$

1 0 → 0010

2 → 0010

(0010 0010) BCD

12
1-0

* what is the need of BCD.

or application

⇒ in digital displays.

Date / /
Page No.

* what is diff b/w BCD & Hexa.

Ans A-f we don't use in BCD only upto (0 to 9).

A-f → forbiddem no in BCD.

A $(f)_{16} = (\quad)_{BCD}$

$(15)_{10} = (\quad)_{BCD}$

$(00010101)_{BCD}$

$(15)_{10} = (\quad)_{BCD}$

~~00010101~~

in binary

then BCD

Assignment

(1) what are Excess-3 and it's application areas.

(2) what are ASCII codes How they are different from unicode.

A Now (1) $(61)_{16} = (\quad)_{10} = (\quad)_{BCD}$

(2) $(1001001-111)_2$

(3) $(-132)_{10} = (\quad)_2$ → binary → BCD → EBCDIC

(4) $(1100010)_2 = (\quad)_8 = (\quad)_{10} = (\quad)_{16}$

Date / /
Page No.

* What is the need of BCD ~~in digital displays~~.

ans: in digital displays, digital calculator weight scales, Gas pumps;

* What is the difference b/w BCD & Hexa Code:

	BCD	Hexa
(i)	Base: 10	Base: 16
(ii)	Number of digits (0-9)	Number of digit (0-9, A-F)
(iii)	range of values 0-99	Range of values 0-FFFF
(iv)	easy for computers to perform arithmetic operations	easy to convert binary to hexadecimal.

(1) $(161)_{16} \rightarrow (\quad)_{10} \rightarrow (\quad)_{BCD}$

8	4	2	1	
0	0	0	1	→ 1
0	1	1	0	→ 6
0	0	0	1	→ 1

$(000101100001)_2$

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

$2^8\ 2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$256 + 64 + 32 + 1$

$(353)_{10} \rightarrow (\quad)_{BCD}$

$(001101010011)_{BCD}$

Date / /
Page No.

* $(F)_{16} = (\quad)_{BCD}$

$(15)_{10} =$

0001 0101

(0001 0101)

2 | 5
2 | 2 - 1
1 - 0

(101)

* $(1001.00 - 111)_2$

$= 1001.00 + 1.111$

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

0 1 1 1 0 1

$= 11101$

64 +
32
4
36
7

2 | 29
2 | 14 - 1
2 | 7 - 0
2 | 3 - 1
1 - 1

(11101)

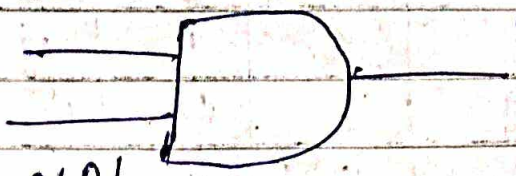
Date / /
Page No.

Logic Gate

Logic gate: A logic gate is an electronic device that makes logical decision based on the different combination.

Types:

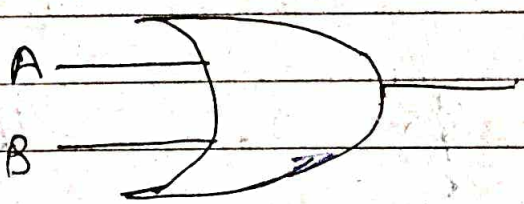
(1) AND - Gate →



A	B	o/p
0	0	0
0	1	0
1	0	0
1	1	1

o/p = A·B

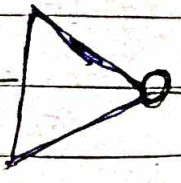
(2) OR Gate →



A	B	o/p
0	0	0
0	1	1
1	0	1
1	1	1

o/p = A+B

(3) Not gate → A

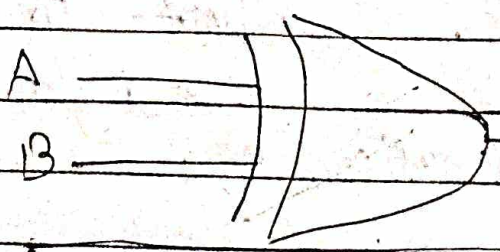


A	o/p
0	1
1	0

o/p = $\bar{A}$

Date / /
Page No.

(4) XOR / EX-OR gate:



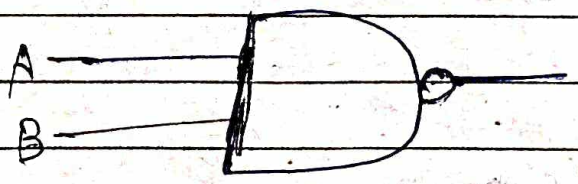
A	B	o/p
0	0	0
0	1	1
1	0	1
1	1	0

$o/p = A \cdot B' + A' \cdot B$

$= A \oplus B$ Shorthand

~~Universal gate~~ universal gate

(1) NAND gate : AND + NOT = NAND

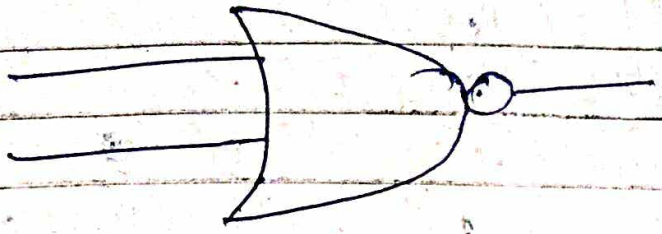


$o/p = \overline{A \cdot B}$ valid

$= A' + B'$

A	B	o/p
0	0	1
0	1	1
1	0	1
1	1	0

(2) NOR → OR + NOT =



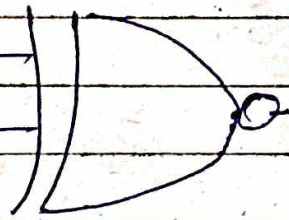
O/P

$O/P = \overline{A+B}$

$= A' \cdot B'$

A	B	O/P
0	0	1
0	1	0
1	0	0
1	1	0

(3) XNOR : A, B



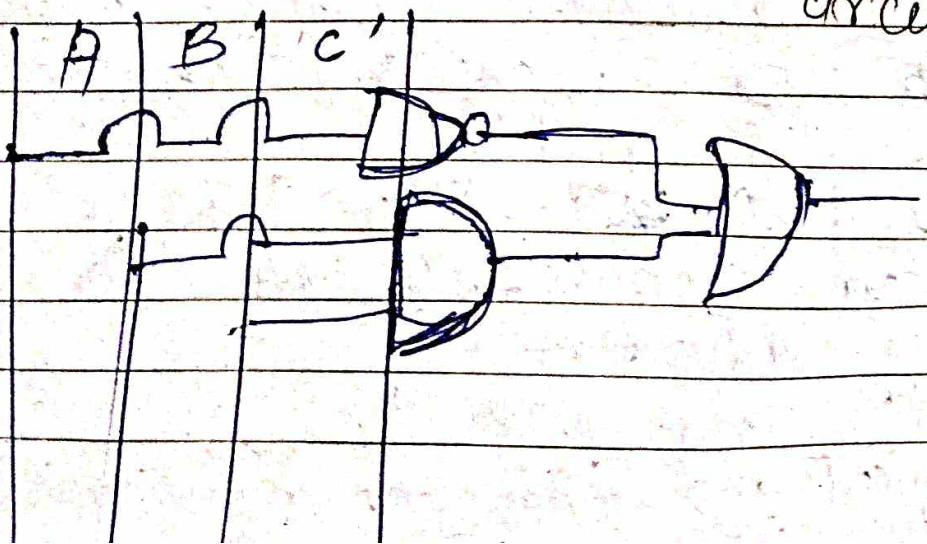
$O/P = A'B' + AB$

$= A \odot B$

A	B	O/P
0	0	1
0	1	0
1	0	0
1	1	1

Make circuit from logic gate.

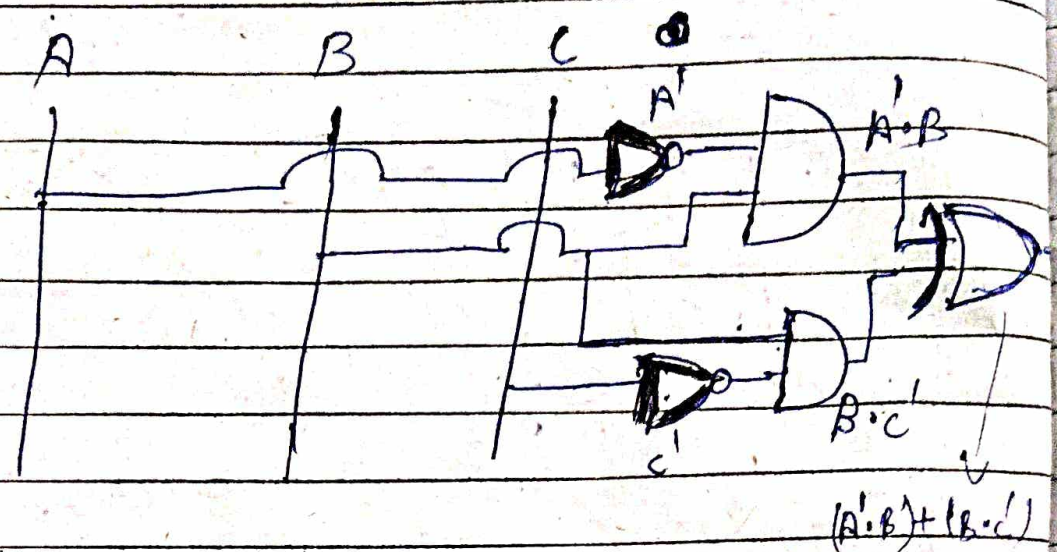
Ans $O/P = A' + B \cdot C$ Using logic gate make circuit



Date / /
Page No.

* $(A' \cdot B) \oplus (B \cdot C')$

Sol^n ⇒



AND
OR
NOT
XOR
XNOR
NAND
NOR

→ Basic gates

→ universal gate

* What is the meaning of universal gates

A gate which can implement any boolean function without using any other gate.

or

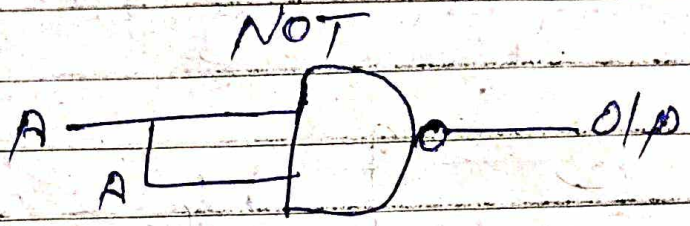
The NAND and NOR gate are called universal gate.

Advantages –

NAND and NOR gates are economical and easier to fabricate (made) than the other gate.

economical and easier to fabricate than the other gates.

(1) NAND
↳ NOT



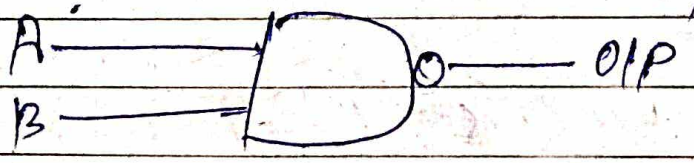
$\overline{A \cdot B} = \overline{A \cdot A} = \overline{A} = A'$

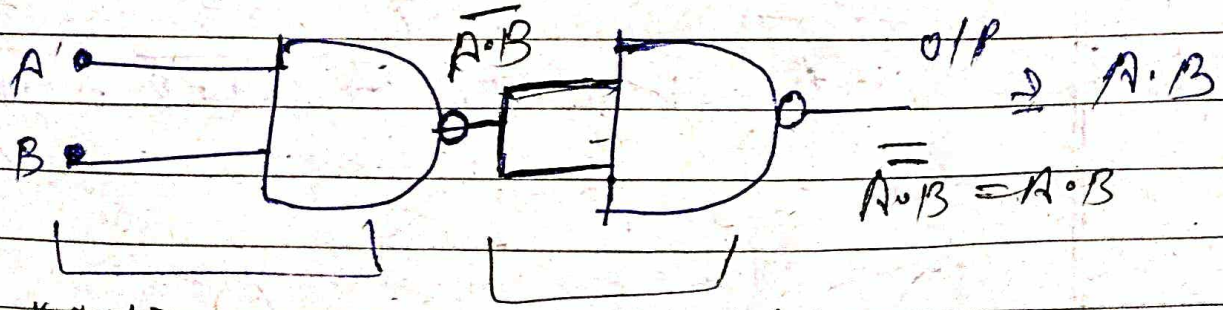
$AND = A \cdot B$, $NAND = \overline{A \cdot B}$

(2) NAND to AND ← MST

AND + NOT = NAND

NAND + NOT = AND





Date / /
Page No.

(3) NAND to OR

OR = A+B

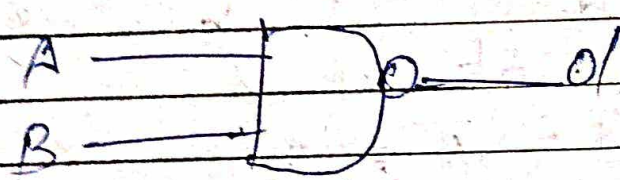
NAND = $\overline{A \cdot B}$

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

$\overline{A+B}$

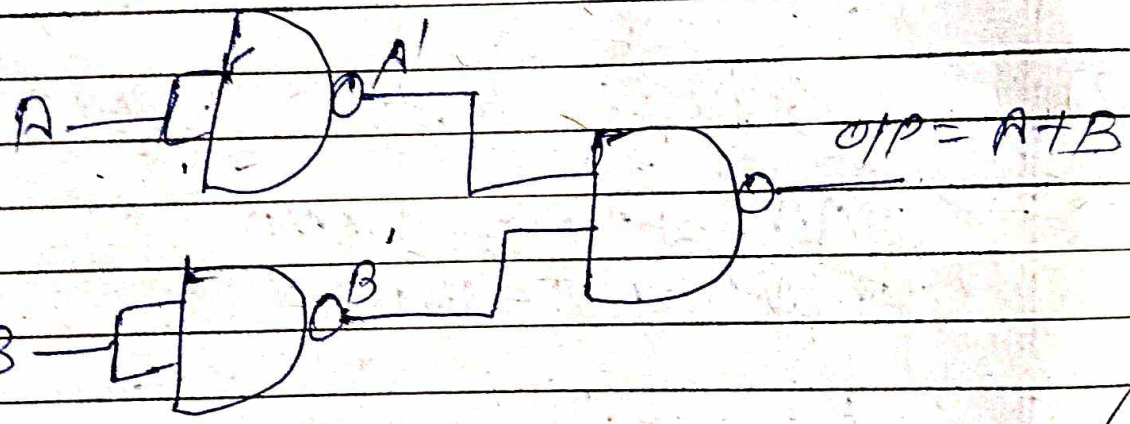
(·) का complement (+)

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



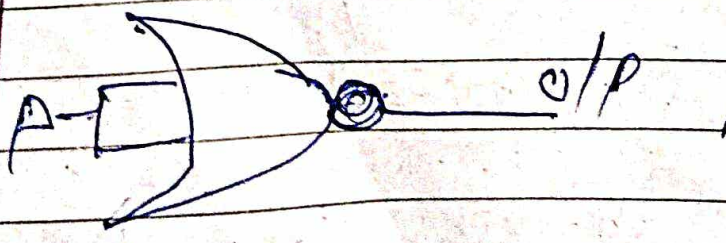
$\overline{A'+B'}$

$A+B$



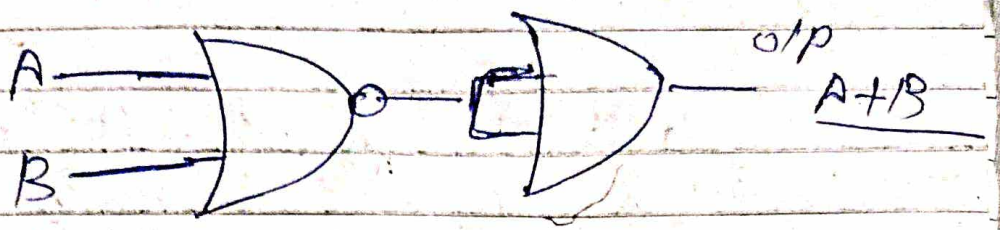
(4) NOR to NOT

$NOR = \overline{A+B} = \overline{A} \cdot \overline{B} = \overline{A+B} = \overline{A}$



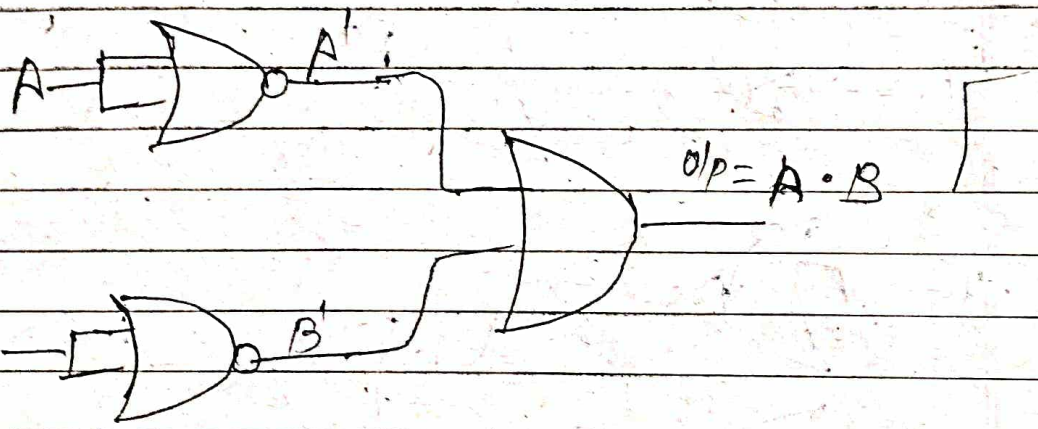
Date / /
Page No.

(5) NOR → OR $\overline{A+B} \Rightarrow A+B$



(6) NOR → AND

$A' \cdot B' \rightarrow A \cdot B$

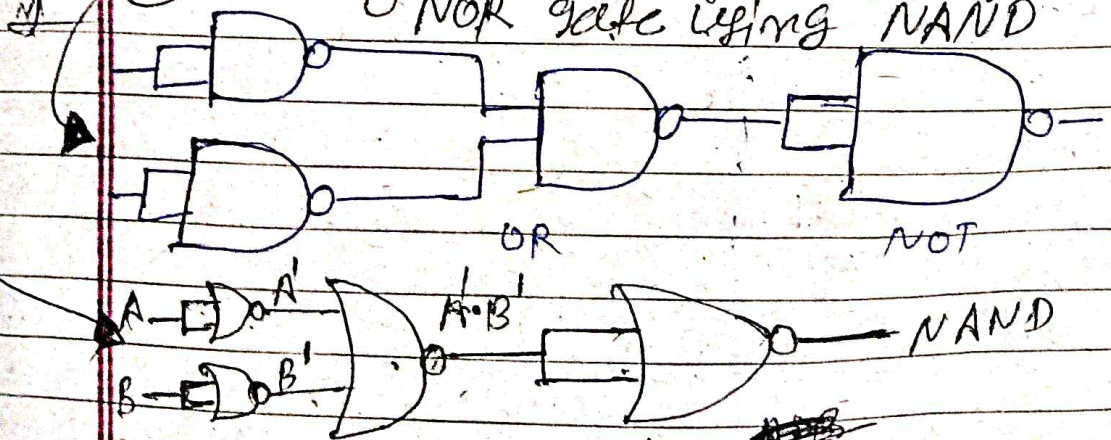


(7) NOR → NAND

AND + NOT = NAND

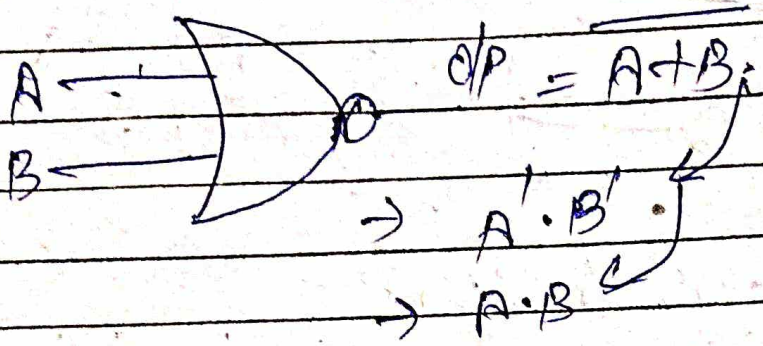
H.W How to create XOR, XNOR gate using universal gates.

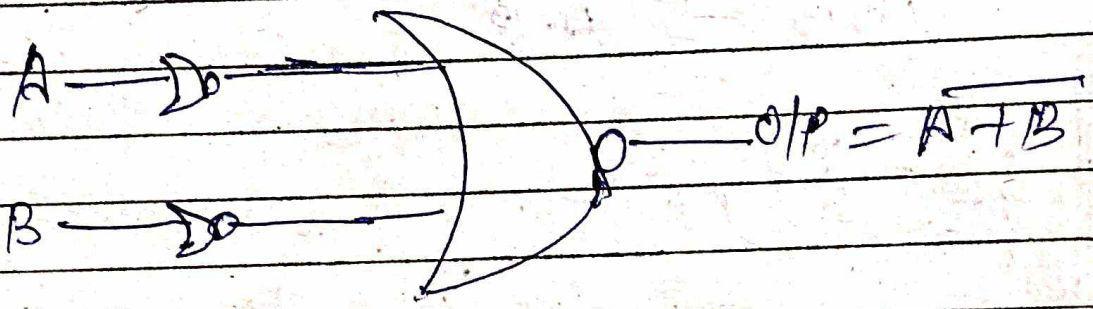
(8) NOR gate using NAND

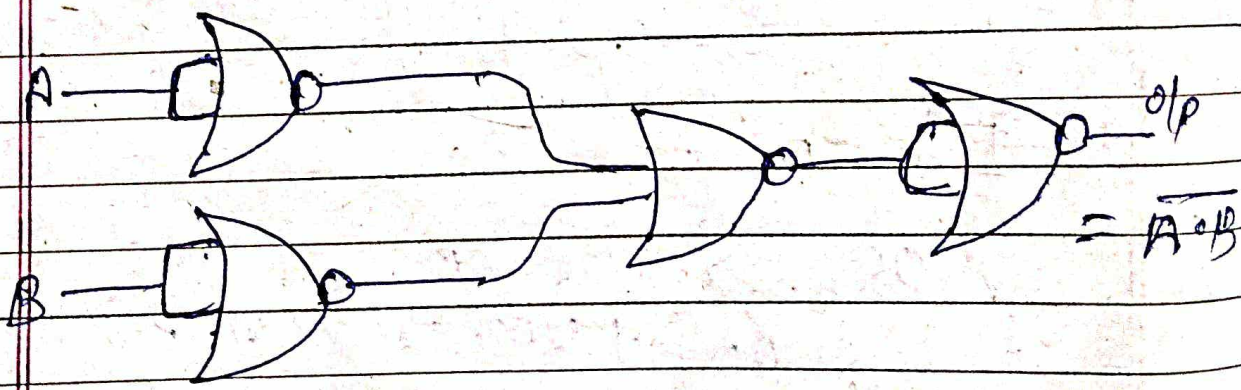


Date / /
Page No.

(7) NAND using NOR



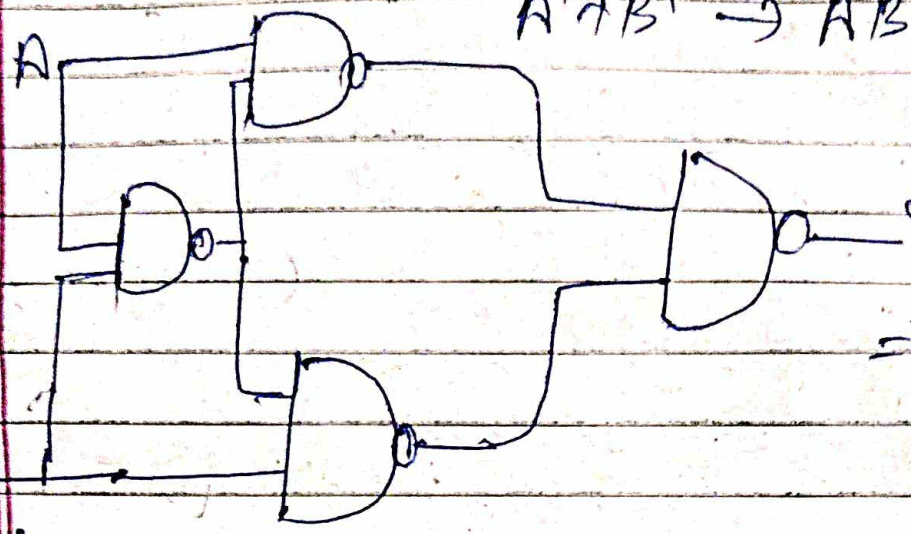




Date / /
Page No.

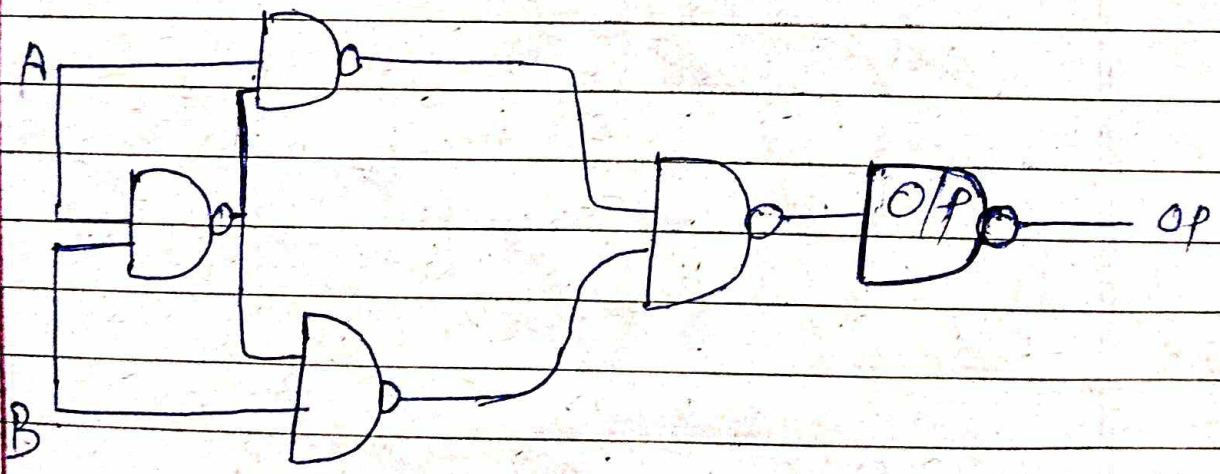
XOR using NAND

$A'+B' \rightarrow A'B + AB'$



XNOR using NAND

add in upper diagram only.



Date / /
Page No.

Boolean Algebra

* What is boolean algebra: It is use to analysis and simplefy the digital circuits, It use only the binary numbers 0 and 1 that's why something it is also called binary algebra.

boolean laws: (i) Commutative laws
- (a) $A \cdot B = B \cdot A$
- (b) $A + B = B + A$

(ii) Associative: (a) $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
- (b) $(A + B) + C = A + (B + C)$

(iii) Distributive: (a) $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$
- (b) $A + (B \cdot C) = (A + B) \cdot (A + C)$

(iv) AND law → (a) $A \cdot 0 = 0$
- (b) $A \cdot 1 = A$
- (c) $A \cdot A = A$
- (d) $A \cdot A' = 0$

(v) OR law: (a) $A + 0 = A$
- (b) $A + 1 = 1$
- (c) $A + A = A$
- (d) $A + A' = 1$

(vi) Double Inversion law:

$(A')' = A$

De morgan's Theorem:

① $(A+B)' = A' \cdot B'$

② $(A \cdot B)' = A' + B'$

Ans How to simplify and minimise boolean expression using boolean law:

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

$= A + AC + AB + BC$

$= A(1+C) + AB + BC$

$= A + AB + BC$

$= A(1+B) + BC$

$= A(1) + BC$

$= A + BC$

(4 Marks)

MF :- $F = A'B + BC' + BC + AB'C'$ [4M]

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

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

$F = BC(A'+1) + AB'C'$

$= B + AB'c' \Rightarrow B + (AB'c')$

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

↓ ⇒ according to distributive law

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

$= B + Bc' + AB + Ac'$

$= B(1+c') + AB + Ac'$

↓ — according to — OR

$= B(1) + AB + Ac'$

$= B(1+A) + Ac'$

↓ according to — OR law

$= \boxed{B + Ac'}$

* $f = abc'd' + abc'd + abcd + ab'cd + abcd' + ab'cd'$

one → $AB + Ac$

$abc'(d+d') + acd(b+b') + acd'(b+b')$

↓ ↓ ↓

$= abc' + acd + acd'$

$= abc' + acd + acd'$

$= abc' + ac(d+d')$

$= abc' + ac \Rightarrow ac + abc'$ distributive law

$(ac+a)(ac+b)(ac+c') \Rightarrow a(c+1)(ac+b)(ac+c')$

↓

$a(ac+b)(ac+c') \Rightarrow (aac+ab)(ac+c') = (ac+ab)(ac+c')$

$ac\cdot ac + acc' + abac + abc' \Rightarrow ac + 0 + abc + abc'$

$\Rightarrow ac + ab(c+c') \Rightarrow ac + ab = \boxed{ab + ac}$

↓ 1

Q# $f = abc'd' + abc'd + abcd + ab'cd + abcd' + ab'c'd + ab'c'd$

ans ⇒ $AB + AC + AD$

~~$f = abc'd' + abc'd + abcd + ab'cd + abcd' + ab'c'd$~~

~~$f = abc' +$~~

$f = abc'd' + abc'd + abcd + ab'cd + abcd' + ab'cd + ab'c'd$

$f = abc'd' + abc'd + abcd + abcd' + ab'cd + ab'cd + ab'c'd$

$f = abc' + acd + acd' + ab'c'd$

$f = abc' + ac(d + d') + ab'c'd$

$f = abc' + ac + ab'c'd$

$f = ac + abc' + ab'c'd$

$= c'(b + b'd) + ac$ $\left\{ \because b + b'd = b + d \right.$

$= ac'(b + d) + ac$

$= abc' + ac'd + ac$

$= a(bc' + c) + ac'd$

$= a(b + c) + ac'd$

$= ab + ac + ac'd$

$= //ab + a(c + c'd)$

$= ab + a(c + d)$

$= ab + ac + ad$

(A) What are SOP and POS forms? How to convert one from to another. (SOP/POS)

Date / /
Page No.

42

* Canonical and standard forms:
A boolean function can be written using canonical forms that is either (SOP) Some of products or (POS) : Product of sum.

$F = x'y'z + xyz' + xy'z'$ ✓
001 110 000

SOP → $= \sum(1, 6, 0)$

SOP ←

POS → x y z

minm → 0 0 0 → 0

max → 1 1 0 → 7

$POS(2, 3, 4, 5, 7)$

$\Pi(0, 2, 3, 4, 5, 7)$

* SOP → Σ → m → minterm → (x'y'z) → 1 − A, 0 → A'

* POS → Π → M → Maxterm → 1 → A', 0 → A

$(x+y').(x+z')$

A	B	m	M
0	0	A'+B'	A+B
0	1	A'+B	A+B'
1	0	A+B'	A'+B
1	1	A+B	A'+B'

$F = \sum(0,1,4)$

SOP = 000 001 100

= 

SOP = a'b'c' + a'bc' + ab'c' → (economical SOP)

(→ canonical form because all terms)

a'b'c' → standard SOP

* $F = \sum(4,5,2,8)$

POS = ?

SOP = (0100 0111 1101 1100)

2	5	2	9
2	2 -	1	4 - 1
2	1 - 1	2	2 - 0
		1	1

2 | 8
2 | 4 - 0
2 | 2 - 0
1

	a	b	c	d
Min	0	0	0	0 → 0
Max	1	1	1	1 → 15

∴ ∴ ∴

POS = Π (0, 4, 5, 6, 7, 10, 11, 12, 13, 14)

pos?

$f = \pi(4,5,2,8)$ 0100 0010 0010 1000

abcd pos = $(a+b'+c+d)\cdot(a+b'+c+d')\cdot(a+b+c'+d)\cdot(a'+b+c+d')$

Q. $f = m_1 + m_4 + m_5 + m_7$

pos = ?

$f = \Sigma(1,4,5,7)$ ⟶ pos → π

$\Rightarrow \pi(0,2,3,6)$

↓

pos = 0000 0010 0011 0110

$f = abc \cdot ab'c \quad ab'c' \quad a'b'c$

$f = (a+b+c)\cdot(a+b'+c)\cdot(a+b'+c')\cdot(a'+b'+c)$

Q. $f = xy + xz + yz$ Canonical POP?

↳ xyz

$= xy(z+z') + xz(y+y') + yz(x+x')$

$= xyz + xyz' + xyz + xy'z + yzx + yzx'$

$= xyz + xyz' + xyz + xy'z + xyz + x'yz$

$= xyz + xyz' + xy'z + x'yz$ → canonical form

Date / /
Page No.

(4) $F = (a'+b+c)(b'+c+d')$ ..... convert canonical form POS?

a b c d

$F = (a'+b+c+d\cdot d')(b'+c+d'+a\cdot a')$

$a'+b+c+d\cdot d' = (a'+b+c+d)(a'+b+c+d')$

A B·C

by distributive law $(A+BC) = (A+B)(A+C)$

$(b'+c+d'+a\cdot a') = (b'+c+d'+a)(b'+c+d'+a')$

A B·C

ans → $(a'+b+c+d)(a'+b+c+d')(b'+c+d'+a)(b'+c+d'+a')$

* Commutative law: Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.
① $A \cdot B = B \cdot A$ ② $A + B = B + A$

* Associative law: This law states that the order in which the logic operations are performed is irrelevant as their effect is the same.
① $(A \cdot B) \cdot C = A \cdot (B \cdot C)$ ② $(A + B) + C = A + (B + C)$

* Distributive law: $A \cdot (B + C) = A \cdot B + A \cdot C$
$A + (B \cdot C) = (A + B) \cdot (A + C)$

* AND law: These laws use the AND operation Therefore they are called as AND laws.
① $A \cdot 0 = 0$ ② $A \cdot 1 = A$
③ $A \cdot A = A$ ④ $A \cdot A' = 0$

* OR law: These laws use the OR operation Therefore they are called as OR laws.
① $A + 0 = A$ ② $A + 1 = 1$
③ $A + A = A$ ④ $A + A' = 1$

* Inversion law: This law uses the NOT operation. The inversion law states that double inversion of a variable result in the original variable itself.
$(A')' = A$

Date / /
Page No.

(PYQ) De Morgan's theorem: This theorem states that the complement of a and product of two or more variables ~~or complement of a OR product of two or more variables~~ is equal to the sum of the complements of those variables.

① $(A+B)' = A' \cdot B'$

② $(A \cdot B)' = A' + B'$

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

① proof: $L.H.S = (A+B)'$

$R.H.S = A' \cdot B'$

① let $A=0, B=0$

L.H.S → $(0+0)' = 0' = 1$

R.H.S → $(0' \cdot 0') = (1 \cdot 1) = 1$

② let $A=0, B=1$

L.H.S = $(0+1)' = 1' = 0$

RHS = $(0' \cdot 1') = 1 \cdot 0 = 0$

③ let $A=1, B=0$

LHS = $(1+0)' = 1' = 0$

RHS = $(1' \cdot 0') = (0 \cdot 1) = 0$

④ let $A=1, B=1$

LHS = $(1+1)' = 1' = 0$

RHS = $(1' \cdot 1') = (0 \cdot 0) = 0$

② LHS = $(A \cdot B)'$

RHS = $A' + B'$

① let $A=0, B=0$

LHS: $(0 \cdot 0)' = 0' = 1$

RHS: $0' + 0' = 1 + 1 = 1$

② let $A=0, B=1$

LHS: $(0 \cdot 1)' = 0' = 1$

RHS: $0' + 1' = 1 + 0 = 1$

③ let $A=1, B=0$

LHS: $(1 \cdot 0)' = 0' = 1$

RHS: $1' + 0' = 0 + 1 = 1$

④ let $A=1, B=1$

LHS: $(1 \cdot 1)' = 1' = 0$

RHS: $1' + 1' = 0 + 0 = 0$

Date / /
Page No.

Q) What are SOP and POS forms? How to convert one form to another.

→ SOP: In SOP form, a Boolean expression is represented as a sum of multiple product. It is represented by Σ and it is minterm (m).

SOP → Σ → m → minterm → 1 → A, 0 → A'

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

→ POS: In POS form, a Boolean expression is represented as product of multiple sum. It is represented by Π, it is maxterm

POS → Π → M → Maxterm → 1 − A'
0 − A

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

→ SOP to POS conversion

Q. $\Sigma(1, 6, 0)$ 6 → 1 1 0
a b c
min → 0 0 0 → 0

$POS(2, 3, 4, 5, 7)$

$\Pi(2, 3, 4, 5, 7)$ max → 1 1 1 → 7

Q. POS to SOP. $F = \Pi(1, 3, 5)$ 5 → 101 min 0 0 0 → 0

$\Sigma(0, 2, 4, 6, 7)$

max → 1 1 1 → 7

Canonical form

SOP → POS 1 - A'
0 - A

$\Pi_1(2,3,4,5,7)$

010, 011, 100, 101, 111

$ab'c, ab'c', a'bc, a'bc', abc$

Canonical → $(a+b'+c)\cdot(a+b'+c')(a'+b+c)(a'+b+c')\cdot(a+b+c)$

POS → SOP 1 — A
0 → A'

$F = \Sigma(0,2,4,6,7)$

[redacted] 000, 010, 100, 110, 111

$a'b'c', a'bc', ab'c', abc', abc$

Canonical

$F = (a'\cdot b'\cdot c') + (a'\cdot b\cdot c') + (a\cdot b'\cdot c') + (a\cdot b\cdot c') + (a\cdot b\cdot c)$

Date / /
Page No

* K Maping : K maping is the graphical technique of simplifying Boolean expression. K map refers to as 2D truth table. K map is nothing but a different formate of representing the values present in a one-dimensional truth table. K-maps basically deal with the technique of inserting the values of the output variable in cells within a rectangle or square grid according to definite pattern. It works on a formula.

K map - Rules

- Group may not include any cell containing a zero.
- Group may be horizontal or vertical, but not diagonal.
- Group must contain 1, 2, 4, 8 or in 2^n cells
- Each group should be as large as possible
- [16, 8, 4, 2, 1]
- Group may overlap.
- Group may wrap around the table.
- There should be as few groups as possible

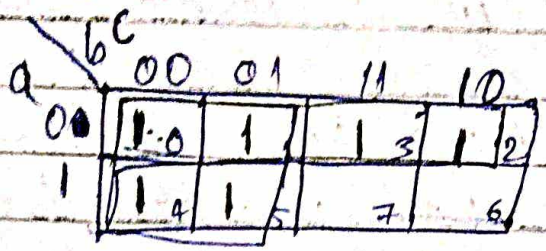
50

Date / /
Page No.

* $F = \Sigma(0, 1, 3, 2, 4, 5)$

(abc)

$2^3 = 8$



2 | 5
2 | 2 - 1
1 - 1

(11)

0 = A'
1 = A

g1

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

a' b'

g2

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

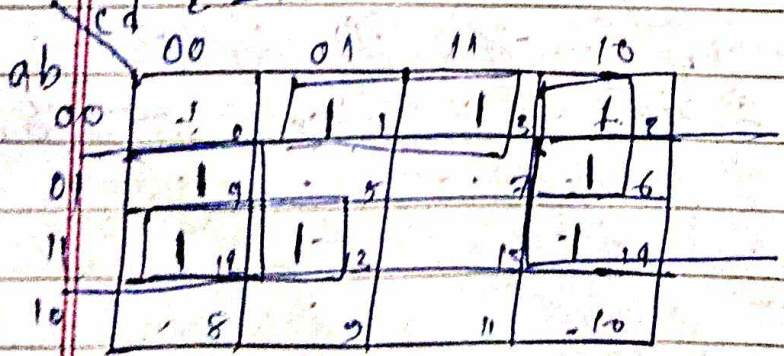
a'

$f = a' + b'$

* $F = \Sigma(1, 3, 2, 6, 4, 12, 13, 14)$

abcd

$2^4 = 16$



2 | 14
2 | 7 - 0
2 | 3 - 1
1 - 1

(1110)

g1

a	b	c	d
∅	1	∅	0
1	1	0	0
∅	1	1	0
1	1	1	0

g2 → b d' ⇒ bd'

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

g3 → a' c d' = a'cd'

a	b	c	d
0	0	∅	1
0	0	1	1

a' b' d

(a'b'd)

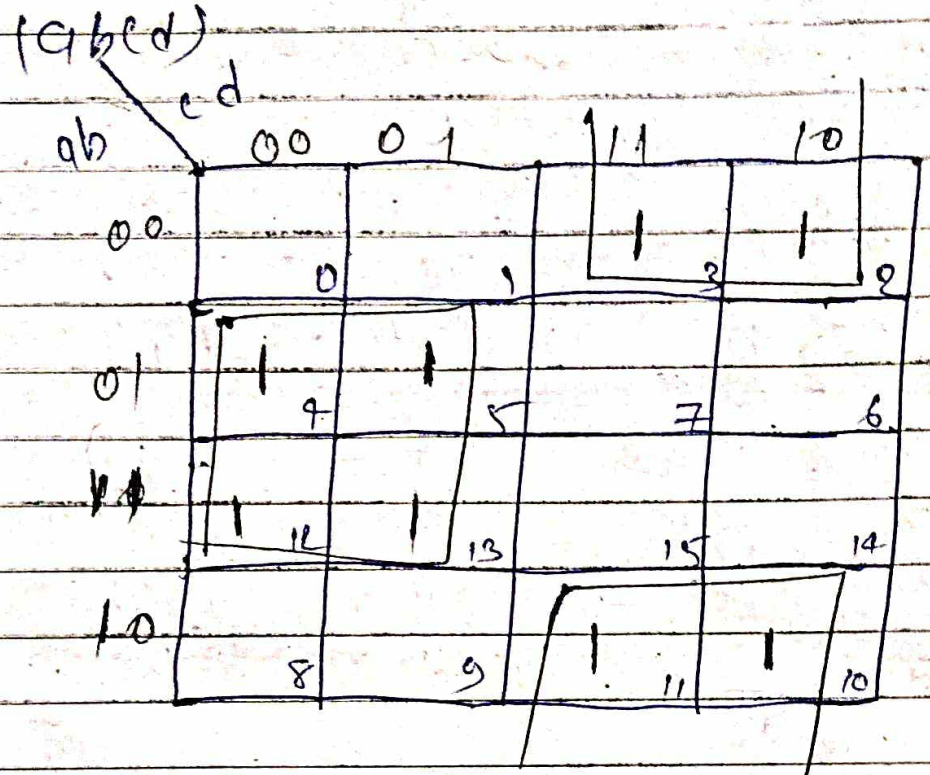
a	b	c	d
1	1	0	0
1	1	0	1

a b c'

(ab(c')

$$f = bd' + a'cd' + a'b'd + abc'$$

Q $F = \Sigma(2, 3, 4, 5, 12, 13, 10, 11)$



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

$b\ c'$

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

$b'\ c$

$$f = bc' + b'c$$

* $f = \Sigma(0,2,3,6,7,8,10)$

2	10
2	8 - 0
2	2 - 1
	1

(1 1 1 0)

(abcd)

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

g1

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

a' c

g2

a	b	c	d
0	0	0	0
0	0	X	0
1	0	0	0
1	0	X	0

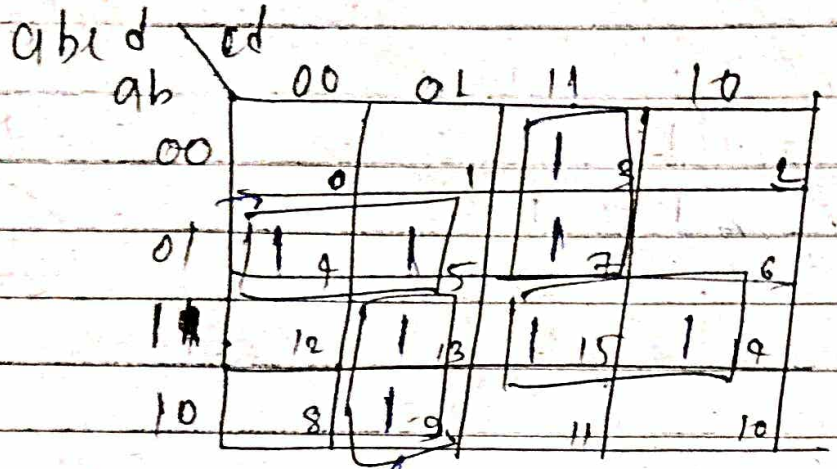
b' d'

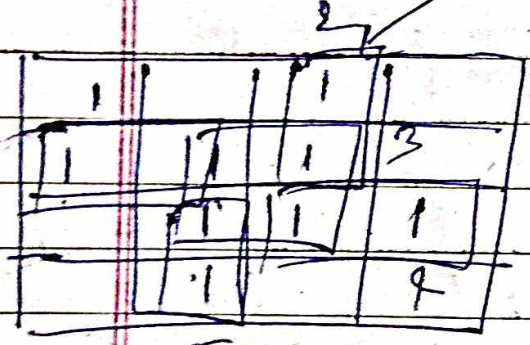
$F = a'c + b'd'$

यहाँ तक SH question था

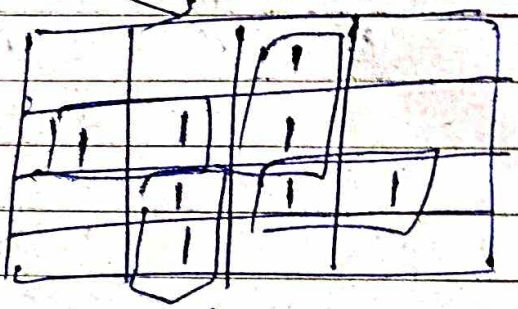
* $f = \Sigma(3,4,5,7,9,13,14,15)$

* $F = \sum(3, 4, 5, 7, 9, 13, 14, 15)$





5 group



only 4 group

मतलब हमारी guard की ज्यादा priority है but कम group जितने बनेगा वही लेंगे

g1

a	b	c	d
0	1	0	x
0	1	0	1

$a' b c'$

g2

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

$a' c d$

g3

a	b	c	d
1	x	0	1
1	1	0	1

$a c' d$

g4

a	b	c	d
1	1	1	x
1	1	1	0

$a b c$

$F = a'bc' + a'cd + acd' + abc$

$F = m(5, 6, 11, 12, 13, 14)$

2	14
2	7 - 0
2	3 - 1
	1 - 1

1110

abcd

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

g1

a	b	c	d
0	~~0~~	0	1
0	~~1~~	0	1

$a'c'd$

g2

a	b	c	d
1	1	0	~~0~~
1	1	0	1

$ab'd$

g3

a	b	c	d
~~0~~	1	1	0
1	1	1	0

bcd'

g4

a	b	c	d
1	0	1	1

$ab'cd$

$F = a'c'd + abc' + bcd' + ab'cd$

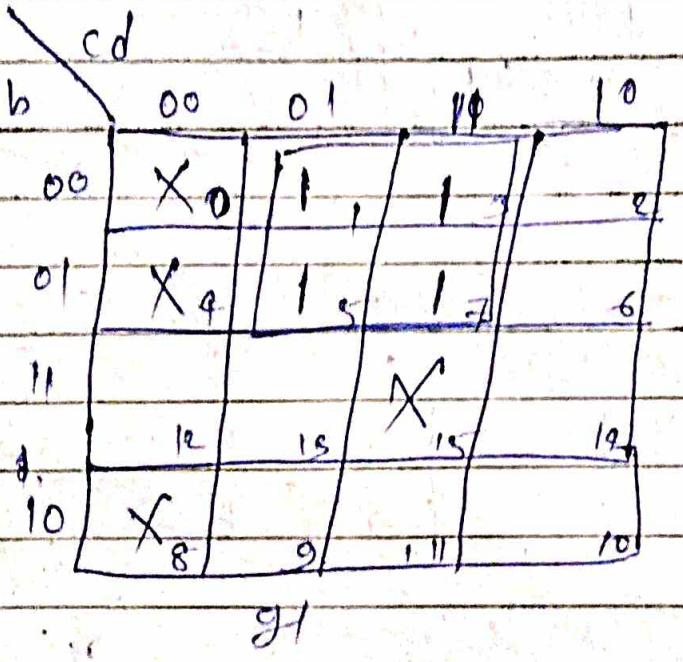
K-map with don't care

K map with don't care : It is an input sequence for which function output does not matter.

$F = \Sigma(1,3,5,7) + d(4,8,15)$

1, 3, 5, 7 → High

4, 8, 15 → High / or / Low (don't care)



g1

a	b	c	d
0	x	0	1
0	0	x	1
0	x	0	1
0	x	x	1

$a' \quad d = a'd$

Date / /
Page No.

(1) $f = m(0,1,3,5,9,13) + d(2,4,6,7)$

ab \ cd	00	01	11	10
00	1 0	1 1	1 3	X 2
01	X 4	1 5	X 7	X 6
11	12	13	15	14
10	8	9	11	10

g1

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

$c'd$

g2

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

$a'c'$

g3

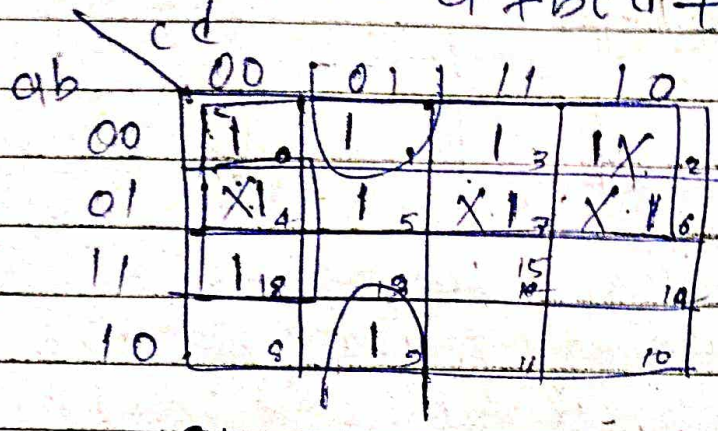
a	b	c	d
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1

$a'd$

$$f = c'd + a'c' + a'd$$

(4) $F = m(0,1,3,5,9,12) + d(2,4,6,7)$

ans → $a' + bc'd' + b'c'd$



g1

a	b	c	d
0	0	0	0
0	0	0	1
0	0	x	1
0	0	1	0
0	1	0	0
0	1	0	1
0	1	1	1
0	1	1	0

a'

g2

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

$bc'd'$

g3

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

$b'c'd$

$$f = a' + bc'd' + b'c'd$$

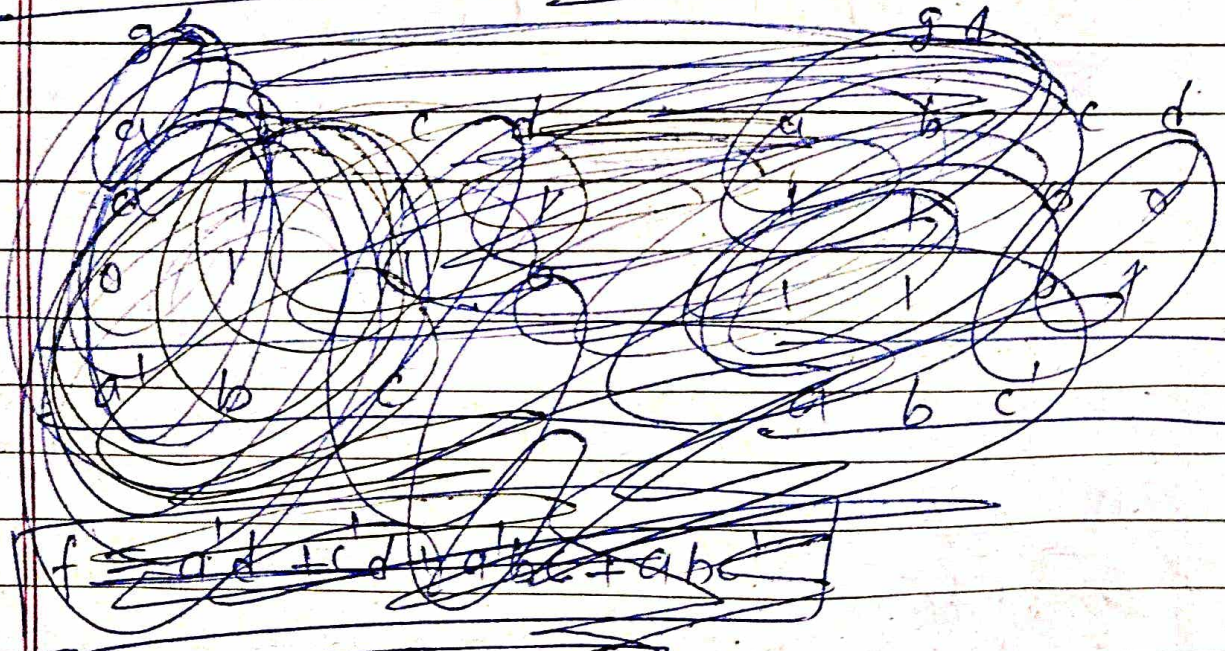
don't cover cross

Date / /
Page No.

* $F = \Sigma(1,3,5,7,9) + d(6,12,13)$

ab \ cd	00	01	11	10
00	0	1 4	1 3	2
01	4	1 5	1 7	X 6
11	X 12	X 13	15	14
10	8	1 9	11	10

g1				g2			
a	b	c	d	a	b	c	d
0	0	0	1	0	0	0	1
0	0	1	1	0	1	0	1
0	1	0	1	1	1	0	1
0	1	1	1	1	0	0	1
a'			d			c'	d



$f = a'd + c'd + a'bc + abc'$

$$\boxed{F = a'd + c'd}$$

Q. $f = \Sigma(1, 3, 4, 6) \rightarrow$ POS

Q. $a'b'c + ab'c + abc' + ac' + b'c$ ⇒ maping using boolean la

Q
- $x + x'y + yz \rightarrow$ canonical SOP?
  std SOP → canonical

Q. $f = \Sigma(0, 1, 4, 5, 6) + d(2, 3)$
minimize using K map.

$\Sigma(0, 1, 4, 5, 6) + d(2, 3)$

a \ bc	00	01	11	10
0	1 (0)	1 (1)	X (3)	1 (2)
1	1 (4)	1 (5)	X (7)	1 (6)

2 | 6
2 | 3 = 0
1 - 1
1 | 0

g1

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

b'

g2

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

c'

$f = b' + c'$

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

Date / /
Page No.

(4) $x'y'z + yz + xz$

$\rightarrow$

$z(x'y' + y + x)$

$z(x + y + x'y')$ by distribution law

$= z((x + y + x')(x + y + y'))$ OR law

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

$= z((y + 1)(x + 1))$

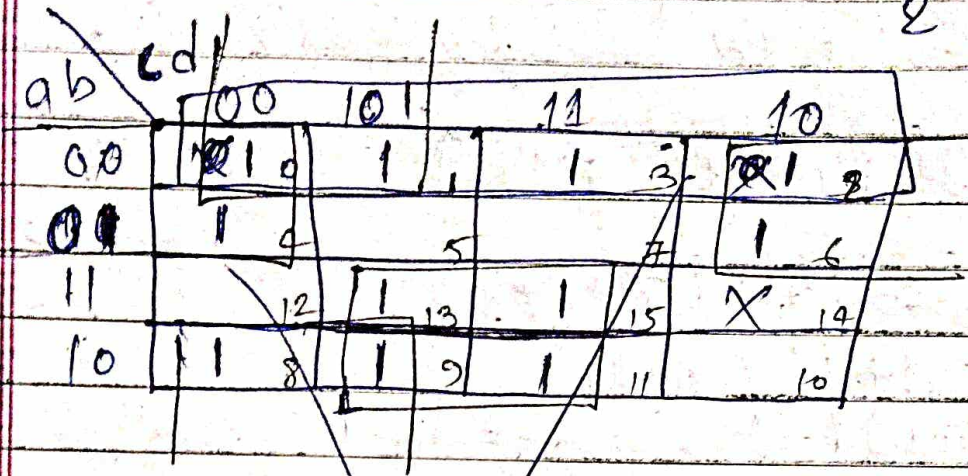
$= z(1)(1)$

$= z$ answer

Date / /
Page No.

* $F(A,B,C,D) = \Sigma m(1,3,4,6,8,9,11,13,15) + \Sigma d(0,2,14)$

$2^4 = 16$



g1

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

a' b'

g2

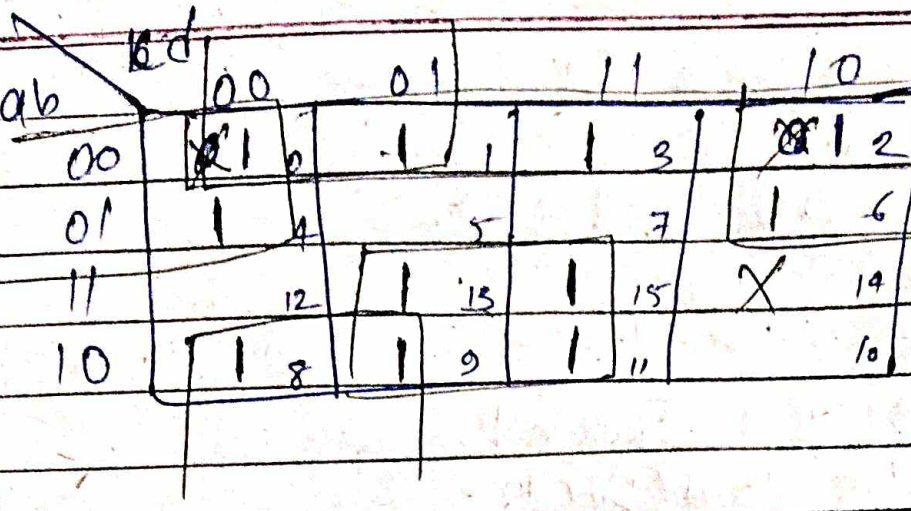
a	b	c	d
1	1	1	0
1	1	1	1
1	0	1	0
1	0	1	1

g3

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

g4

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



g1

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

a' b'

g2

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

a d

g3

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

a' d'

g4

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

b' c'

$$F = a'b' + ad + ad' + b'c'$$

Date / /
Page No.

A $A'BC + AB'C + ABC' + ABC$

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

$A'BC + AB'C + AB$

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

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

$A'BC + AB + AC$

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

$B(A+C) + AC$

$AB + BC + AC$

J $(21.8.6)_{10} \leq ()_{16}$

$16 \overline{)218}$ → D $(DA.99-)_{16}$

→ A

$.6 \times 16 = 9.6 \quad | = 9$

$.6 \times 16 = 9.6 \quad - 9$

① $A'BC + AB'C + ABC' + ABC + ABC + ABC$

$BC(A'+A) + AB'C + ABC + ABC' + ABC$

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

$BC + AC + AB$

② $(ABC)'(A+C)(A+C) \Rightarrow (ABC)'(A + AC + AC + CC)$

$\Rightarrow (ABC)'(A + AC' + AC)$

$(ABC)' A(1 + C' + C) \Rightarrow (ABC)A \Rightarrow (A'B'C')A$

$= AA'B'C' = 0$

Q) $ab + (ac)' + ab'c(ab+c)$

$= ab + (ac)' + 0 + ab'c$

$= a(b+bc) + (ac)'$ $\quad \because A + A'B = A + B$

$a(b+c) + (ac)'$

$ab + ac + (ac)'$

$a \quad a' \quad // a + a' = 1$

$ab + 1 \quad // a + 1 = 1 \qquad a + 0 = 1$

→ 1

Q) ① $A'BC + AB'C + ABC' + ABC$

② $(ABC)'(A+C)(A+C')$

③ $F(P,Q,R) = PQ' + QR' + QR + PQ'R'$

④ $f(x,y,z) = x'y'z' + x'yz' + xy'z + xyz'$

③ $PQ' + QR' + QR + PQ'R'$

$PQ' + Q(R+R') + PQ'R'$

$PQ' + PQ'R' + Q$

$PQ'R' + PQ' + Q \Rightarrow PQ'R' + Q(P'+1) = PQ'R' + Q$ distributive

④ $x'y'z' + x'yz' + xy'z + xyz'$

$x'z'(y'+y) + y(x'z + xz')$

$x'z' + y(x \oplus z)$

$Q + (PQ'R') \Rightarrow (P+Q)(Q+Q')(Q+R')$

$(P+Q)(Q+R')$

$PQ + PR' + Q + QR'$

$PQ + PR' + Q(1)$

$PR' + PQ + Q$

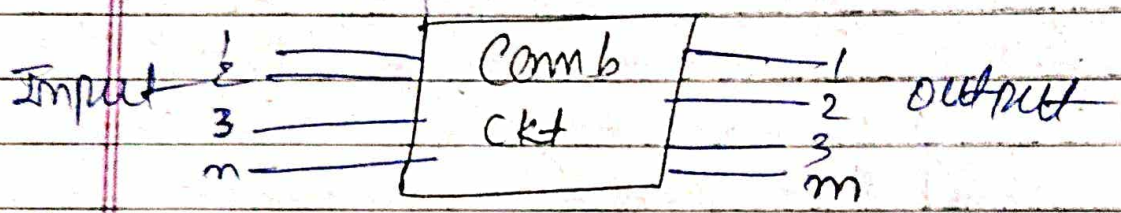
$PR' + Q$ answer

MST-2

Combinational circuits: It is a circuit in which we combine different gates to realise a functions circuits.

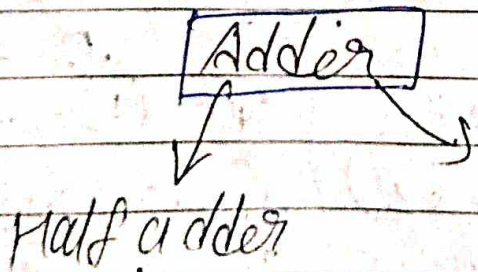
Characteristic of Combinational circuits.

(1) It can have n number of inputs and m number of outputs.



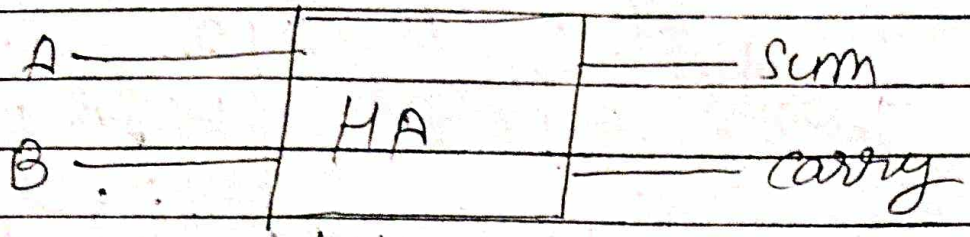
(2) The output of Combinational circuits at any instense depends only on the levels present at the input terminals.

(3) They do not any memory.



* Half adder: It is a Combinational circuits with two inputs and two outputs, It is design to add two single bit binary numbers, It has two

outputs sum and carry.



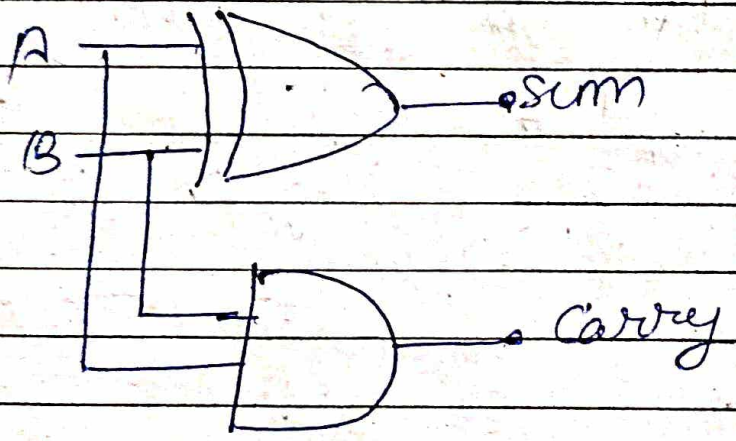
truth table

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

XOR & AND gates

$S = A \oplus B$

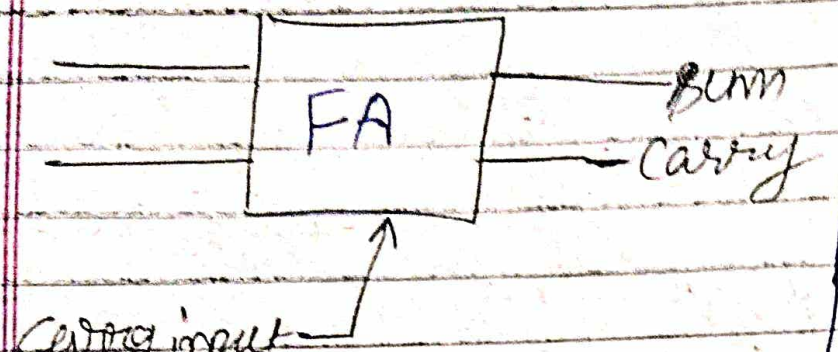
$C = A \cdot B$



* Limitations Of Half Adder: because it can take only two inputs so there is no provision to handle the carry output.

* Full adder: It is a combinational circuit with three input & two output. It is designed to add two single bit binary numbers, it has two output sum and carry, and three input A, B & carryin.

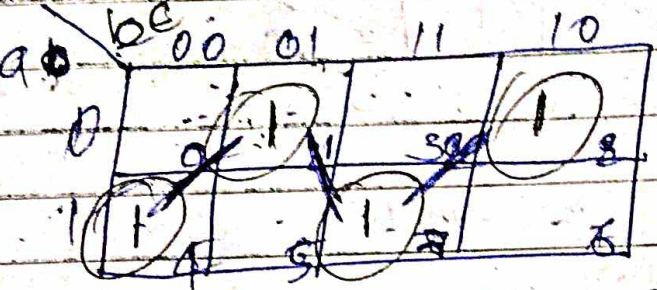
4 Marks ?



A	B	Cin	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

for Sum

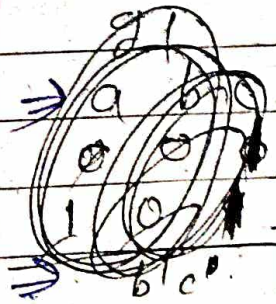
K-map



zig-zag pattern

zig-zag pattern में जितने भी variables होंगे / ~~उनके~~ XOR होगा /

answer $a \oplus b \oplus c$



g1	g2
a b c	a bc
0 0 1	0 1 0
a'b'c	a'bc'

23

$\rightarrow a'b'c' + a'b'c + abc + a'bc'$ → XOR

$b'(ac' + a'c) + b(ac + a'c')$ → XNOR

$= b'(a \oplus c) + b(a \odot c)$

$\rightarrow b'(a \oplus c) + b(a \oplus c)'$

$\Rightarrow a \oplus b$

$a \oplus c \oplus b$ equation for Sum

let $a = a \oplus c$

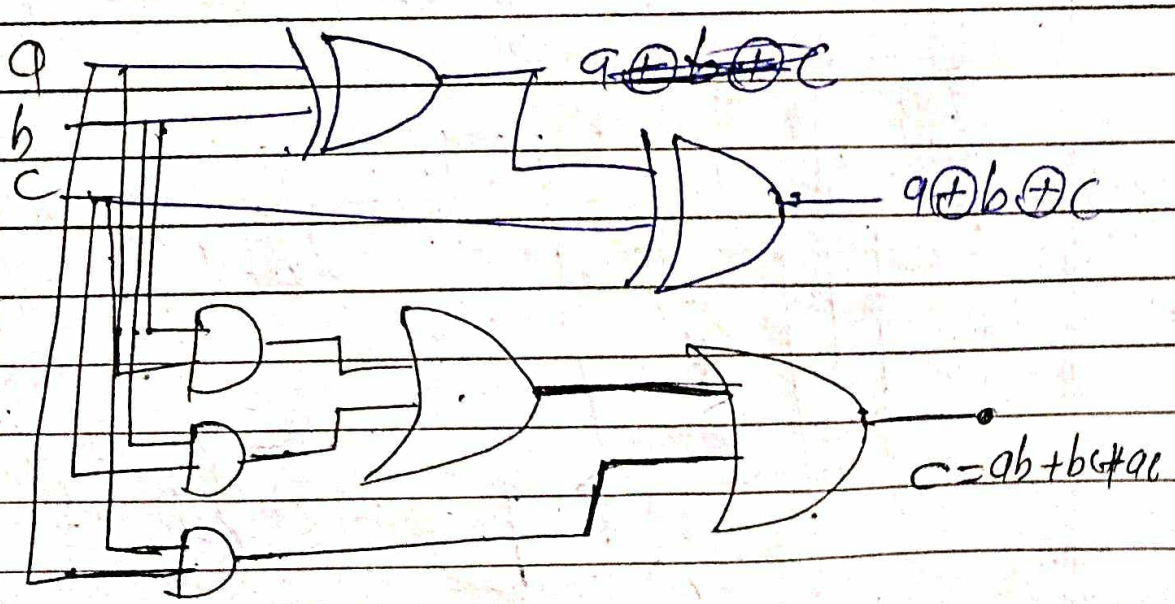
For carry

a \ bc	00	01	11	10
0	0	1	1	2
1	4	5	7	6

g1			g2			g3		
a	b	c	a	b	c	a	b	c
0	1	1	1	0	1	1	1	x
1	1	1	1	x	1	1	1	a
	bc			ac			ab	

$y = ab + bc + ac$ equation for carry output

draw the circuit



Combinational circuits

$S = a \oplus b \oplus c$

$y = ab + bc + ac$ full adder (most use in industry purpose)

$ab + ac(b+b') + bc(a+a')$

$ab + abc + ab'c + abc + a'bc$

$= ab + abc + ab'c + a'bc$

$= ab + abc + ab'c + a'bc$

$= ab + c(ab' + a'b)$

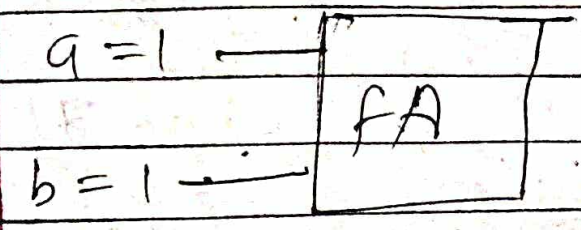
$\Rightarrow ab + c(a \oplus b)$

full adder diagram

* Parallel Adder: It is a digital circuit capable of finding the arithmetic sum of two binary numbers, that are greater than one bit in length. It consists of full adder connected in a chain.

what is the need of parallel adder? needed for high-speed arithmetic operations in digital system.
- Microcontrollers
- GPUs
- CPUs
- DSPs

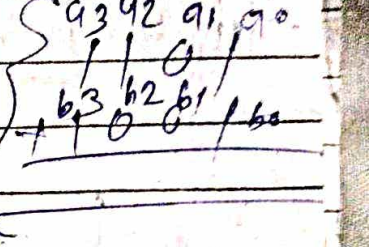
A n bit parallel adder requires n full adder.

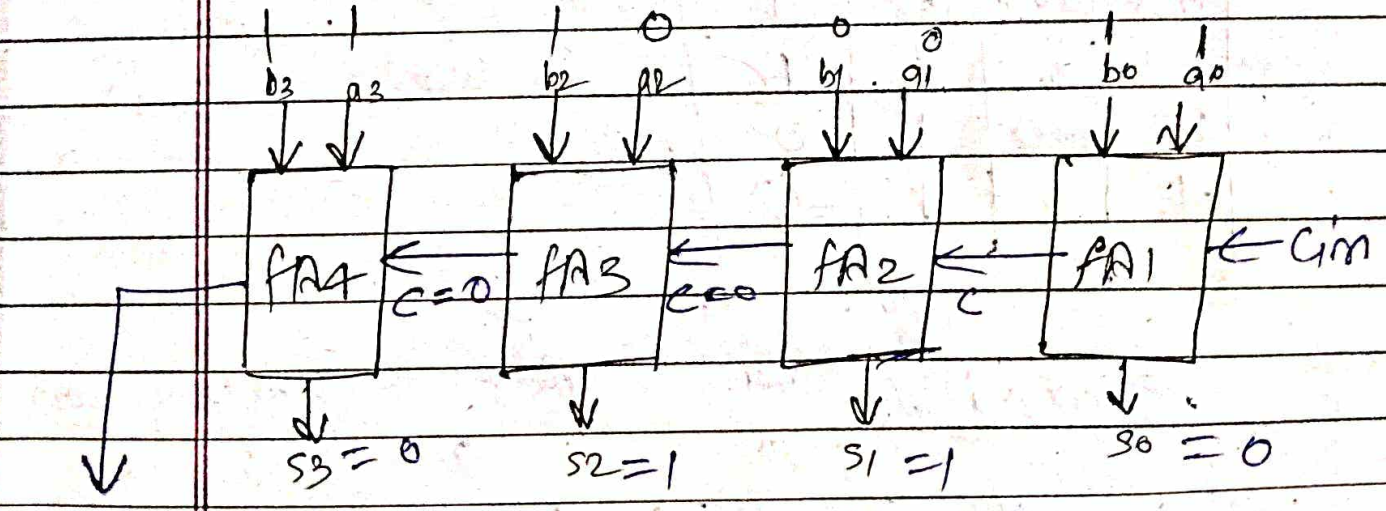


full adder

parallel adder

4 bit parallel adder

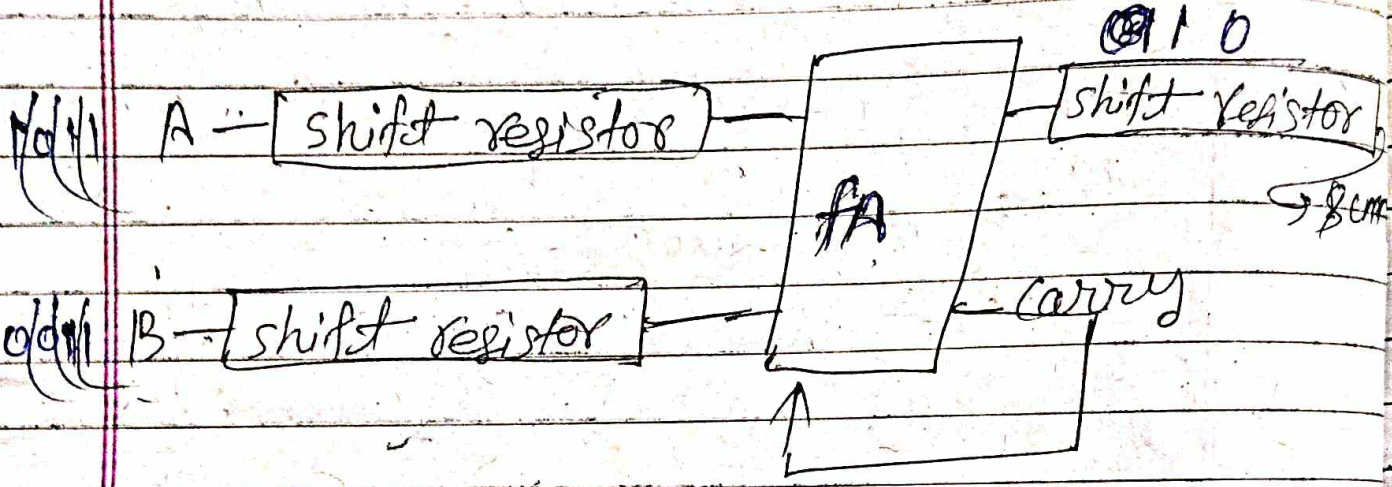




⇒ 10110.

* Serial Adder: A serial adder consists of a single full adder and several shift registers, two shift registors are used to holds the numbers (a & b) and one shift registor is used to store the sum

Advantage: As there is only a single full adder so they are economical, effecient and simple to built.



A	B	S	C
1011	0011	0	1
101	001	1	1
10	00	1	0
1	0	1	0

ये economical होते हैं क्योंकि registers (memory) में तो ज्यादा जगह रहता है, any is (registers) (memory) सस्ता पड़ता है, compare to (adder).

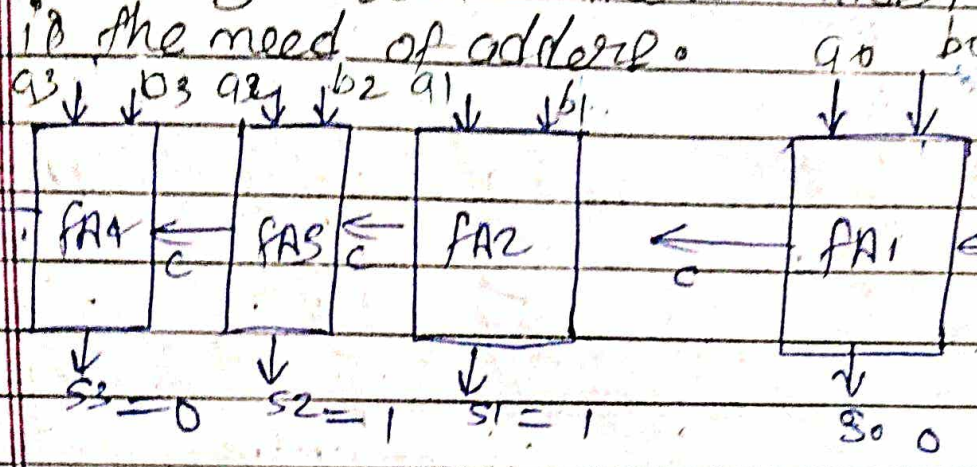
CLA (4 marks)

Carry look ahead adder

Design

Carry look ahead adder and what is the need of adder.

Need:



Carry look ahead adder : a CLA reduces a propagation delay by introducting a complex hardware. It remove the dependencies between the adders and enhance the performance of calculation in advance.

A	B	Cin	S	Cout	
0	0	0	0	0	← No carry generate
0	0	1	1	0	→
0	1	0	1	0	←
0	1	1	0	1	← carry propagate
1	0	0	1	0	
1	0	1	0	1	←
1	1	0	0	1	← carry generate
1	1	1	1	1	←

$P = a \oplus b$

$G1 = a \cdot b$

$C = G1 + Cp$

$C = ab + Cin(a \oplus b)$

$a \oplus b$ → P

ab → G1

~~$G1 + Cin(P)$~~

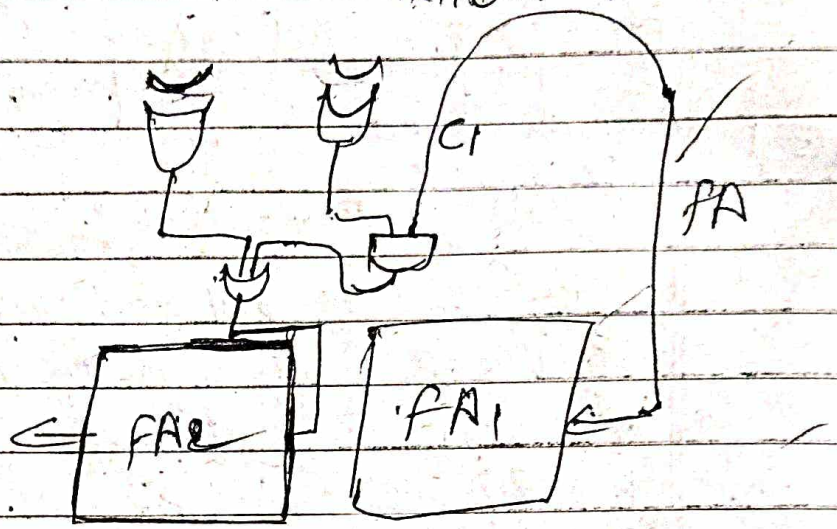
$\rightarrow C = G1 + Cin(P)$

$c_1 = a_0b_0 + c_{in}(a_0 \oplus b_0)$

$$\begin{array}{r} 101 \\ +\ 110\ b_0 \\ \hline \end{array}$$

$c_1 = 10 + 0(\ ) = 1, 0, c_2$

$c_2 = a_1b_1 + c_{in}(a_1 \oplus b_1) = 1, 0, c_2$



$c_1 = a_0b_0 + c_{in}(a_0 \oplus b_0)$

$c_2 = a_1b_1 + c_1(a_1 \oplus b_1)$

Q.NO) What is the advantage of & disadvantage CLA.

a (i) faster, reduce propagation delay.

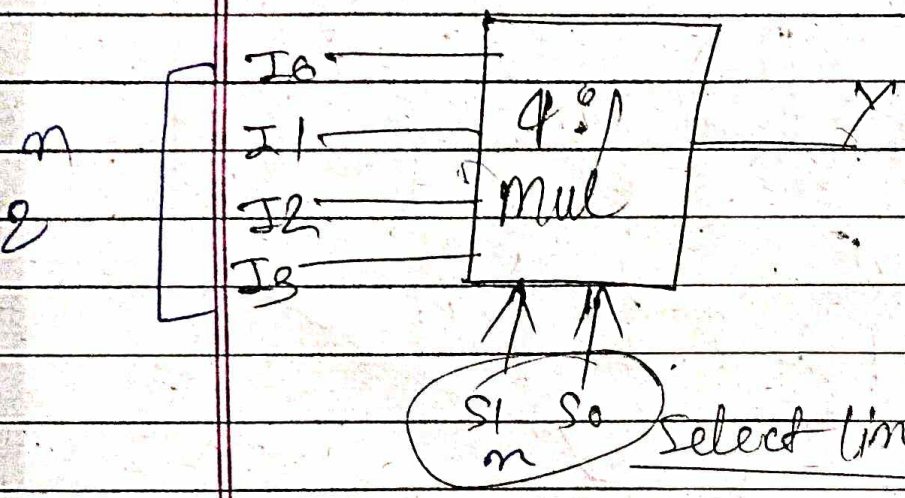
d (ii) costly, complex.

Needs of adders: Adders are essential for performing arithmetic operations on digital computers. They are used in the arithmetic logic unit (ALU) which is the part of the CPU that performs all arithmetic and logical operations. It also use in memory management unit and floating point unit.

Date / /
Page No.

(*) Multiplexer / Demultiplexer
(mux) Demux

Multiplexer :- It is the combinational circuit that has many inputs and one output. It has select lines which are use to select which input line send to the output.



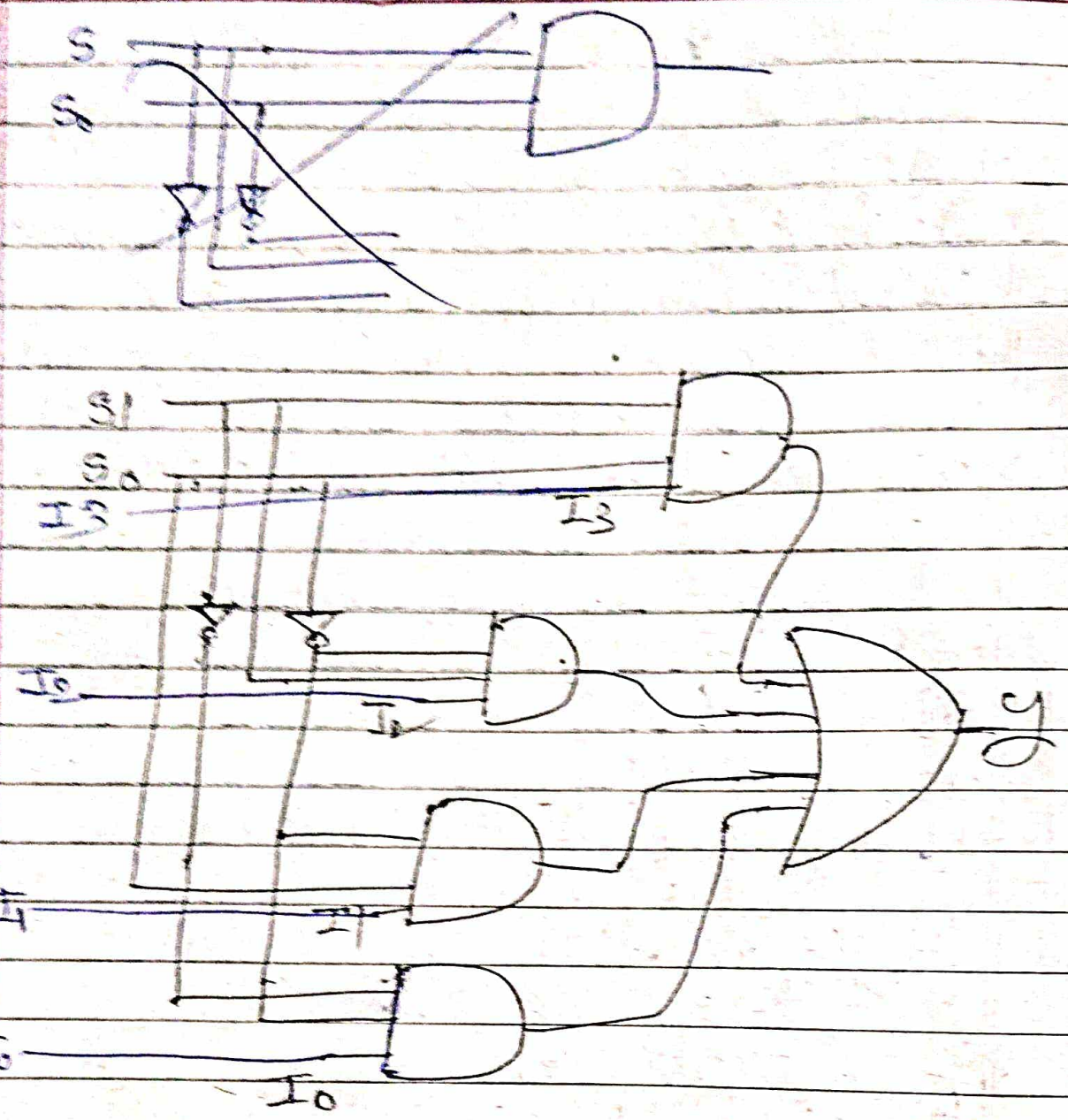
Truth table

S1	S0	y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

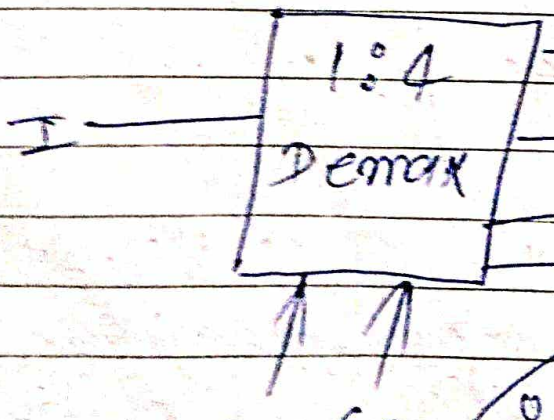
$y =$

$Y = S_1' S_0' \cdot I_0 + S_1' S_0 I_1 + S_1 \cdot S_0' \cdot I_2 + S_1 S_0 \cdot I_3$

Use of Multiplexer: Communication system, computer system, industrial systems, consumer electronics.



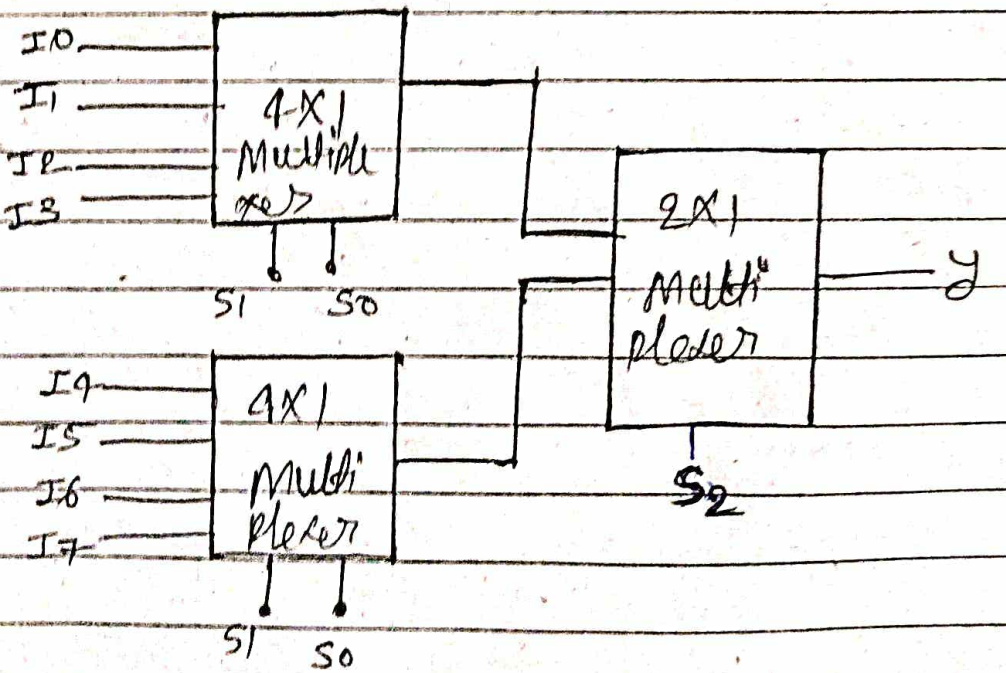
Demultiplexer : It is the combinational circuit that has many outputs and single input, it has select lines, used to distribute the input signal to one of the output lines,



It have select lines which is used to select which output line send to the input.

(Pg5) Design an 8×1 Multiplexer using 4×1 and 2×1 multiplexer.

Inputs			output
S2	S1	S0	Y
0	0	0	I0
0	0	1	I1
0	1	0	I2
0	1	1	I3
1	0	0	I4
1	0	1	I5
1	1	0	I6
1	1	1	I7



Truth table

S1	S0	X0	y1	y2	y3
0	0	1	0	0	0
0	1	0	1.	0	0
1	0	0	0	1	0
1	1	0	0	0	1

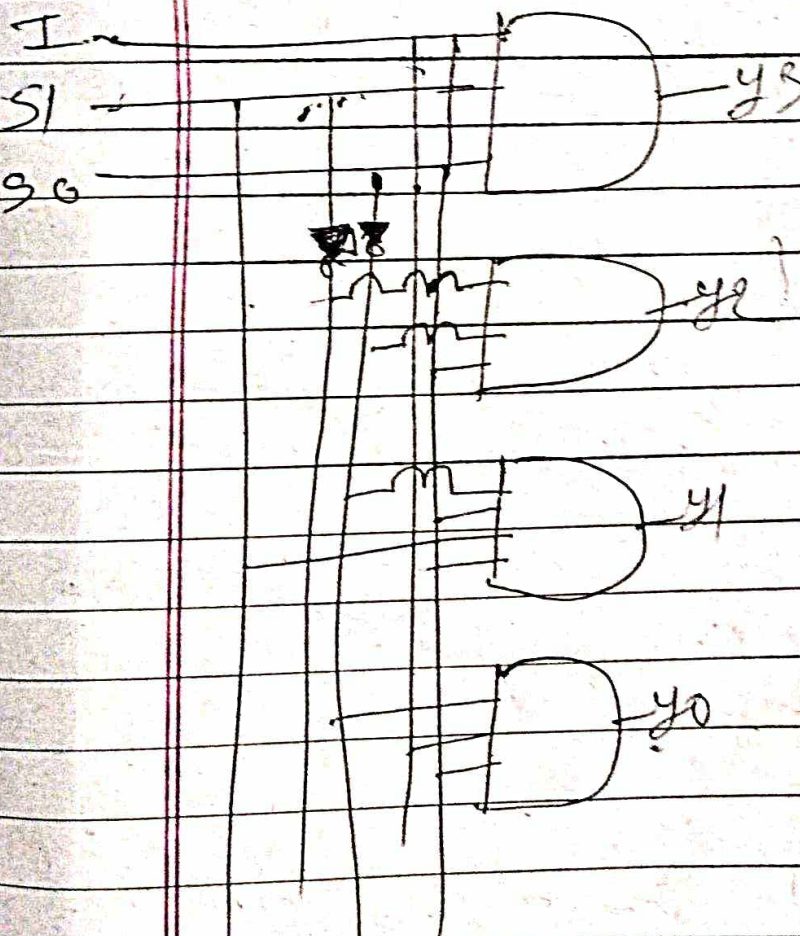
$y_0 = S_1' S_0' I$

$y_1 = S_1' S_0 I$

$y_2 = S_1 S_0' I$

$y_3 = S_1 S_0 I$

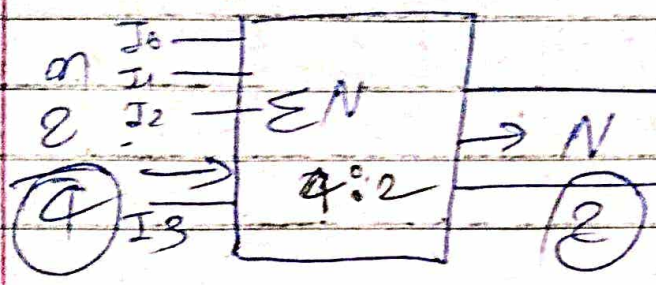
CKt - Diagram



Use of Demultiplexer: Data distribution, Serial to parallel conversion, Data alignment.

Condition of encoder is only one input is active at a time.

(*) Encoder/Decoder :- Encoder : Encoder convert a set of input signals into a single output signal, typically in the form of a binary code.



4:2
8:3
16:4

Application of Encoder
(i) Data compression.
(ii) Analog to digital signal.

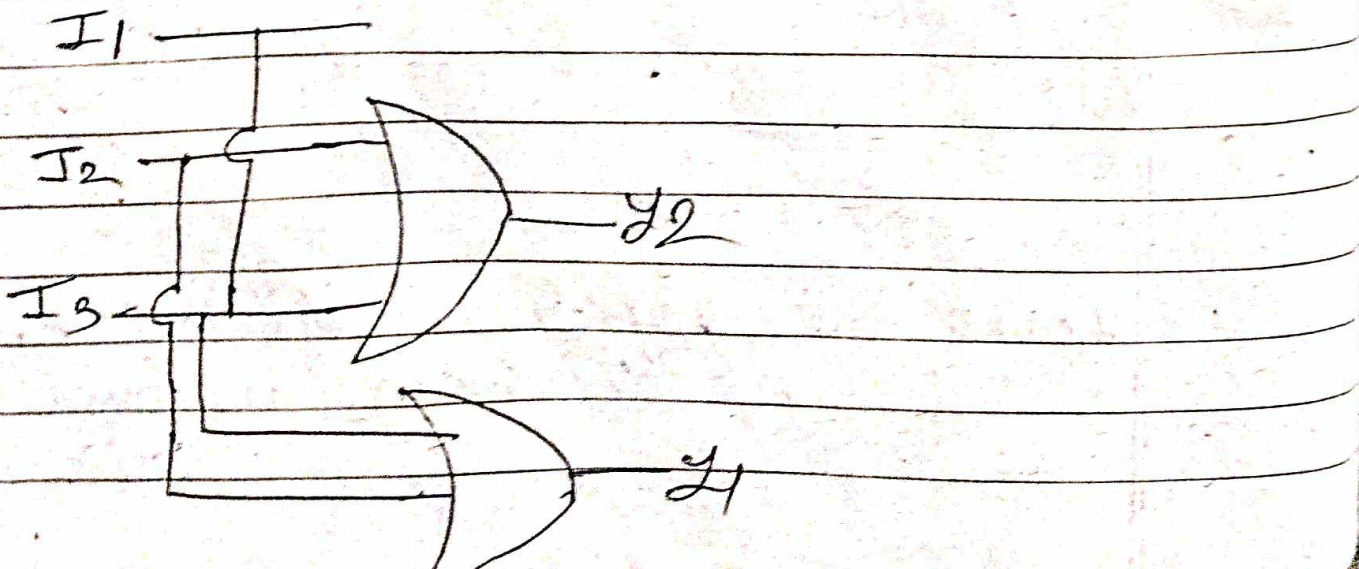
Input				output	
I_0	I_1	I_2	I_3	y_1	y_2
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Time at each of input state

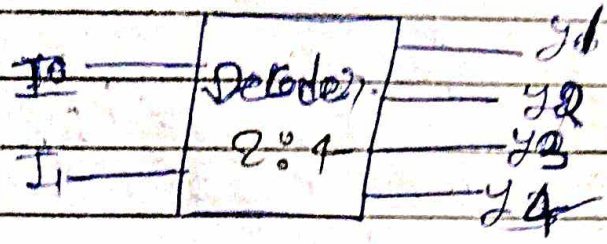
$y_1 = I_2 + I_3$

$y_2 = I_1 + I_3$

Circuit diagram



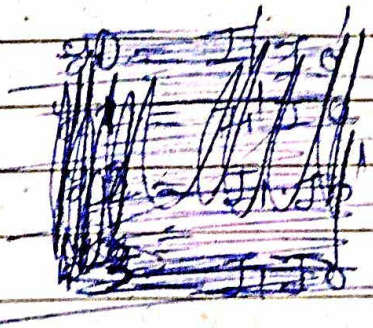
Decoder: Decoders perform the opposite operation, converting ~~a~~ binary code into a set of output signals.



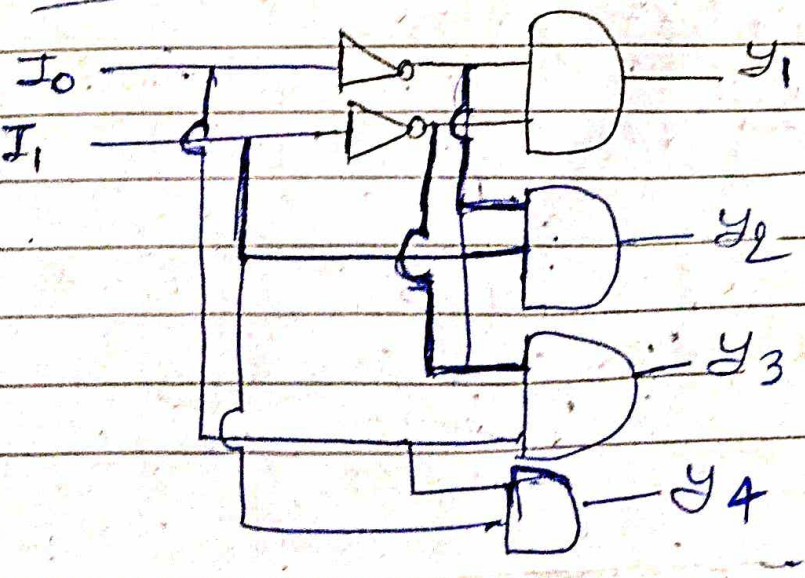
2:4 ✓
3:8 ✓ final
4:16

I/P T.T. O/P

I_0	I_1	y_1	y_2	y_3	y_4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



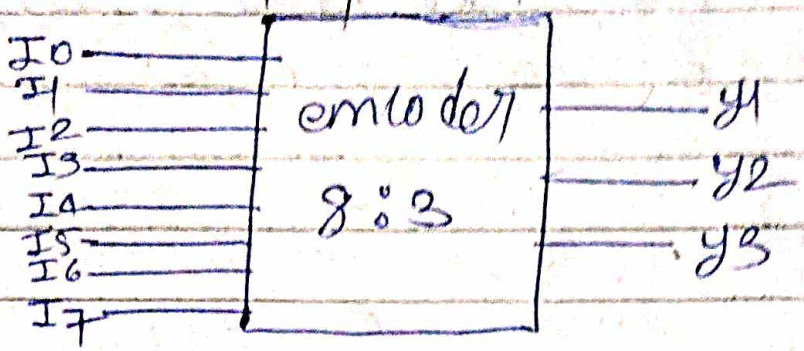
$y_1 = I_0' I_1'$
$y_2 = I_0' I_1$
$y_3 = I_0 \cdot I_1'$
$y_4 = I_0 I_1$



Q in final

8:3 in encoder

T.T

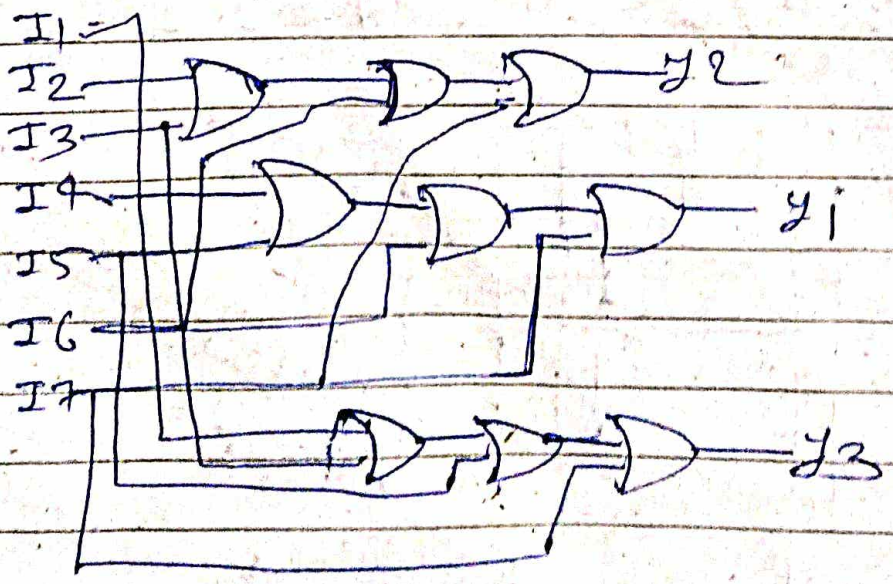


I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7		y_1	y_2	y_3
1	0	0	0	0	0	0	0		0	0	0
0	1	0	0	0	0	0	0		0	0	1
0	0	1	0	0	0	0	0		0	1	0
0	0	0	1	0	0	0	0		0	1	1
0	0	0	0	1	0	0	0		1	0	0
0	0	0	0	0	1	0	0		1	0	1
0	0	0	0	0	0	1	0		1	1	0
0	0	0	0	0	0	0	1		1	1	1

$y_1 = I_4 + I_5 + I_6 + I_7$

$y_2 = I_2 + I_3 + I_6 + I_7$

$y_3 = I_1 + I_3 + I_5 + I_7$



Parity checker

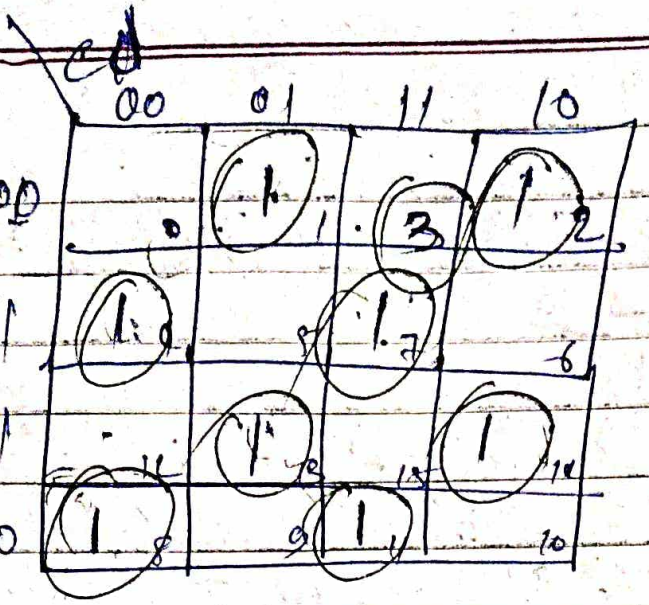
parity bit: A parity bit or parity check is a bit added to a string of binary data to insure that no of ones (1s) in the data must be even or odd. It is used for error detection.

It is of two types.

(i) even parity (ii) odd parity

Even parity checker.

	A	B	C	pd	P⊕C
0	0	0	0	0	0 (correct)
1	0	0	0	1	1 (error)
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0



g1				g2				g3			
a	b	c	p	a	b	c	p	a	b	c	p
0	0	0	1	0	0	1	0	0	1	0	0
$a'b'c'd$				$a'b'cd'$				$a'bc'd'$			

g4				g5				g6			
a	b	c	p	a	b	c	p	a	b	c	p
0	1	1	1	1	1	0	1	1	1	1	0
$a'bcd$				$abc'd$				$abcd'$			

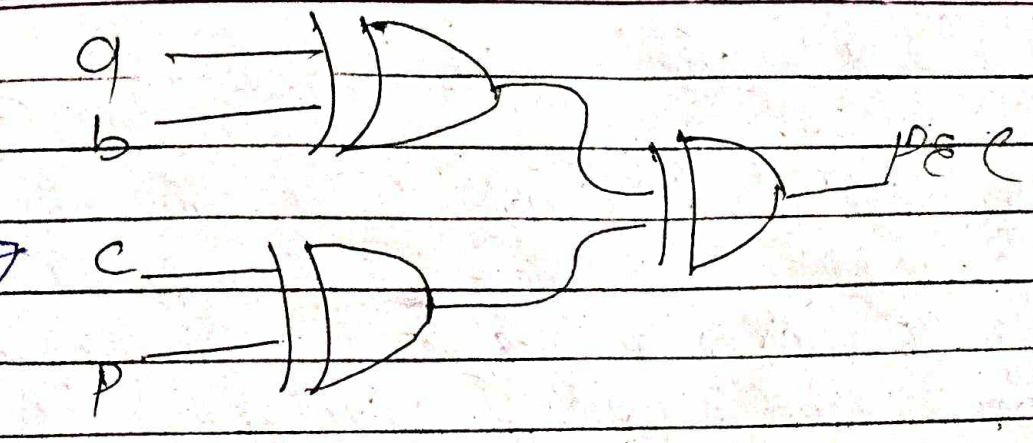
g7				g8			
a	b	c	p	a	b	c	p
1	0	0	0	1	0	1	1
$ab'c'd'$				$ab'cd$			

answer = $(a \oplus b' \oplus c \oplus d)$
p&c

$y = a'b'c'p + a'b'cp' + a'bc'p' + a'bcp + abc'p + abcp' + ab'c'p' + ab'cp$ $[p=d]$

$y = a'b'[c'p + cp'] + a'b[c'p' + cp] + ab[c'p + cp'] + ab'[c'p' + cp]$

Ckt diagram.



$(a \oplus b) \oplus (c \oplus p)$

$y = a'b'[c \oplus p] + a'b[c \odot p] + ab[c \oplus p] + ab'[c \odot p]$

$y = a'b'[c \oplus p] + ab[c \oplus p] + a'b[c \odot p] + ab'[c \odot p]$

$y = (c \oplus p)[a'b' + ab] + [c \odot p](a'b + ab')$

$y = (c \oplus p)[a \odot b] + (c \odot p)(a \oplus b)$

$y = (c \oplus p)(a \oplus b)' + (c \oplus p)'(a \oplus b)$

($c \oplus p$: a, $a \oplus b$: b)

$a'b + ab' \Rightarrow a \oplus b$ $\quad [p = d]$

$y = (c \oplus p) \oplus (a \oplus b)$

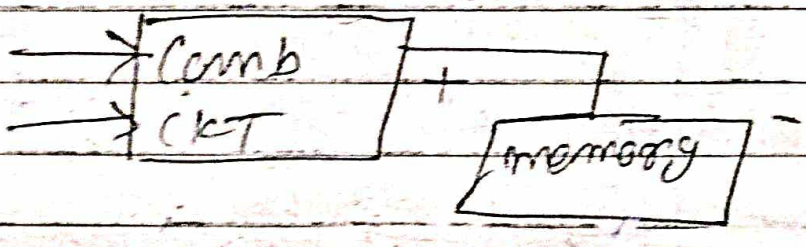
$y = (a \oplus b) \oplus (c \oplus d)$

$y = (a \oplus b) \oplus (c \oplus d)$

Date / /
Page No.

Sequential circuit.

* Sequential circuit : Combinational circuit does not use any memory hence previous state of input circuit does not have any effect on circuit but sequential circuit has a memory so this type of circuit uses previous input and previous output.



* Explain the concept of memory in sequential circuits.

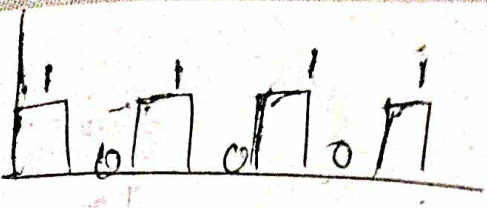
→ The concept of memory in sequential circuits is realized through flip-flops which are basic storage elements. These memory elements can store binary information (0s & 1s) can be used to hold the current state of the circuit.

(Ans) flip flop :- It is a binary storage device, It can store binary bit either 0 or 1

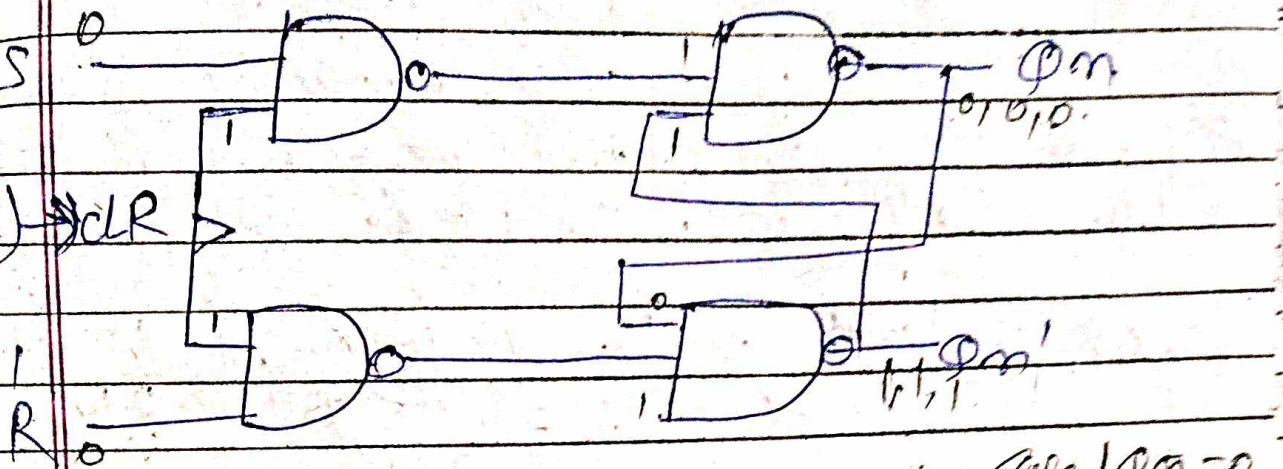
(Ans) Types of flip flop

1. ↳ SR ff / Set Reset ff
2. ↳ JK ff / mom
3. ↳ D ff / Direct ff
4. ↳ T ff / Toggle

(1) SR flip flop :- The SR flip-flop has two inputs S (Set) and R (reset), and two outputs, Qn and Qn' when S input is high, the output Qn is set to high, when R input is high, the output Qn is reset to low.



86

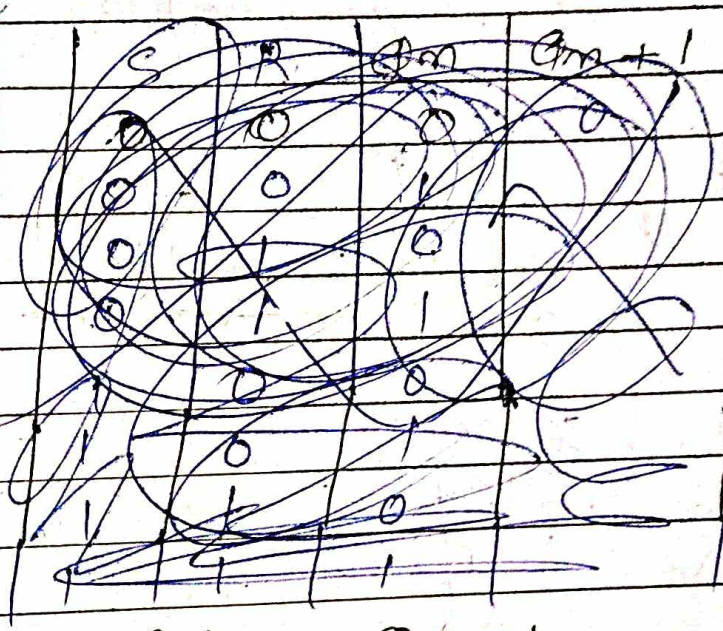


Case 1 $Q_n = 0$

Case 2 $Q_n = 1$

$Q_n = 0 \rightarrow Q_n' = 1$

$Q_n = 1 \rightarrow Q_n' = 0$

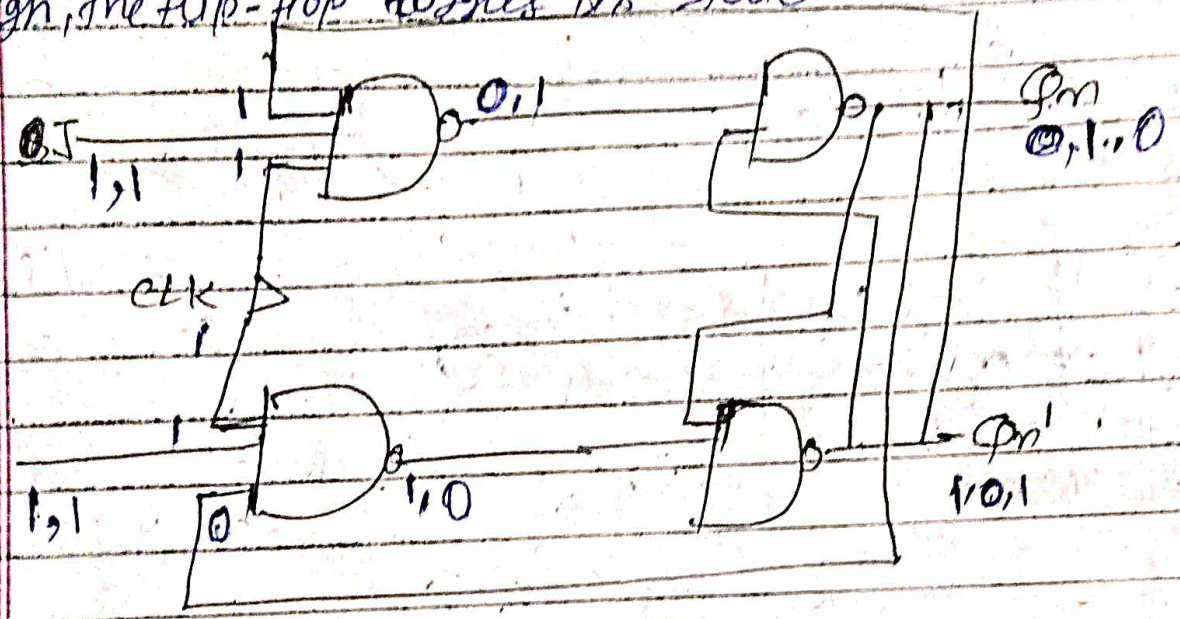


Case 2 $Q_n = 1$

S	R	Q_n	Q_{n+1}	
0	0	0	0	memory state
0	0	1	1	
0	1	0	0	reset state
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X (invalid)	invalid
1	1	1	X	

The JK flip-flop is similar to the SR flip-flop, but it has an additional

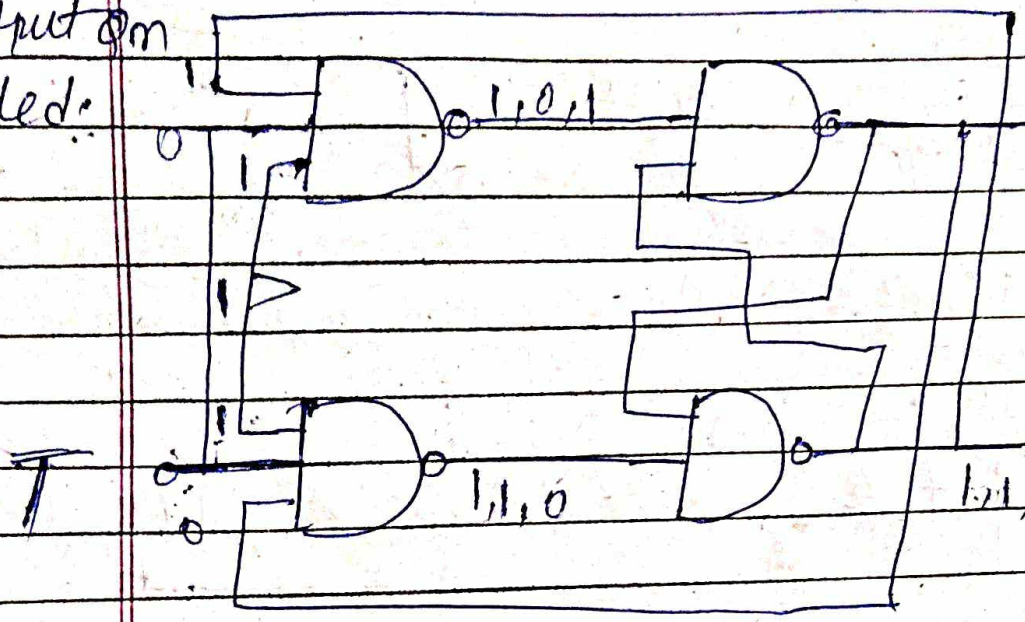
(X) JK flip flop ∴ J input, when the J and K inputs both are high, the flip-flop toggles its state.



J	K	Qn	Qn+1	
0	0	0	0	Memory State
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

Date / /
Page No.

(*) T flip flop : T flip-flop is a simplified version of the JK flip-flop. It has a single input T and two outputs Q_n & Q_n' when the clock input is high the output Q_n is toggled.



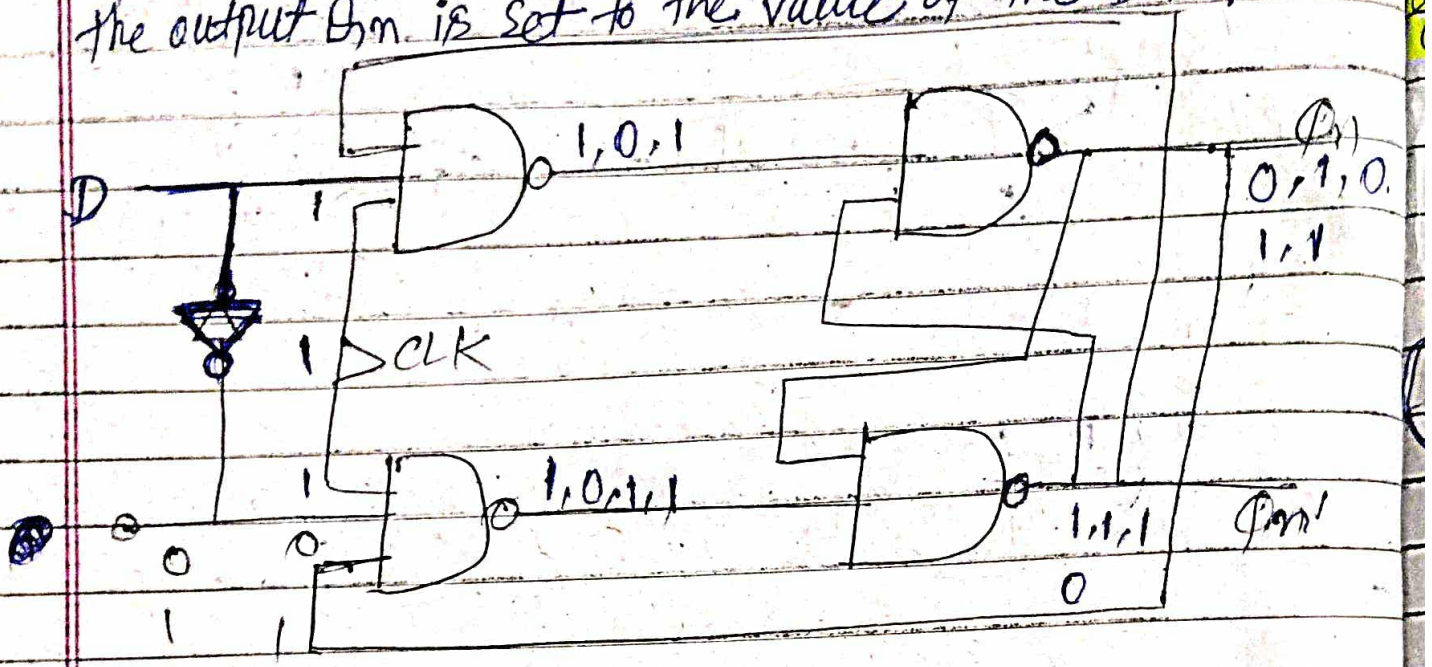
T	Q_n	Q_{n+1}
0	memory state	Q_n
1	Toggle state	Q_n'

→

T	Q_n	Q_{n+1}	
0	0	0	memory state
0	1	1	
1	0	1	toggled
1	1	0	

(*) Race around Condition : Race around condition in flip flops when clock plus is applied, output changes when the input changes. But in JK flip flop when J and K are one and clock is high for a long period of time then the output Q will toggle as soon as the clock is high. This problem is called race around condition

(*) D flip flop: The D flip-flop has a single input D and two outputs Qn and Qn' when the clock input is high the output Qn is set to the value of the D input.



D	Qn
0	Reset
1	set

→

D	Qn	Qn+1	
0	0	0	Reset state
0	1	0	
1	0	1	set state
1	1	1	

* Race around condition: For J-K flip-flop, if J=1, K=1 and if clock pulse =1 for a long period of time, the output Q will toggle as long as the clock pulse is remain high which makes the output unstable or uncertain. This is called race around condition.

Master slave flip flop rectifies race around condition

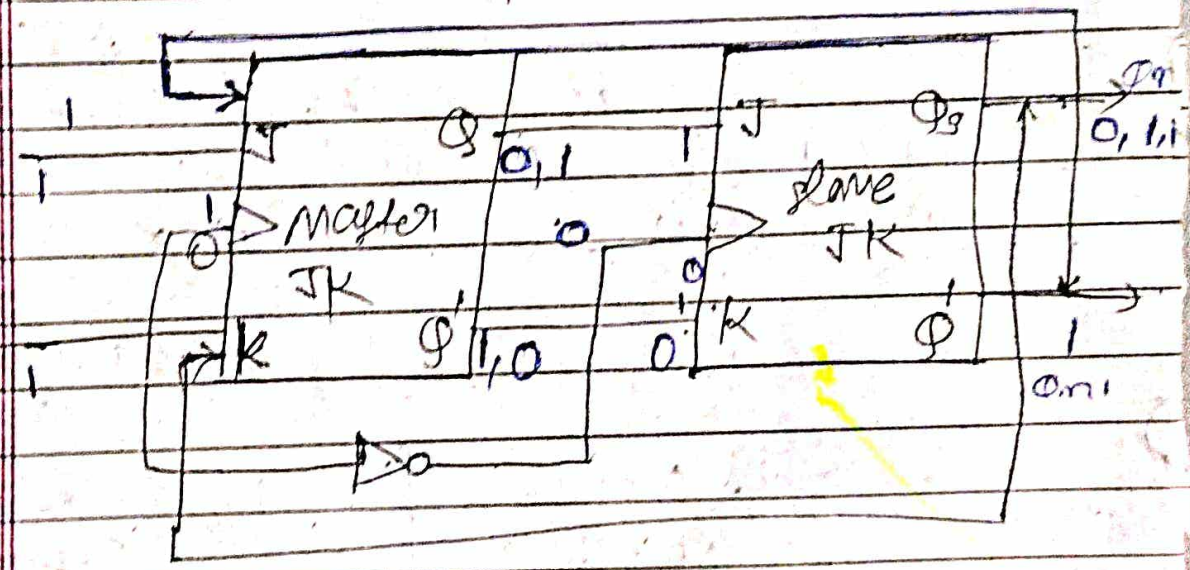
How to remove race around condition.

ans:

(i) if we reduce the clock high time then racing can be avoided

(ii) Master slave flip flop.

Master slave flip flop.



A master Slave flip-flop rectifies the race around conditions by using two separate flip-flops. The first flip-flop called the master. The second flip-flop called the slave.

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

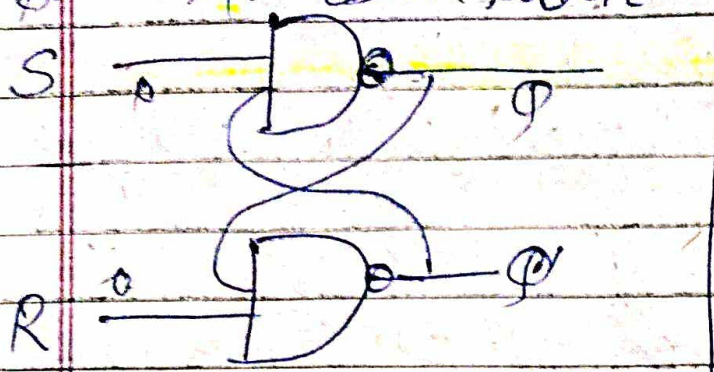
So race around condition is rectified.

COU

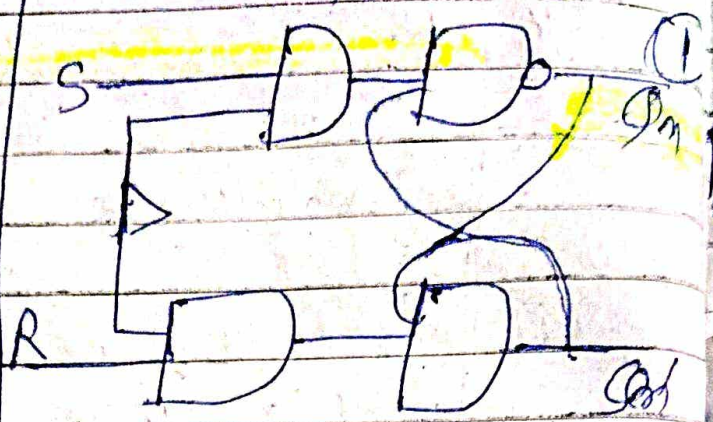
Date / /
Page No.

DLff

SR latch



SR ff



SR ff

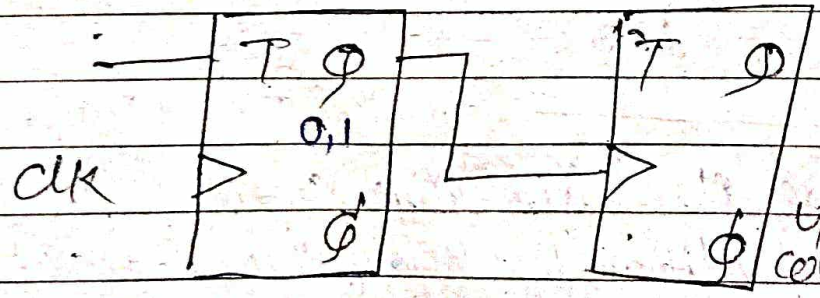
* What are counters?

or ⇒ It is a circuit which is used to count the pulses, it is a group of flip-flop with a clock signal apply, it is of two types.

(i) Asynchronous counters.

(ii) synchronous counters. ←

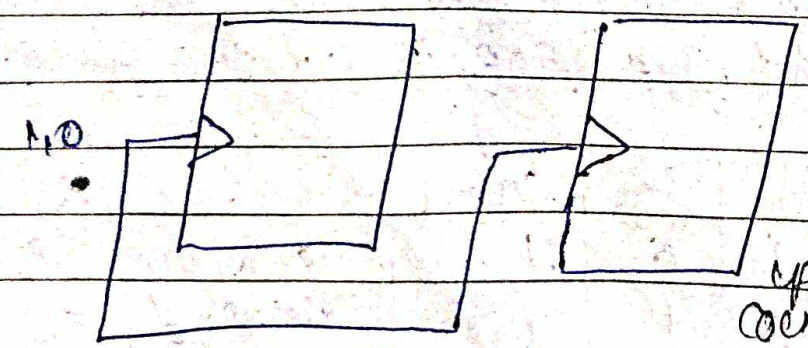
2-bit counter using T ff



Asynchronous counter

up counter

down counter



Synchronous counter

up counter

down counter

Date / /
Page No.

counter → Asynchronous : → up counter [redacted]
→ down counter ([redacted] 0)

counter → [redacted] synchronous → up counter
→ down counter

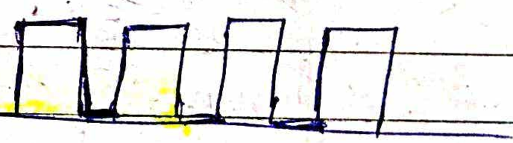
Up counte: 3, 2, 1 ↑ starting from here

down counter: 3 → starting from here, 2, 1 ↓ down

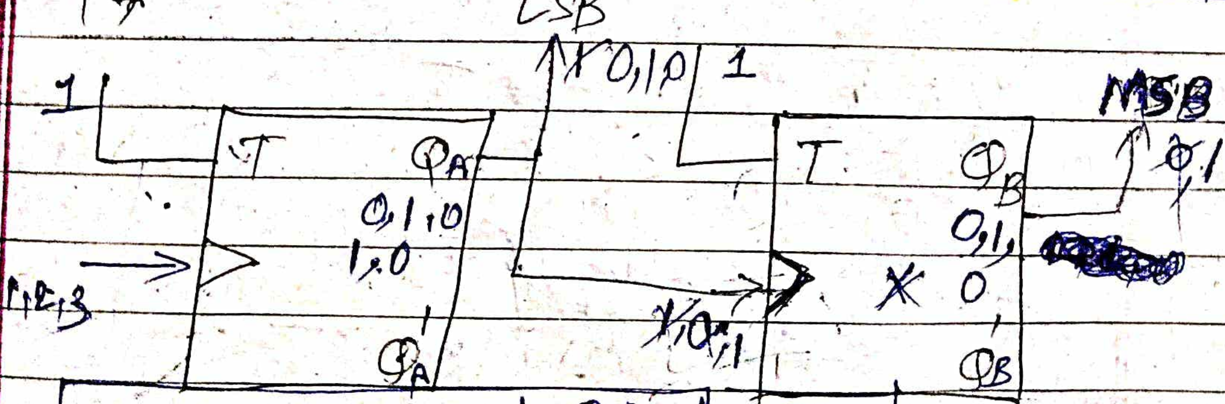
→ Asynchronous 2 bit up counter using Tff

ff = 2

Type = T, J.K



T →

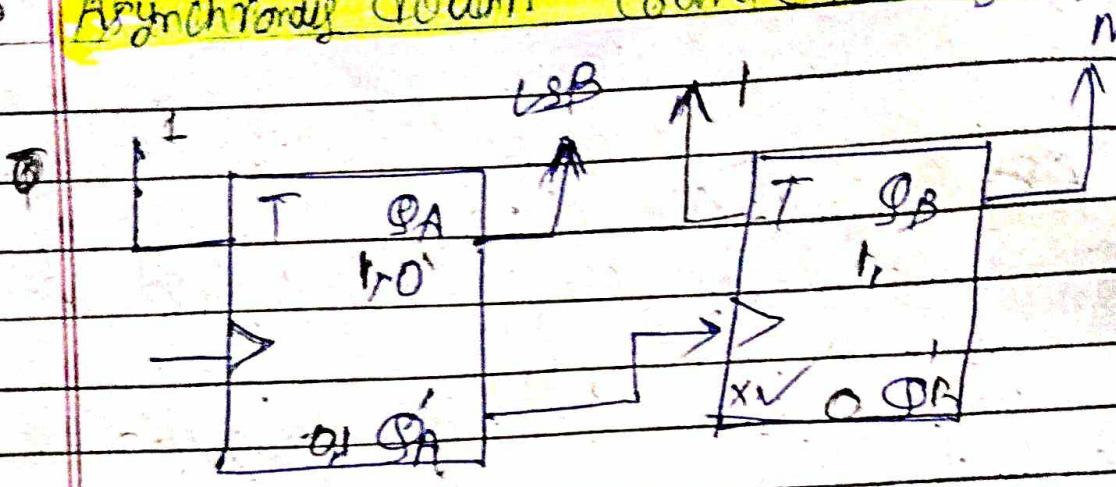


clock	QB	QA
0	0	0
1	0	1
2	1	0
3	1	1

93 27

Date / /
Page No.

* T → Asynchronous down Counter Using T ff.



Binary

Clock	QB	QA
3	1	1
2	1	0
1	0	1
0	0	0
3	1	1

2-bit

* 2-bit Synchronous up Counter Using T ff

(1) Type = 2 ff = T

(2) Excitation table

Present Q	Next Q+1	T
0	0	0
0	1	1
1	0	1
1	1	0

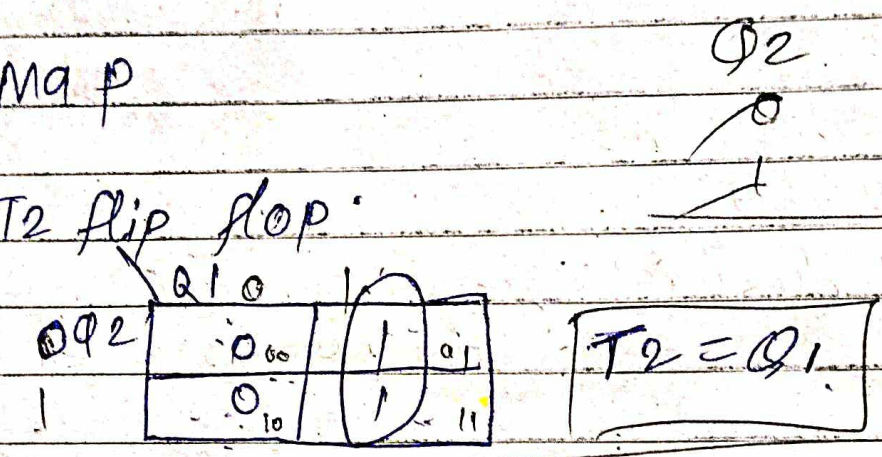
(3) State table with excitation table

Date / /
Page No.

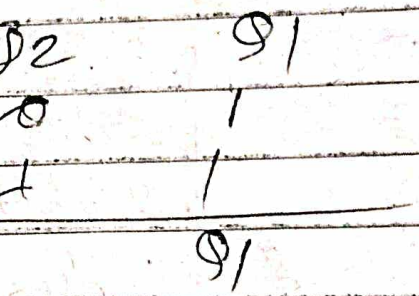
present state		next state		flip flop	
Q_2	Q_1	Q_2^+	Q_1^+	T_2	T_1
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

K Map

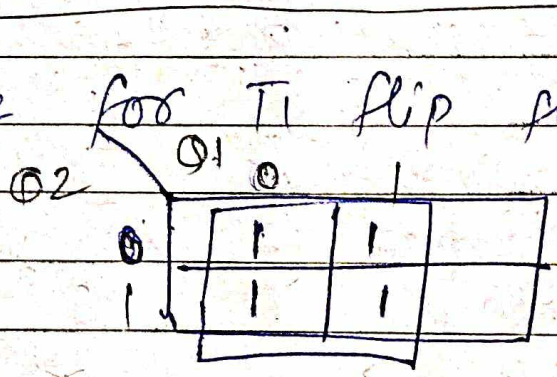
for T_2 flip flop:



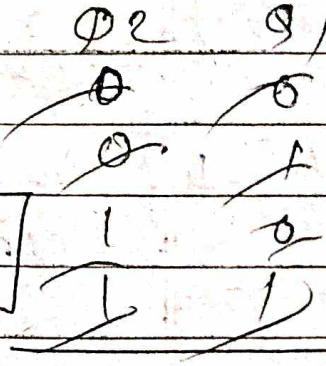
$T_2 = Q_1$



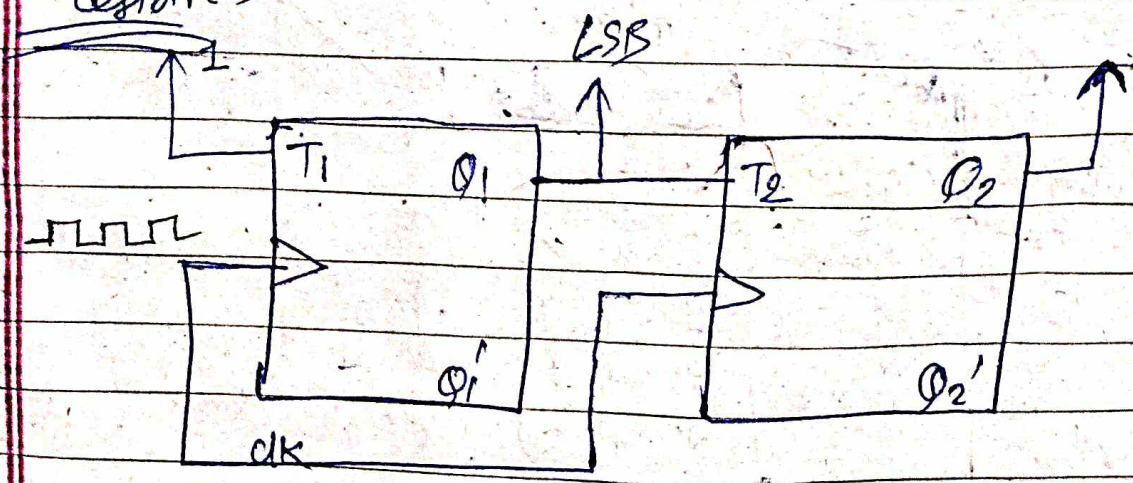
for T_1 flip flop



$T_1 = 1$



design →

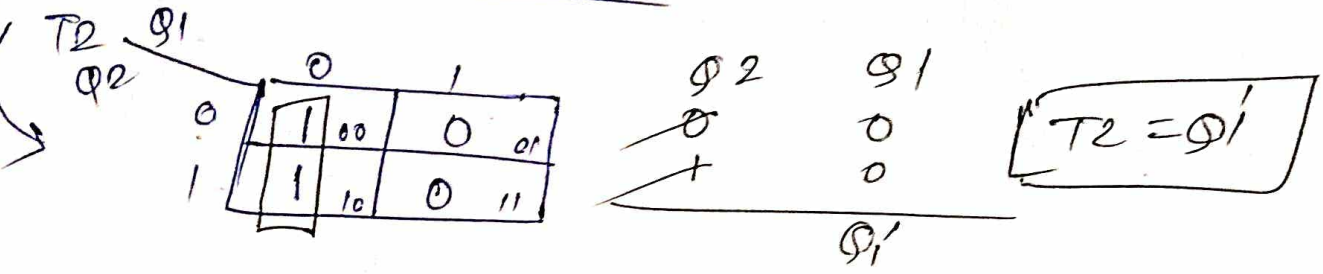


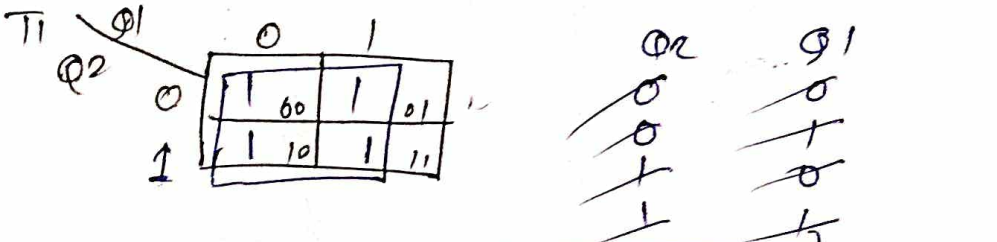
State table → sym → down counter

Previous state		Next state		flip flop	
Q2	Q1	Q2+	Q1+	T2	T1
1	1	1	0	0	1
1	0	0	1	1	1
0	1	0	0	0	1
0	0	1	1	1	1

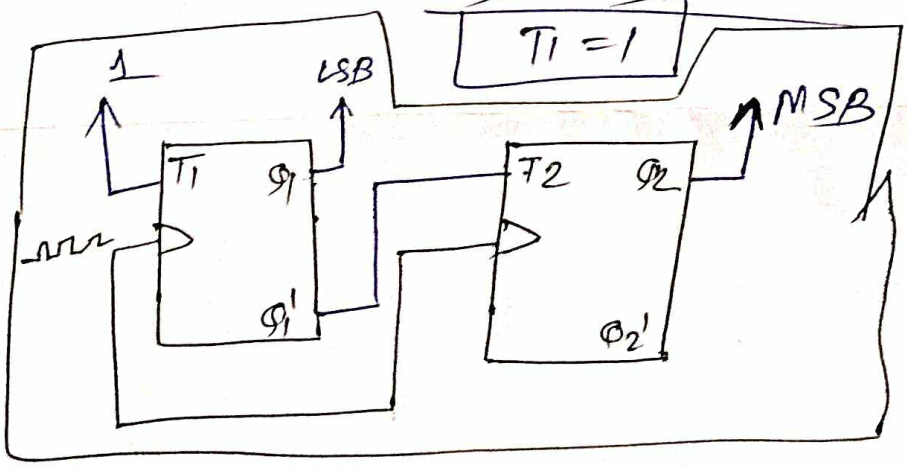
Step = 3

K map





Step 4



Excitation Table

Previous Qn	Next Qn+1	T
1	1	0
1	0	1
0	1	1
0	0	0

Step = 2

Step-1 = Type = T

ff → 02

2-bit

Synchronous down counter

MST

2- bit synchronous up counter using JK ff

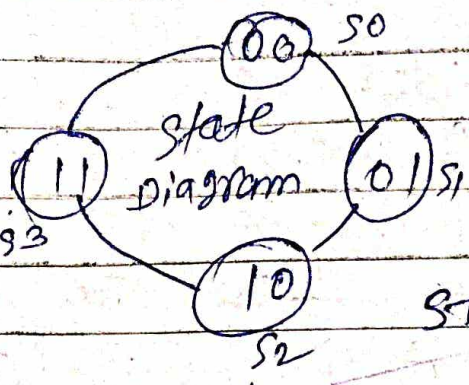
Step-1: No of flip flops = 2

Type of flip flop = JK

Step:2 → Excitation table.

Present state	Next state	ff	
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Step 3: State table:



$n \quad 2$
$2^n - 1 = 2^2 - 1 = 3$

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

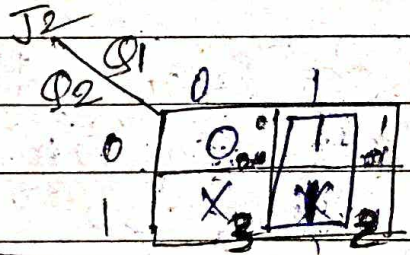
$00 \rightarrow 01 \rightarrow 10 \rightarrow 11$

$n \quad 2$
No of state $= 2^n = 2^2 = 4$

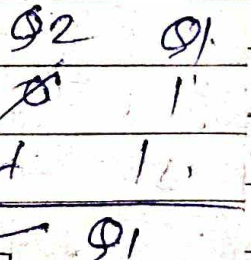
state table

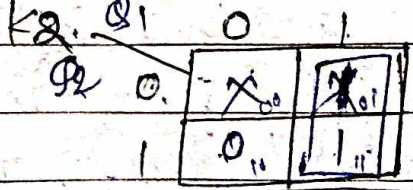
	Previous States		Next States		flip flop		flip flop	
	Q_2	Q_1	Q_2^+	Q_1^+	J_2	K_2	J_1	K_1
0	0	0	0	1	0	X	1	X
1	0	1	1	0	1	X	X	1
2	1	0	1	1	X	0	1	X
3	1	1	0	0	X	1	X	1

Step 4: K-map

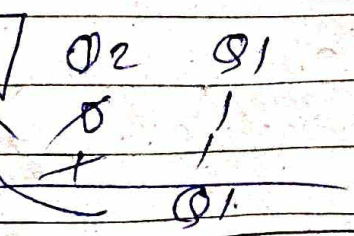


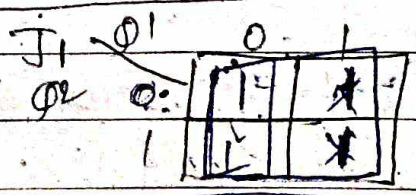
$J_2 = Q_1$



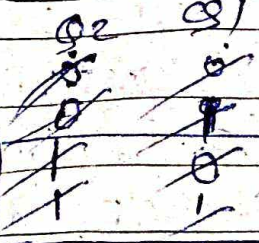


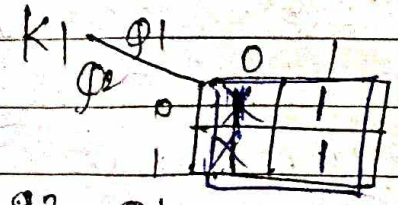
$K_2 = Q_1$



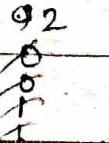


$J_1 = 1$





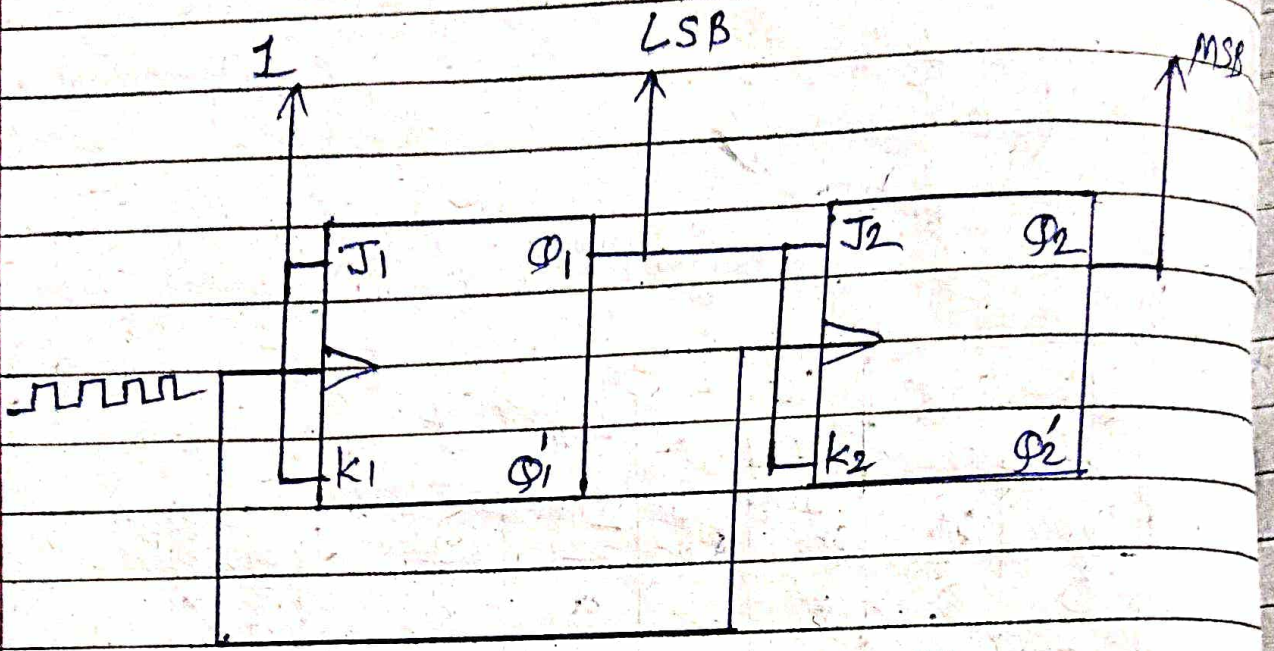
$K_1 = 1$



$K_1 = 1$

Step :5 Circuit diagram

$J_2 = Q_1$, $K_2 = Q_1$ $J_1 = 1$ $K_1 = 1$



(Q9) Describe synchronous and asynchronous counters.

Synchronous counter : A synchronous counter is also known as a parallel counter, In the synchronous counter there are continuous clock input signals with flip-flops used to produce the output, it is faster than Asynchronous.

Asynchronous counter : Asynchronous counter also known as the serial counter. In Asynchronous counters, different clock signals are used to produce the output. It is slower than synchronous.

Date / /
Page No.

(Pg) Compare level and edge triggering of clock.

	Edge triggering	level triggering
(1)	It allows a circuit to become active at the positive edge or the negative edge of the clock signal.	It allows a circuit to becomes active when the clock pulse is on a particular level.
(2)	Example: flip-flops Counters	Example: Latches Memory cells
(9)	Application: Digital clocks	Application: Digital to analog converters.
(10)	It provides precise timing to control synchronous operations	It doesnot provide precise timing to control and typically used for asynchronous events

(Pg) Difference b/w flipflop and latch

	flipflop	latch
(i)	flipflop is edge-triggered	latch is level-triggered
(ii)	flipflops are synchronous.	latches are asynchronous.
(8)	It is used in Registers, Counters, memory	It is used in asynchronous sequential logic.
(9)	It has a clock signal.	It donot have a clock signal.

Date / /
Page No.

(pyq) Difference between Combinational and Sequential circuits.

Combinational circuit	Sequential circuit
(1) It's output depends on present inputs only	It's output depends on inputs and previous outputs.
(2) It does not have memory	It have memory.
(3) It is not time dependent	It is time dependent
(4) It is use to perform Arithmetic and Boolean operation.	It is use to Data Storage, Sequencing events.
(5) Examples: logic gates Encoder, decoder, adders Multiplexers, Demultiplexer	Examples: Flip flops, Counters, Registers latches, memory chips.
(6) It is easy to design.	It is harder to design than sequential circuits.

(pyq) What is shift register, Describe shift register, Explain PISO and SIPO shift register

ans→ A shift register is a device, it is a group of flip-flops connected in series used to store multiple bits of data.

Type of shift Registers

(1) Serial in Serial out SR (SISO)
(2) Serial in parallel out SR (SIPO)
(3) parallel in Serial out SR (PISO)
(4) parallel in parallel out SR (PIPO)

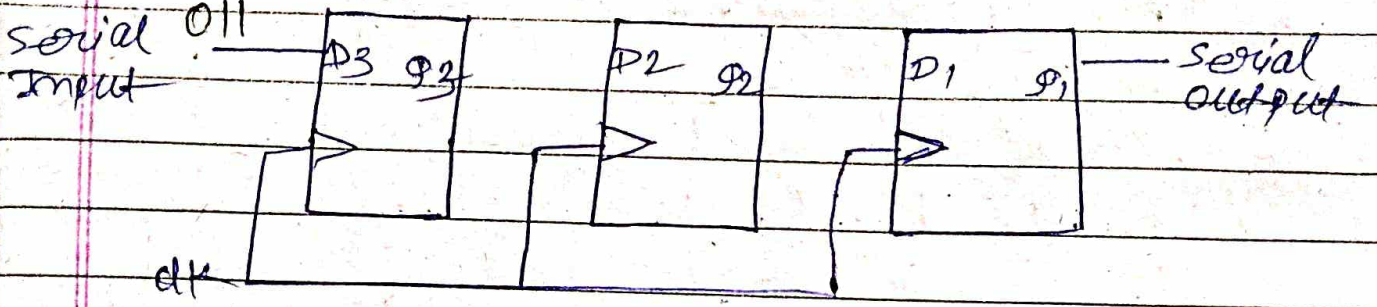
100

(Q10) Explain the working of 3 bit shift register

ans⇒ If the register is capable of shifting data bits either towards right hand side or towards left hand side is known as shift register. It is a group of flip-flop connected in series.

SISO shift register

Serial in - serial out



clk	input	Q3	Q2	Q1
0	—	0	0	0
1	(LSB) 1	1	0	0
2	1	1	1	0
3	(MSB) 0	0	1	1 (LSB)
4	—	—	0	1
5	—	—	—	0 (MSB)

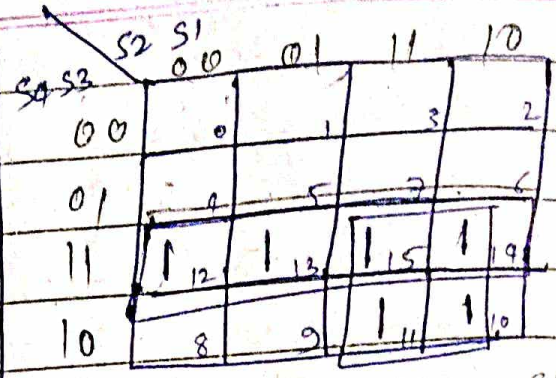
working ÷ We are sending '011' binary info assume initial status of the flip flop from left most to right most is Q3 Q2 Q1 = 000

The initial status of the D ff in the absence of clock signal is Q3 Q2 Q1 = 000 the serial output is coming from Q1. So the LSB (1) is received at 3 positive edge of clock. and the MSB (0) is received at 5 positive edge of clock. Therefore 3 bit SISO shift register requires five clock pulses in order to produce the valid output.

(Pts) 101

Draw the circuit of BCD adder and explain it.

	S_4	S_3	S_2	S_1	y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1



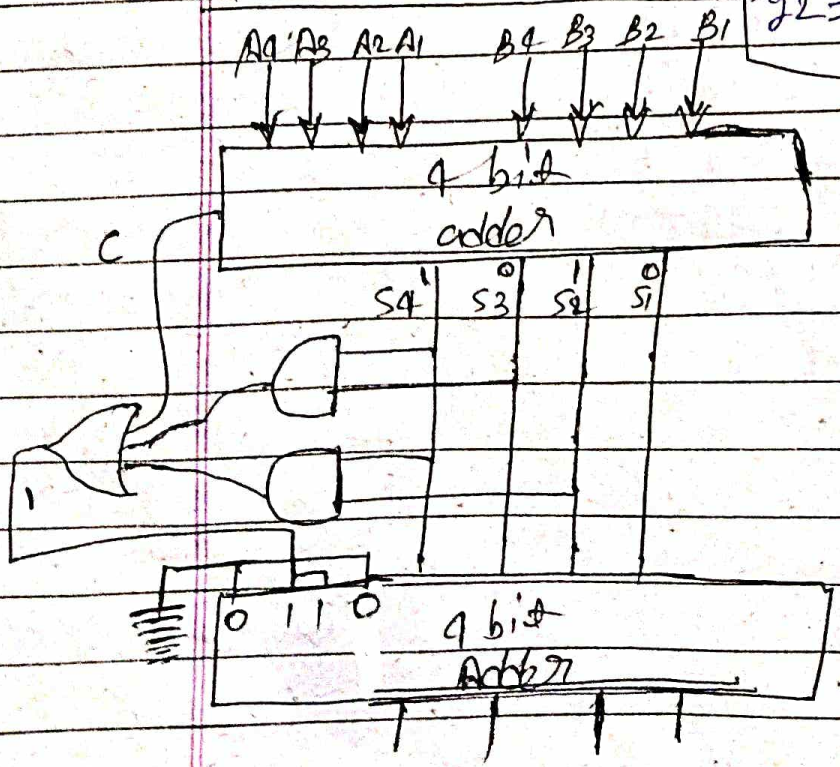
$g_i = S_4 S_3 + S_4 S_2 S_1$

$g_1 = S_4 S_3 + S_4 S_2$

$g_2 = S_4 S_2$

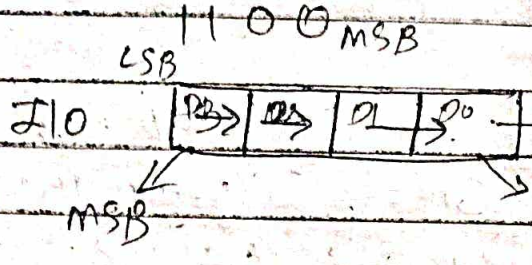
$y = S_3 S_4 + S_2 S_4$

$y = S_3 S_4 + S_4 S_2$

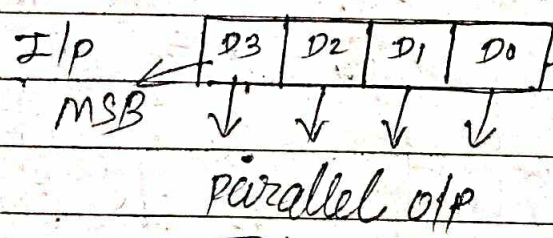


Date / /
Page No.

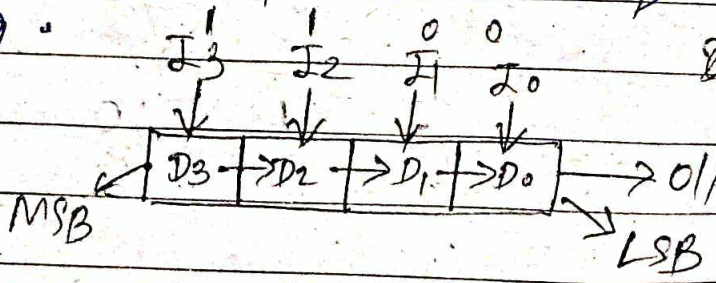
SISO° shift Register: This shift register allows serial input and produces a serial output is known as a serial in Serial-out shift register.



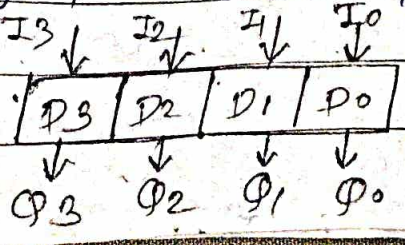
(2) SIPO: This shift register allows serial input and produces a parallel output is known as the Serial-in parallel out shift register.



(3) PISO shift register: This shift register allows parallel input and produces a serial output is known as a parallel-in serial out shift register.



(4) PIPO shift register: This shift register allows parallel input and also produces a parallel output is known as parallel-in parallel out shift register.



Explain the working of gray code.

a 4 bit binary to gray converter.

(Q18) Design and implement ~~4 bit counter~~ using ~~K-map~~. its importance and its ~~using~~

Truth

	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

to 15

for

0, 11, 14, 13, 19, 15)

16, 7, 8, 9, 10, 11)

9, 5, 10, 11, 12, 13)

5, 6, 9, 10, 13, 14)

& G3

11 10

3	2
7	6
15	14
11	10

B_3 B_2 B_1 B_0

$G_3 = B_3$

B_3

table

$2^{4} = 16$

Range = 0 to 15

K Map for

$G3 = \Sigma m(8, 9, 10, 11, 12, 13, 14, 15)$

$G2 = \Sigma m(4, 5, 6, 7, 8, 9, 10, 11)$

$G1 = \Sigma m(2, 3, 4, 5, 10, 11, 12, 13)$

$G0 = \Sigma m(1, 4, 5, 6, 9, 10, 13, 14)$

K Map for G3

B3 B2 \ B1 B0	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	1 12	1 13	1 15	1 14
10	1 8	1 9	1 11	1 10

G3

B3	B2	B1	B0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	0	0
1	0	0	1
1	0	1	1
1	0	1	0

$G3 = B_3$

Date / /
Page No.

GR

G12

B3B2 \ B1B0	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

B3	B2	B1	B0
0	+	0	0
0	+	0	1
0	+	+	1
0	+	+	0

B3' B2

B3	B2	B1	B0
1	0	0	0
1	0	0	1
1	0	+	1
1	0	+	0

B3 B2'

B3' B2 + B3 B2'

G12 = B3' B2 + B3 B2'

G12 = B3 ⊕ B2

G11 for G13

B3B2 \ B1B0	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

G11 = B2 B1' + B2' B1

G11 = B2 ⊕ B1

G1

B3	B2	B1	B0
0	1	0	0
0	1	0	1
1	1	0	0
1	1	0	1

B2 B1'

G2

B3	B2	B1	B0
0	0	1	1
0	0	1	0
1	0	1	1
1	0	1	0

B2' B1

Date / /
Page No.

G1 0

B3 B2 \ B1 B0	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

g1

B3	B2	B1	B0
0	0	0	1
0	1	0	1
1	1	0	1
1	0	0	1

$B_1' \quad B_0$

g2

B3	B2	B1	B0
0	0	1	0
0	1	1	0
1	1	1	0
1	0	1	0

$B_1 \quad B_0'$

$G_{10} = B_1' B_0 + B_1 B_0'$

$G_{10} = B_1 \oplus B_0$

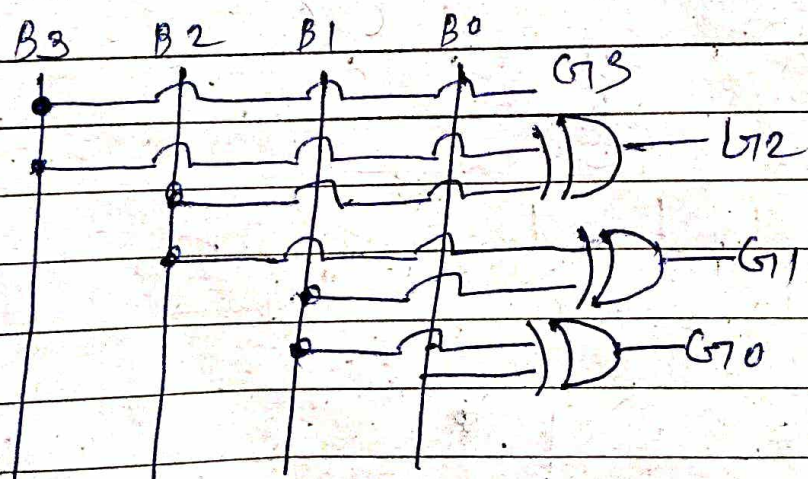
$G_{G3} = B_3$

$G_{12} = B_3 \oplus B_2$

$G_{11} = B_2 \oplus B_1$

$G_{10} = B_1 \oplus B_0$

Circuit diagram



(iv) Its importance and uses.

(i) Error Reduction in A to D Conversion.

(ii) Communication System

(iii) Minimizing Glitching in Digital circuits.

PYQ Design 2-bit synchronous Down counter using JK flip flop.

ans ⇒ Step 1: Type = JK ff = 2

② Excitation table

S1 → No of ff = 2

Type of ff = JK

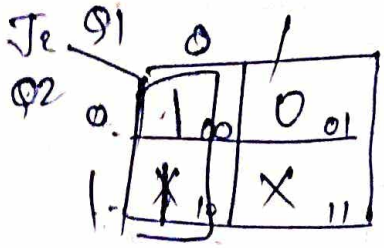
Excitation ff

Previous state		Next state	
Qn	Qn+1	J	K
0	1	X	0
1	0	X	1
0	1	1	X
0	0	0	X

State table

Previous state		Next State		flip flop			
Q2	Q1	Q2+	Q1+	J2	K2	J1	K1
1	1	1	0	X	0	X	1
1	0	0	1	X	1	1	X
0	1	0	0	0	X	X	1
0	0	1	1	1	X	1	X

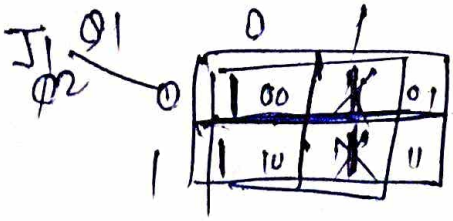
K-Map



Q2	Q1
0	0
1	0

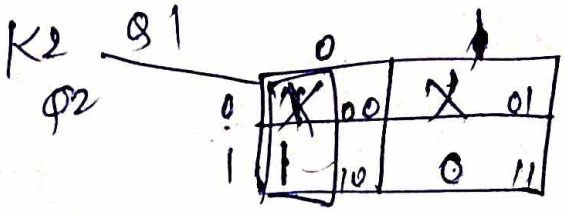
Q1'

$J_2 = Q_1'$



Q2	Q1
0	0
0	1
1	0
1	1

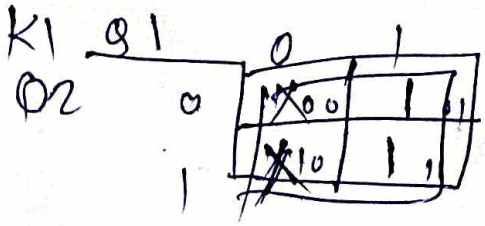
$J_1 = 1$



Q2	Q1
0	0
1	0

Q1'

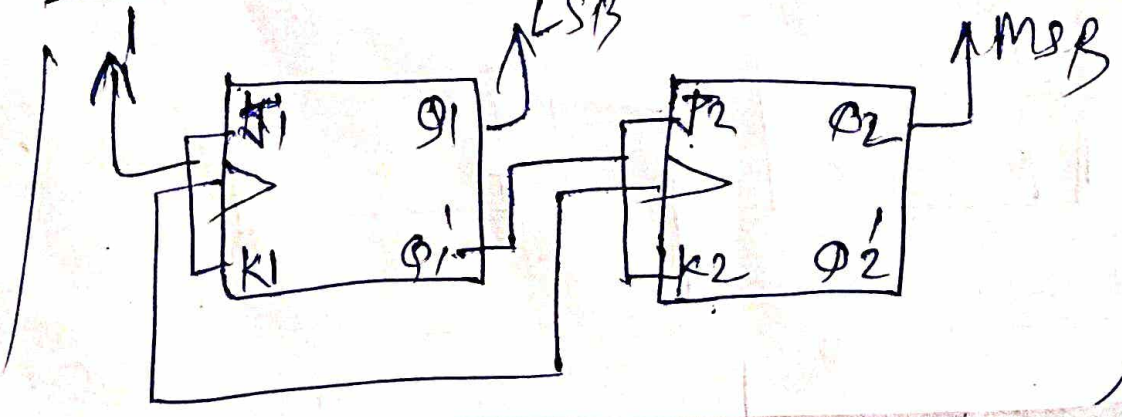
$K_2 = Q_1'$



Q2	Q1
0	0
0	1
1	0
1	1

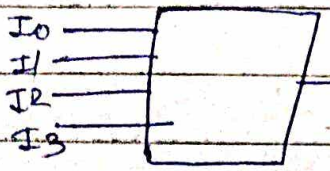
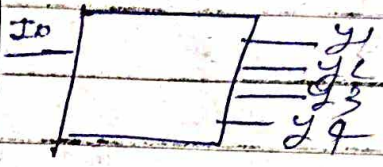
$K_1 = 1$

$J_2 = Q_1'$, $J_1 = 1$, $K_2 = Q_1'$, $K_1 = 1$



Date / /
Page No.

Q8) Compare encoder decoder

encoder	decoder
(i) It convert a set of input into a single output signal.	It convert a single input into a set of output signal.
(ii) 	
(iii) application: memory addressing, data selection	application: Synchronization delay.
(iv) Multiple input & single output	Multiple output & single input.

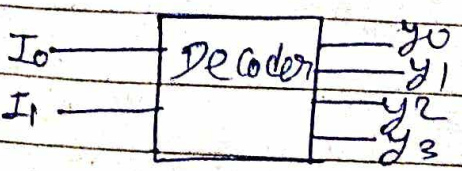
Q9) Compare decoder and demultiplexer with suitable block diagram.

ans) decoder

(i) Multiple inputs. ~~and single output.~~

(2) It Converts coded input into a set of outputs.

(3) App: memory system

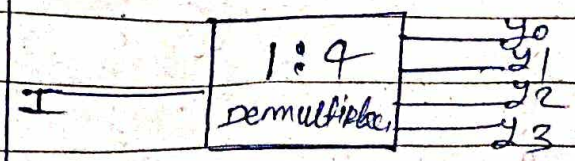
(4) 

demultiplexer

single input.

It distributes a single input to multiple outputs.

App: Communication system. data routing.



Date / /
Page No.

(Q.10) Add following decimal perform subtraction using 2's complement $(101001)_2 - (001100)_2$

Ans) Step: 1 → find 2's complement of $(101001)_2$

$$\begin{array}{rr} & 1\ 0\ 1\ 0\ 0\ 1 \\ 1's\ c \rightarrow & 0\ 1\ 0\ 1\ 1\ 0 \\ 2's\ c \rightarrow & \qquad +1 \\ \hline & 0\ 1\ 0\ 1\ 1\ 1 \end{array}$$

Step: 2 → Add $(010111)_2$ to $(001100)_2$

$$\begin{array}{rr} & 0\ 0\ 1\ 1\ 0\ 0 \\ & 0\ 1\ 0\ 1\ 1\ 1 \\ \hline \text{result} \rightarrow & 1\ 0\ 0\ 0\ 1\ 1 \end{array}$$

Step: 3 if there will carry then omit the carry and that will be your answer. but in case of No carry take again 2's complement of result and put (-) symbol after taking 2's complement that will be your ans.

$$\begin{array}{rr} \text{result} \rightarrow & 1\ 0\ 0\ 0\ 1\ 1 \\ 1's \rightarrow & 0\ 1\ 1\ 1\ 0\ 0 \\ 2's\ c \rightarrow & \qquad +1 \\ \hline & 0\ 1\ 1\ 1\ 0\ 1 \end{array}$$

$$-011101$$

Date 1. 1
Page No.

(pg9) add the following decimal numbers using BCD addition and verify your answer. 599 and 984

Note:- ① Sum ≤ 9, final carry = 0 answer is ✓

② Sum ≤ 9, final carry = 1 answer is ✗

then add 6 (0110)

③ Sum > 9 final carry = 0. ans is ✗

add (0110).

```
  9 8 4
+ 5 9 9
-------
 1 5 8 3
```

	8	4	2	1
4 →	0	1	0	0
8 →	1	0	0	0
9 →	1	0	0	1
5 →	0	1	0	1

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

>9 <9 >9 → add 6 (0110)

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

carry: 1 | 5 | 8 | 3

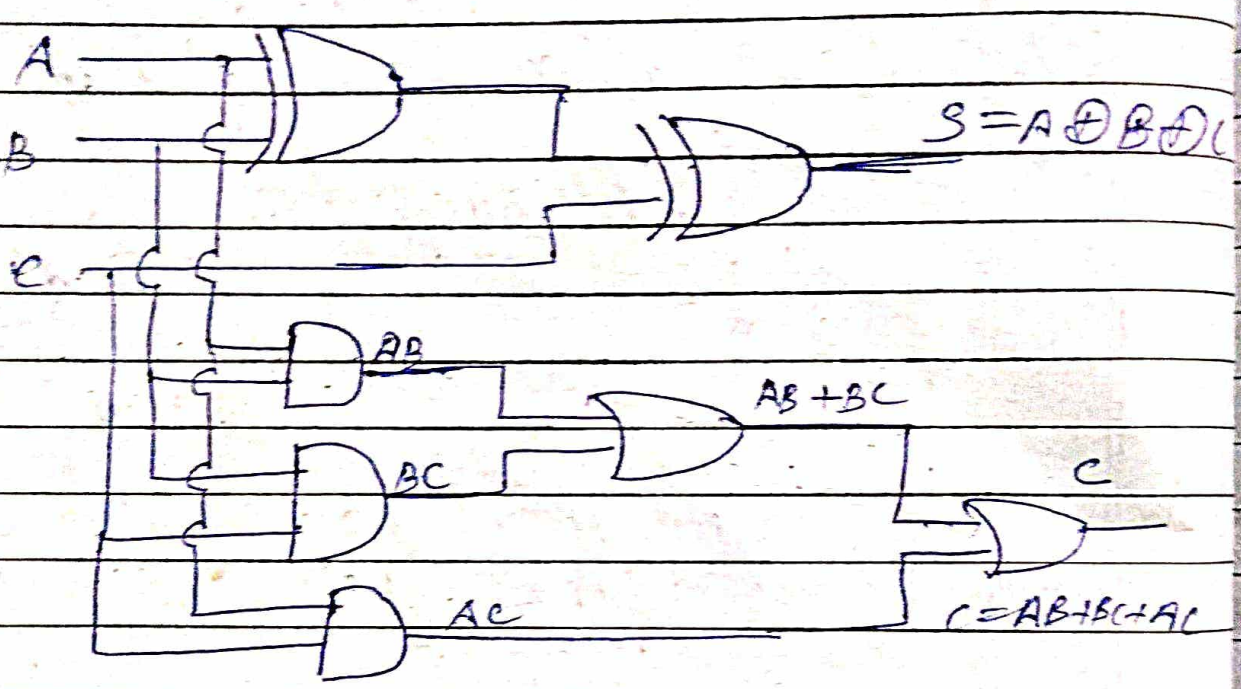
→ 1583

Q3 Implement full adder using two half adders and one OR gate.

Ans ⇒ We know that in case of full adder $S = A \oplus B \oplus C$

$C = AB + BC + AC$

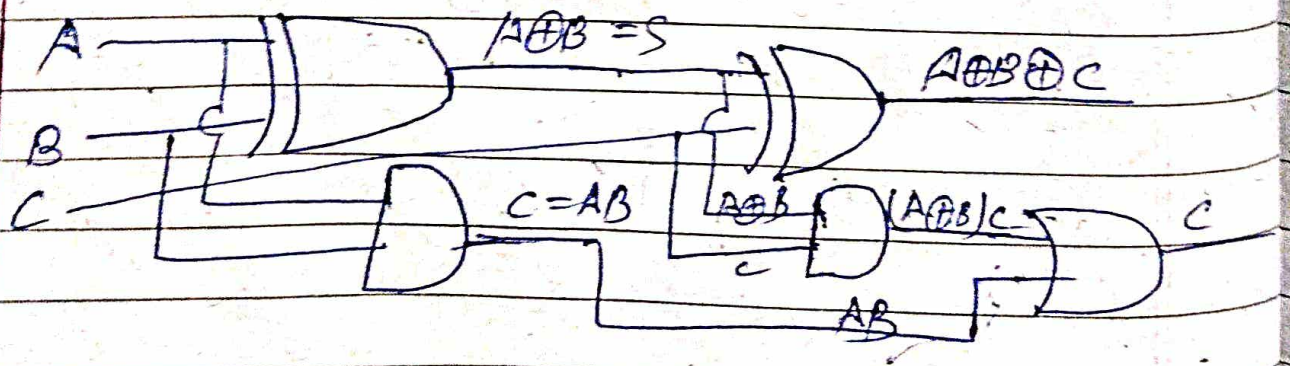
circuit diagram:



So full adder using two half adder and one OR gate

in case of half adder

$S = A \oplus B$ $C = AB$



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

Date / /
Page No.

$(A \oplus B)C + AB$

$(AB' + BA')C + AB$

$AB'C + A'BC + AB$

$A'BC + AB'C + AB$ { $\because A + A'B = A + B$

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

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

$A'BC + AB + AC$

$A'BC + AC + AB$

$AB + C(A'B + A)$ { $\because A + A'B = A + B$

$AB + C(A + B)$

$AB + A \cdot C + BC$

$AB + BC + AC$ proved

Date / /
Page No.

(Pg8) Design 4 to 16 decoder with 2 to 4 decoder

4×16 decoder using 2×4 decoder

any → I/p ↓ O/p I/p ↓ O/p

No of decoder = $\frac{\text{req. output}}{\text{output}} = \frac{16}{4} = \frac{4}{4} = 1$

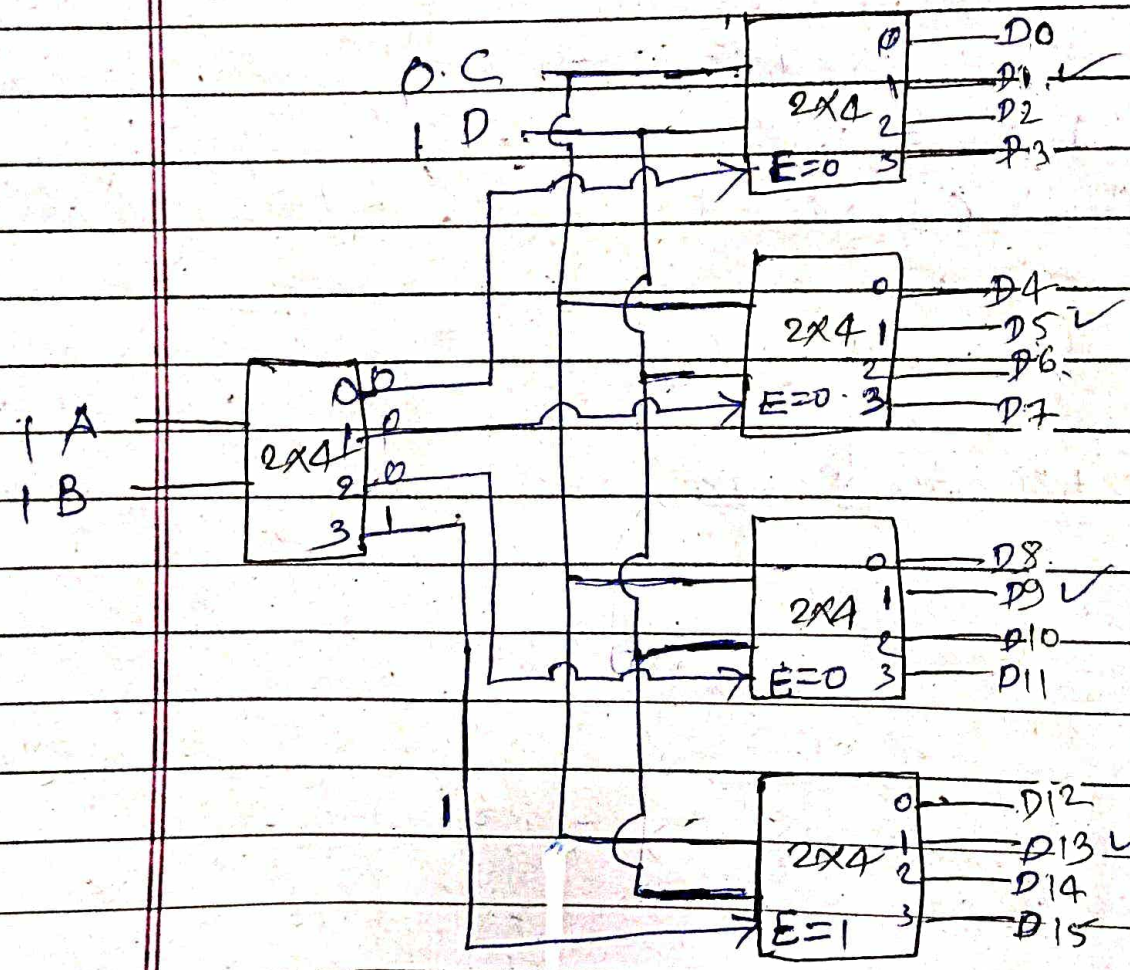


Diagram ↑↑↑

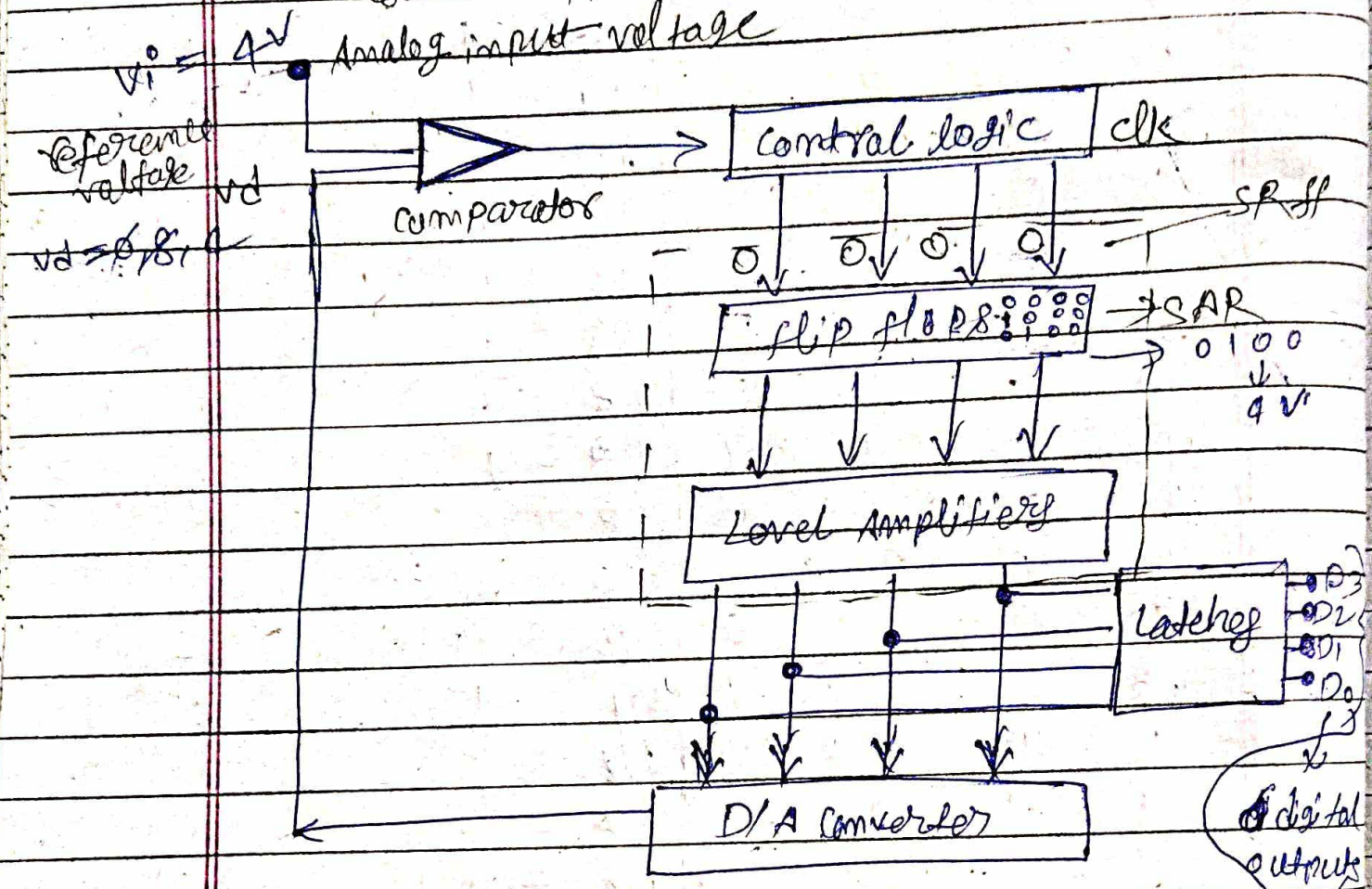
verify: A B C D
1 1 0 1

Truth table

A	B	C	D	Output
0	0	0	0	D0 = 1
0	0	0	1	D1 = 1
0	0	1	0	D2 = 1
0	0	1	1	D3 = 1
0	1	0	0	D4 = 1
0	1	0	1	D5 = 1
0	1	1	0	D6 = 1
0	1	1	1	D7 = 1
1	0	0	0	D8 = 1
1	0	0	1	D9 = 1
1	0	1	0	D10 = 1
1	0	1	1	D11 = 1
1	1	0	0	D12 = 1
1	1	0	1	D13 = 1
1	1	1	0	D14 = 1
1	1	1	1	D15 = 1

Analog to Digital

Successive Approximation A/D Converter:



if $V_i > V_d$ → MSB set

$V_i < V_d$ → MSB reset, Next bit set

$V_i = V_r$ → output

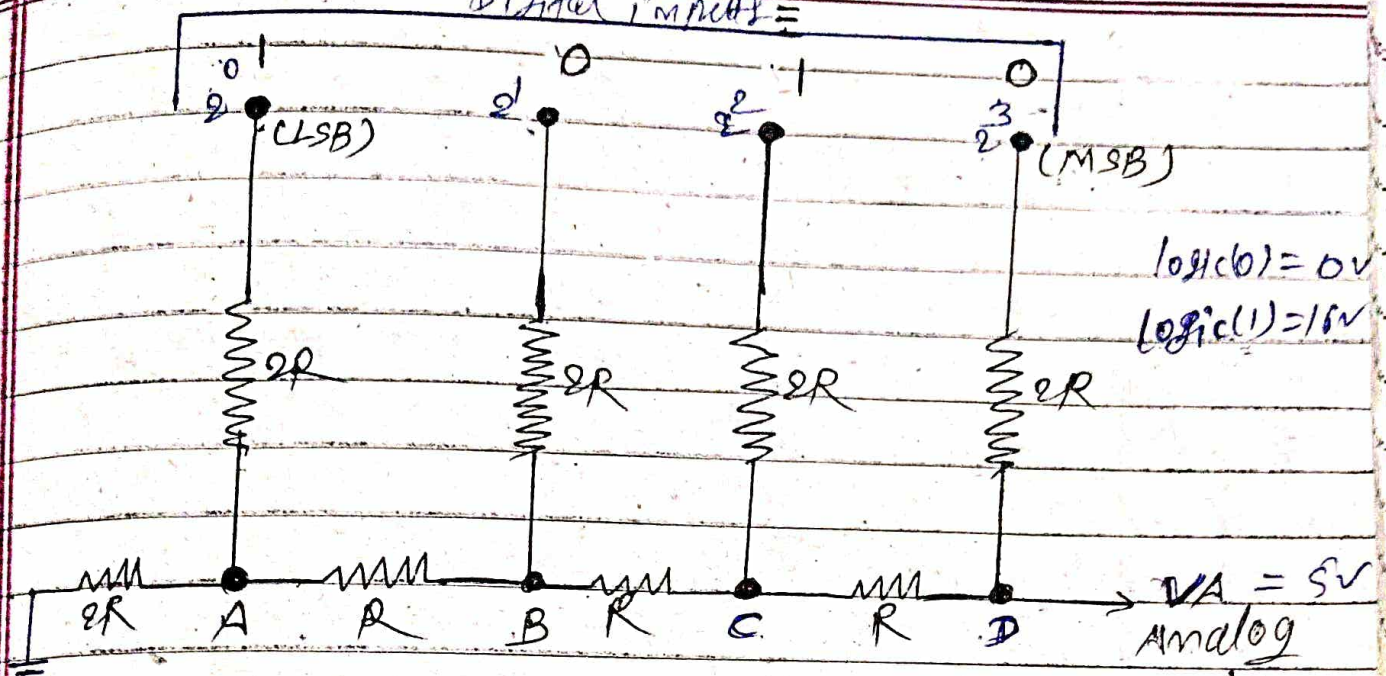
Application of A/D & D/A Converters.

A/D → ① Sensors ② Audio & Video processing

D/A → ① Audio and Video playback

② Motor Control

Digital to analog
R-2R Ladder D/A C



$$\text{formula} = \frac{V_0 \times 2^0 + V_1 2^1 + V_2 2^2 + V_3 2^3 + \cdots}{2^n} \quad \text{where } n = \text{no. of bits}$$

V_A

Let's take an example: 0 1 0 1

$$V_A = \frac{16 \times 2^0 + 0 \times 2^1 + 16 \times 2^2 + 0 \times 2^3}{2^4}$$

$$V_A = \frac{16 \times 1 + 0 \times 2 + 16 \times 4 + 0 \times 8}{16}$$

$$V_A = \frac{16 + 0 + 64 + 0}{16} \qquad V_A = \frac{80}{16} = 5V$$

$$\boxed{V_A = 5V}$$

Thus analog output is 5V for binary 0101