LEARNING MATERIAL OF

DIGITAL ELECTRONICS & MICROPROCESSOR

PREPARED BY – ER. BISWARANJAN JENA

&

ER. SASWATI SANGHAMITRA PRADHAN

Basics of Digital Electronics :-

Number System

1. Decimal Number System.
2. Binary Number System.
3. Octal Number System
4. Hexadecimal Number System

(1) Decimal Number System :-

↳ means 10. So this system has 10 distinct digits or symbols

i.e 0 1 2 3 4 5 6 7 8 9.

Eg :- $(7639)_{10}$, $(346.71)_{10}$

(10) → Base

→ In decimal number system the base is 10.

$$(7639)_{10} = 7 \times 1000 + 6 \times 100 + 3 \times 10 + 9.$$

$$= 7 \times 10^3 + 6 \times 10^2 + 3 \times 10^1 + 9 \times 10^0$$

Positional weight of the decimal number system.

In general any number in decimal number system can be written as,

$$N = a_n \times 10^n + a_{n-1} \times 10^{n-1} + \cdots + a_2 \times 10^2 + a_1 \times 10^1 + a_0 \times 10^0 + a_{-1} \times 10^{-1} + \cdots + a_{-m} \times 10^{-m}$$

$$(346.71)_{10} = 3 \times 100 + 4 \times 10 + 6 \times 1 + 7 \times 10^{-1} + 1 \times 10^{-2}$$

→ The power raised to 10 depends on the position of coefficient.

→ The positional power raised to 10, which is known as the radix or base of this decimal number system.

General form of any number system can be written as,

$$N = a_n \times (r)^n + a_{n-1} \times (r)^{n-1} + \cdots + a_1 \times (r)^1 + a_0 \times (r)^0 + a_{-1} \times (r)^{-1} + a_{-2} \times (r)^{-2} + \cdots + a_{-m} \times (r)^{-m}$$

$r \rightarrow$ Radix of the number system.

The weighted coefficient are a_n to a_{-m}

a_{-m} → coefficient is called LSD (Least Significant Digit).

a_n → is known as the most significant digit (MSD).

(2) Binary Number System :-

→ Binary number system consists of 2 distinct elements or digits i.e 0 & 1 only.

→ Base or radix of the number system is 2.

→ The digit 0 & 1 is known as bits.

e.g :- $(1011.01)_2$, $(1010.101)_2$

bit

The binary numbers are pronounced in the following manner.

'0' is pronounced as "zero"

'1' is " " "one".

'10' is " " one zero not ten.

'11' is " as one one not Eleven.

ing of the Binary Number :-

Digital Number	Binary Number
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
17	10001
18	10010
19	10011
20	10100
21	10101

4 3 2 1 0 ⟶ position of bits.

e.g :- $(10110)_2$

MSB ↑ ⟶ LSB

$(10110)_2 = 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1$

$= 16 + 4 + 2 = (22)_{10}$

→ It is very essential to show the suffix to the numbers which indicates the base of the number system.

Q) Find the decimal equivalent of the binary number $(11011001.0101)_2$.

Sol^n :- $(\overset{7\ 6\ 5\ 4\ 3\ 2\ 1\ 0\ \ -1\ -2\ -3\ -4}{1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ .\ 0\ 1\ 0\ 1})_2$

$\Rightarrow 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^0$

$+ 1 \times 2^{-2} + 1 \times 2^{-4}$

$\Rightarrow 128 + 64 + 16 + 8 + 1 + \frac{1}{4} + \frac{1}{16}$

$\Rightarrow (217.3125)_{10}$

Practice Question :-

① $(11)_2$, ② $(101)_2$, ③ $(010)_2$

④ $(10011)_2$, ⑤ $(1010)_2$

⑥ $(1010.11)_2$, ⑦ $(1011.01)_2$, ⑧ $(111.11)_2$

Octal number System :-

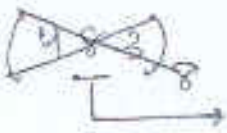
The base or, radix of the octal number system is 8. (Octal means 8).

-> Digits or elements will be,

0 1 2 3 4 5 6 7.

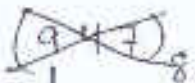
Decimal	Octal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	10
9	11
10	12
11	13
12	14
13	15
14	16
15	17

e.g :- $(246)_8$ ✓



→ Due to the presence of '8'. This is not an octal number system

$(736)_8$ ✓



→ not an octal number.

Q:- The decimal equivalent of octal number is _____

Ans:- $(24)_8 = 2 \times 8^1 + 4 \times 8^0$

$= 16 + 4 = (20)_{10}$

Q) $(7126.45)_8 = (?)_{10}$

Soln:- $\overset{3}{7}\overset{2}{1}\overset{1}{2}\overset{0}{6}.\overset{-1}{4}\overset{-2}{5}$ $= 7 \times 8^3 + 1 \times 8^2 + 2 \times 8^1 + 6 \times 8^0 + 4 \times 8^{-1} + 5 \times 8^{-2}$

$= 512 \times 7 + 64 \times 1 + 16 + 6 + 4 \times 0.125 + 5 \times 0.0156$

$= 3584 + 64 + 16 + 6 + 0.5 + 0.078$

$= (3670.578)_{10}$

Practice:-

(i) $(37)_8$ (ii) $(311)_8$ (iii) $(125.7)_8$

(iv) $(100.21)_8$ (v) $(217.31)_8$

Hexadecimal Number System :-

→ In this system the radix or base is "16".

→ It consists of 16 distinct symbol or element.

0 1 2 3 4 5 6 7 8 9 A B C D E F

↓ ↓ ↓ ↓ ↓ ↓

10 11 12 13 14 15

Decimal Number System	Hexadecimal number System
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10
17	11
18	12

eg :- $(ABC)_{16}$, $(123)_{16}$, $(12A)_{16}$

$(943)_{16}$:

fractional :- $(AB1.CD)_{16}$, $(123\ A)_{16}$.

→ Decimal Equivalent of Hexadecimal numbers,

$$(\overset{2}{A}\ \overset{1}{B}\ \overset{0}{C})_{16} = A \times 16^2 + B \times 16^1 + C \times 16^0$$

$$= 10 \times 16^2 + 11 \times 16^1 + 12 \times 16^0$$

$$= 2560 + 176 + 12$$

$$= (2748)_{10}$$

eg :- $(A0F9.0EB)_{16} = 10 \times 16^3 + (0 \times 16^2) + (15 \times 16^1) + (9 \times 16^0) + (0 \times 16^{-1}) + (14 \times 16^{-2}) + (11 \times 16^{-3}) = (41209.0572)_{10}$

Convention of Number system :-

Binary → Decimal
Octal → Decimal.
Hexadecimal → Decimal.

positional weight method.

Decimal → Binary
Decimal → Octal.
Decimal → Hexadecimal

Long Division method.

→ Binary to Octal
→ Binary to Hexadecimal
→ Octal to Binary
→ Octal to Hexadecimal
→ Hexa to Binary
→ Hexa to Octal

Integer conversion

Decimal → Binary :-

Q) $(35)_{10} = (?)_2$

A)

2	35	
2	17	— 1
2	8	— 1
2	4	— 0
2	2	— 0
2	1	— 1
	0	

$(35)_{10} = (10011)_2$

Q:- $(127)_{10} = (?)_2$ Ans :- $(1111111)_2$

Decimal → Octal :-

(i) $(567)_{10}$ (ii) $(1276)_{10}$.

Q) $(567)_{10} = (?)_8$

Ans>

8	567	
8	70	— 7
8	8	— 6
8	1	— 0
	0	— 1

$(567)_{10} = (1067)_8$

(ii) $(1276)_{10} = (?)_8$

Ans:-

8	1276	
8	159	4
8	19	7
8	2	3
	0	2

$(1276)_{10} = (2374)_8$

Decimal to Hexadecimal :-

e.g:- $(2598.675)_{10}$

Ans:-

16	2598	
16	162	6
	10	2
	0	A

$0.675 \times 16 = 10.8$
$0.8 \times 16 = 12.8$
$0.8 \times 16 = 12.8$

$(A26.ACCC)_{16}$

Application :-

→ Since the base of of octal number system is $8 = 2^3$, so every 3-bit group of binary can be represented by an octal digit.

→ An octal number is thus $\frac{1}{3}^{rd}$ the length of the corresponding binary number

→ In computer work binary numbers with up to 64-bits are not uncommon.

→ These binary numbers do not always represent a numerical quantity, they often represent some type of code which conveys non-numerical information.

A binary number might represent

(a) the actual numerical data

(b) the number corresponding to a location (address) in memory.

(c) an instruction code.

(d) a code expressing alphabetic & other non-numerical other non numerical character

(e) a group of bits representing the status of devices internal or external to devices

→ We use octal only for the convenience of the operations of the system.

→ The digital circuit and systems work strictly in binary

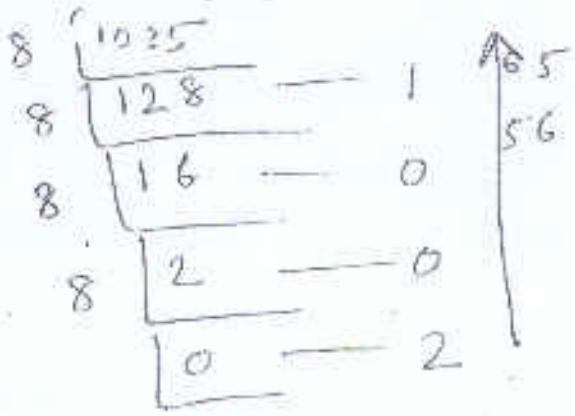
→ A 4-bit group is called nibble.

→ Since computer words come in 8 bit, 16 bit, 32-bit and so on, that is, multiples of 4-bits, they can be easily represented in hexadecimal.

→ Hexadecimal system is particularly useful for human communications with computers.

Hexadecimal :-

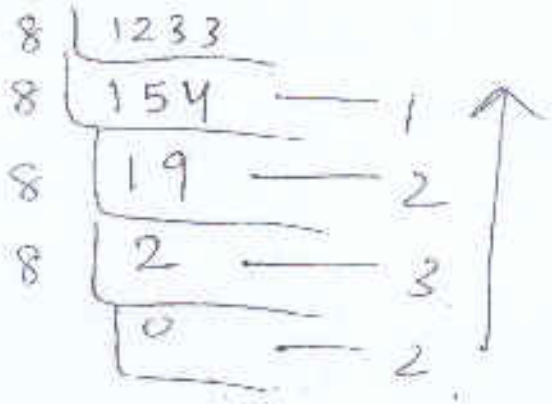
→ Since the base is $16 = 2^4$, every 4 binary digit combination can be represented by one hexadecimal digit



M3

74

1233 202



$0.202 \times 8 = 1.616$

$0.616 \times 8 = 4.928$

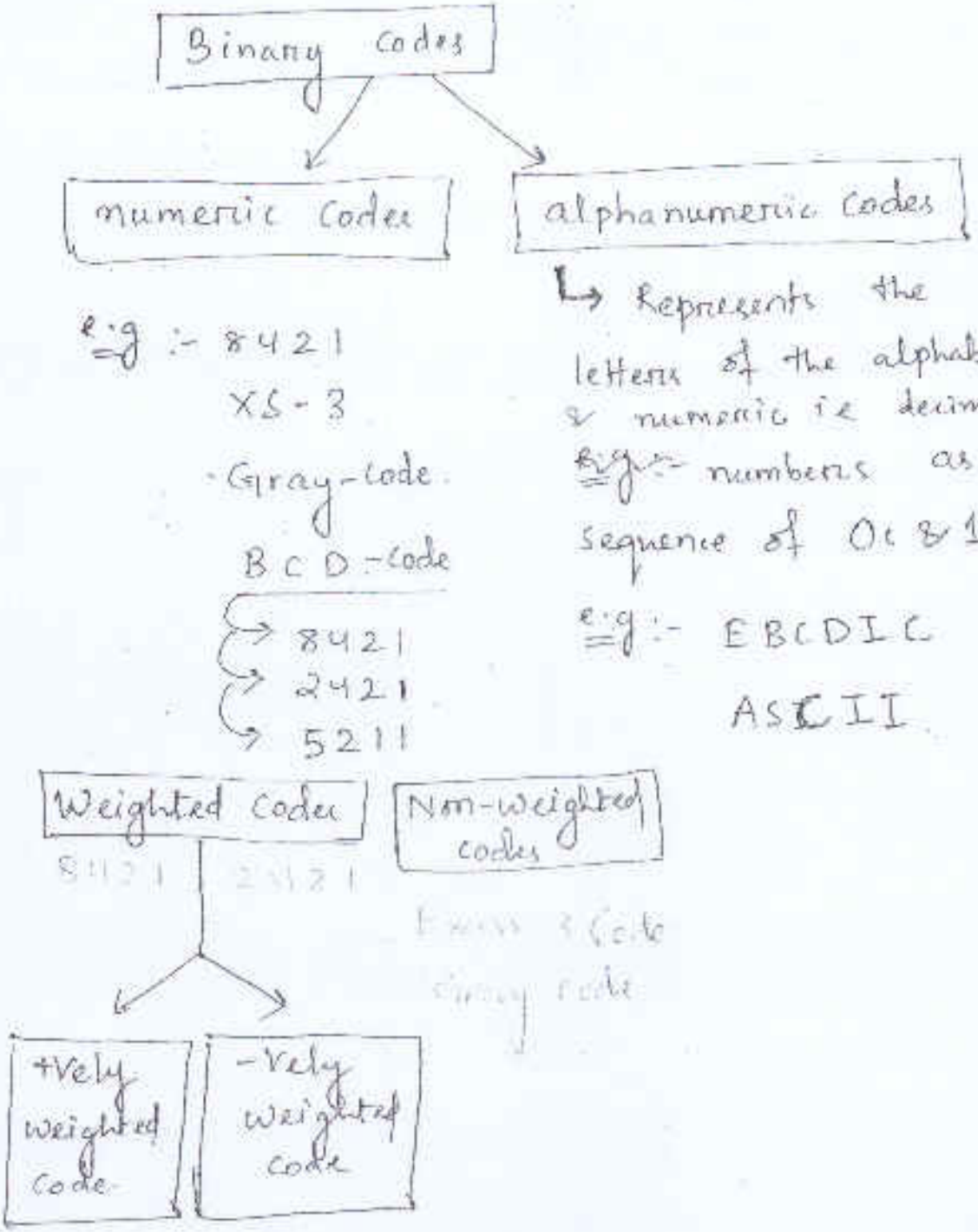
0

7.424

3.392

3.136

BINARY CODES



Positively-weighted codes:-

Positively-weighted codes are those in which all the weights assigned to the binary digits are positive.

→ There are only 17 positively weighted codes.

→ In every +vely weighted codes, the first weight must be 1, 2nd weight must be either 1 or 2, and the sum of the weights must be equal to or greater than '9'.

e.g:- 8421, 2421, 5211, 3321, 4311.

Negatively weighted codes:-

Here some of the weightes assigned to the binary digits are negative.

e.g:- 642-3, 631-1, 84-21, 74-2-1

Error Detecting and Error Correcting codes:-

→ The codes which allow only error detection are called error detecting codes.

e.g:- shift counter code
2 out-of 5, 63210 codes } Error Detecting codes

Error correcting codes :-

Codes which allow error detection as well as error correction are called error correcting codes.

e.g :- Hamming codes

Sequential codes :-

A sequential code is one, in which each succeeding code word is one binary number greater than it's preceeding code word. Such a code facilitates mathematical manipulation of data.

e.g :- 8421, XS-3 are Sequential.
5211, 2421 and 642-3 are non-sequential.

Self complementing codes :-

For a code to be self-complementing the sum of all it's weights must be 9.

e.g :- 2421, 5211, 3321, 4311
↳ Self complementing positively weighted codes.

→ There are 13 negatively-weighted self-complementing codes.

Cyclic Codes :-

Cyclic Codes are those in which each successive code coord differs from the preceding code in only one bit position. They are also called unit distance codes.

→ Advantage :- They minimize transitional errors or flashing.

→ Gray Code is a cyclic Code.

Reflective Codes :-

A reflective code is a binary code. in which the 'n' least significant bits for code coords 2^{nd} through $2^{n+1}-1$ are the mirror image of those for '0' through 2^n-1. The gray code is a reflective code.

1's Complement Representation :-

The 1's complement of a binary numbe obtained by converting each 0 bit of th binary number to a 1, and each 1 b by a '0'.

Q :- find the 1's complement of th binary number 1011101 is ?

Ans :- $1011101 \xrightarrow{1's} 0100010$

2's Complement Representation :-

The 2's complement of a binary numbe is obtained by taking 1's complement of the number and adding '1' to the least significant bit position.

Q :- find the 2's complement of $(25)_{10} = (11001)_2$ ~~is given~~

Ans :-

$11001 \xrightarrow[\text{complement}]{1's} 00110$

$+ \quad 1$

00111

The 2's complement of $(11001)_2$ is (0

Other method :-

The another method of obtaining the 2's complement of a binary number is to scan the number from Right to ~~left~~ and complement all bits appearing after the 1st scan of a '1'.

1 1 0 0 1

(L) ↓ ← (R)

0 0 1 1 1 → Ans.

e.g:-

$(42)_{10} = (101010)_2 \xrightarrow{2\text{'s complement}} 010110$

1st one

	1's	2's
-7 →	1111 →	1000 1001

54

16 | 156

16 | 9 — 12 (C)

0 — 9

16 / 5 / 80

16 / 9 / 134

8: 16 | 313

16 | 19 — 9

16 | 1 — 3

0 — 1

31 / 16 / 15

16 / 9 / 144

160

156 / 144 / 12

16 | 313 | 19

16 / 153 / 144 / 9

41209 . 0572

16 | 41209
16 | 2575 — 9
16 | 160 — 15 (E)
16 | 10 — 10 (A
0

16 | 41209 / 2575
32
92
80
120
112
89
80
9.

0572 × 16 =

16 | 2575) 160
16
97
96
1 | 60
0
1525
15

(AEC2)
(1010 1110 1100 0010)
(1 2 7 3 0 2)8

(DF2F . AAE)
(1101 1111 0010 1111 . 1010 1010 1110)
(1 5 7 4 5 7 5 2 5 6)

Representation of Signed Number using 2's (or 1's) Complement method.

(1) If the number is positive, the magnitude is represented in it's true binary form and a sign bit '0' is placed in front of the MSB.

(2) If the number is negative, the magnitude is presented in it's 2's (or 1's) complement form and a sign bit 1 is placed in front of the MSB.

→ The 2's (or 1's) complement operation on a signed number will change positive number to a negative number and vice versa.

→ The conversion of complement to true binary is the same as the process used to convert true binary to complement.

e.g.- $+51 = \boxed{0 \mid 1\,1\,0\,0\,1\,1}$

sign bit magnitude

$-51 = \boxed{1 \mid 1\,1\,0\,0\,1\,1}$

$-51 = \boxed{1 \mid 0\,0\,1\,1\,0\,1}$

(2's complement form.)

2 | 51	
2 | 25	— 1
2 | 12	— 1
2 | 6	— 0
2 | 3	— 0
2 | 1	— 1
0	— 1

-51 = | 1 | 00 1100 |

(1's complement form)

1 → Sign bit

00 1100 → Magnitude

Q:- Each of the following numbers is a signed binary number. Determine the decimal value in each case. if they are in,

(i) Sign-magnitude form.

(ii) 2's complement form

(iii) 1's " "

Ans:- (a) 01101 (b) 010111 (c) 10111

(d) 1101010.

Given Number	Signed-mag. form	2's	1's
01101	+13	+13	+13
010111	+23	+23	+23
10111	−7	−9	−8
1101010	−42	−22	−21

→ 101010 → 010101

1

1010110

0010

1

0011

01101 —2's→

0111 → 1000

1

1001

To Subtract using 2's (or 1's) complement method :-

(1) To subtract using 2's complement method represent both the subtrahend and the minuend by the same number of bits.

(2) Take the 2's complement of the subtrahend including the sign bit. Keep the minuend in it's original form and add the 2's (or 1's) complement of the subtrahend to it.

→ When the sign bit is a '0', the remaining bit represent magnitude
When the sign bit is '1', the remaining bits represent 2's or 1's complement of the number.

→ The polarity of the signed number can be changed simply by performing the complement on the complete number.

Methods of obtaining the 2's complement of a number:-

(1) By obtaining 1's complement of the given number (by changing all 0s to 1s and 1s to 0s) and then adding 1.

(2) By subtracting the given n-bit number N from 2^n.

(3) Starting at the LSB, copying down each bit up to and including the first 1 bit encountered and complementing the remaining bits.

(Q.1) Express -45 in 8-bit 2's complement from.

Ans:- $(\textcircled{1}\textcircled{0}1\,0\,1)_2 = 45_{10}$

~~+45 = 0 0 0 1 0 1 0 1~~

$+45_{10} = (0\,1\,1\,0\,1)_2$

$+45 = 0\,0\,0\,0\;1\,1\,0\,1$

2 | 45
2 | 22 — 1
2 | 11 — 0
2 | 5 — 1
2 | 2 — 01
2 | 1 — 0
0

2's Complement Arithmetic :-

The 2's complement system is used to represent negative numbers using modulus arithmetic.

→ The word length of a computer is fixed. That means, if a 4-bit number is added to another 4 bit number the result will be only of 4 bit. Carry if any from the fourth bit will overflow. This is called the modulus arithmetic

eg :- 1100
+ 1111
---
(1) 1011 Ans

Q) Subtract 14 from 46 using 8 bit 2's complemented arithmetic.

Ans :- +14 = 0 0 0 0 1 1 1 0
-14 = 1 1 1 1 0 0 0 1 (1's)
-14 = 1 1 1 1 0 0 1 0 (2's)

+46 = 0 0 1 0 1 1 1 0
-14 = 1 1 1 1 0 0 1 0 (2's complement form of 14 is -14)
---
(1) 0 0 1 0 0 0 0 0

ignore the carry

Q.2) Subtract 2 from 5 using 4-bit 2's complement arithmetic.

Ans:-

$+5 = 0101$

$+2 = 0010$

$-2 = 1101$ (1's)

$-2 = 1110$ (2's)

$$\begin{array}{r} +5 = 0101 \\ +\ -2 = 1110 \\ \hline (1)0011 = (+3) \end{array}$$ Ans

ignore.

Q.3) Subtract -2 from 5 using 4 bit 2's complement arithmetic.

Ans:-

$+5 = 0101$

$-2 = 1010$

$+2 = 0101$ (1's)

$+2 = 0110$ (2's)

$5-(-2) = (7)$

$2 = 10$

$-2 = 110$

$-2 = 0110$

$+2 = 1001$

$+\ \ 1$

1010

$$\begin{array}{r} 5 \\ -\ -2 \\ \hline (7) \end{array}$$

$$\begin{array}{r} 0101 \\ 0110 \\ \hline 1011 \end{array}$$

$$\begin{array}{r} 0101 \\ +\ 1010 \\ \hline 1111 \end{array}$$

$-2 = 1110$ (2's complement)

$+2 = 0001$

0010

101

$+1$

$\Rightarrow (-7)$

0101

$+10$

Q.4) Add -75 to +26 using 8-bit 2's complement arithmetic.

Ans:-

$-75 + 26 = -49$

$+75 = 01001011$

$-75 = 10110101$ (2's)

$+26 = 00011010$

$-75 = +10110101$

10011111

No carry

$+15 + 128 = 143$

magnitude of 2's complement form is

$= 01001111$

$01110000 + 1 = 01110001$

$32 + 16 + 1 = 49$

-49

2	75
2	37 —
2	18 —
2	9 —
2	4 —
2	2 —
2	1 —
	0 —

Logic Gates

1. AND
2. OR
3. NOT

} Basic Gates

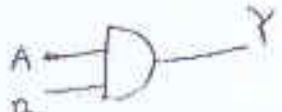
4. NOT (INVERTER)
5. NAND
6. NOR

} universal gate

7. XOR
8. XNOR
9. Buffer gate → The IC 7408 contains 4 i/p AND gates

(1) AND Gate :-

symbol

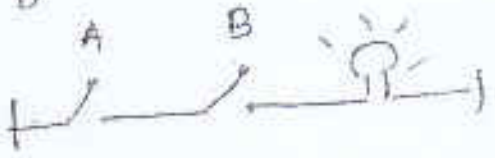


Truth Table

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

2 i/p

3 i/p



AND operation in Boolean algebra is similar to multiplication in ordinary algebra

(2) OR gate :-

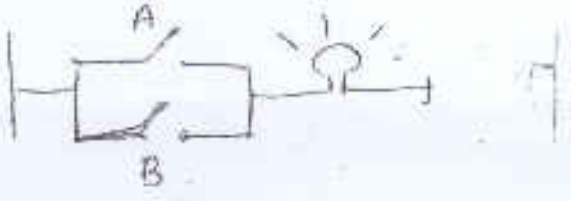
2 input

How many combination ?

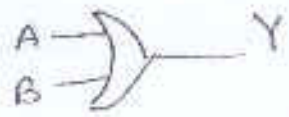
$2^n = 2^2 = 4$ combinations

T.T

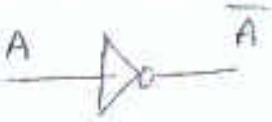
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



Symbol of the logic gate



NOT :-

Symbol : 

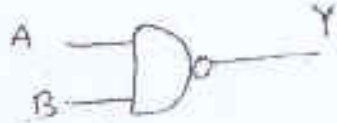
$Y = \overline{A}$

T.T

A	Y
0	1
1	0

→ The NOT operation in boolean algebra is nothing but complementation or inversion

NAND :-

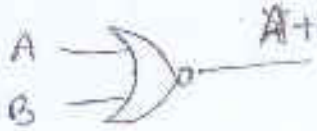


$Y = \overline{A \cdot B}$

Truth Table

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

NOR :-



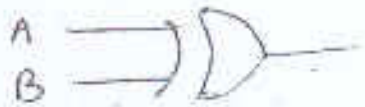
$\overline{A+B} = Y$

Truth Table

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

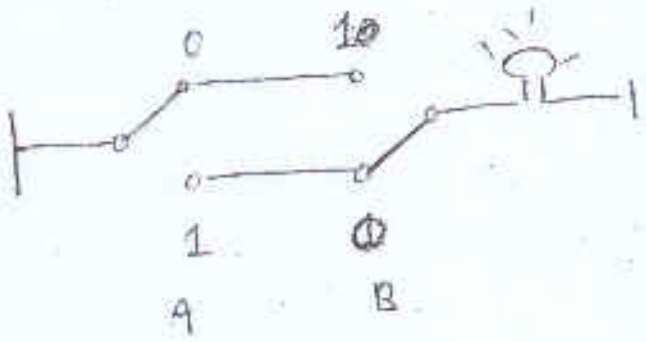
→ All the gates can be designed using the NAND & NOR gate. So this is called Universal gate.

XOR Gates :-



Truth Table

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



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

XNOR Gates :-

Truth Table :-

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

① Logic Gates are the fundamental building block of digital system.

AND

→ The IC 7408 contains four i/p AND gates.

→ The IC 7411 contains 3-i/p AND gates

→ IC 7421 contains 4-i/p AND gates.

What is Logic Gates?

Ans:- Logic gates are electronic circuit because they are made up of a number of electronic devices and components

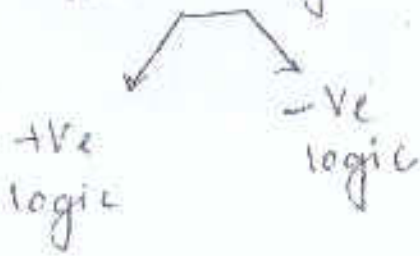
→ They are usually embedded in a large scale integrated circuit (LSI).

→ Very large scale integrated circuit (VLSI) along with a large number of other devices.

→ Each gate is dedicated to a specific logic operation

→ Logic gates are also constructed in (a) SSI (b) MSI (c) VLSI.

~~Logic levels~~ Level Logic



→ Voltage levels represents Logic 1 and Logic 0.

→ +ve logic → 0V → Logic 0
15V → Logic 1

→ -ve logic → Lower of the two voltages represents the Logic 1
→ Higher of the two voltages represents the logic 0

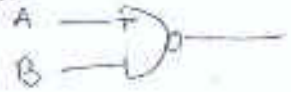
OR: IC-7432 contains four two-i/p OR gates.

NOT:- IC 7404 contains six inverters.

Universal Gate (NOR, NAND) :-

→ Using NAND and NOR gates we can perform all the three basic logic function.

NAND



→ IC -7400 contains 4 two-i/p NAND gates.

IC-7410 contains 3 three-i/p NAND gates

IC-7420 contains 2 four-i/p NAND gates

IC-7430 " 1, 8-i/p NAND gate

NOR Gates :-

NOR means NOT + OR

→ The o/p is 'I' high when both the i/ps are low (0). If any of the i/p is high the o/p is zero.

→ For 3 i/p case also if all the i/ps are '0' low then the o/p is high otherwise the o/p is zero.

→ The I.C-7402 contains four two i/p NOR gates.

→ The IC-7427 contains three 3-i/p NOR gate.

→ IC 7425 contains two 4-i/p NOR gates.

XOR gate :-

o/p is logic-1 state when one and only one of its two i/ps assumes a logic-1 state. When both the i/ps are logic-0 or both the i/ps logic-1 state the o/p assumes logic-0 state.

→ Since an X-OR gate produces an o/p-1 only when the i/ps are not equal, it is called as anti-coincidence gate or inequality detector.

→ Three or more variable x-OR gates do not exist.

→ TTL IC-7486 contains 4 x-OR gates.

→ IC-74C86 " 4 XOR gates.
↓
CMOS

→ High Speed CMOS IC 74HC86 contains 4-XOR gates

X-NOR :-

→ In equality detector.

→ comparative gate

→ IC 74LS266

IC 74C266.

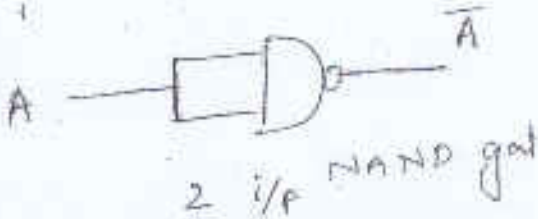
IC 74HC266 contains four each X-NOR gates.

→ **Realize AND, OR, NOT operations using NAND & NOR gates :-**

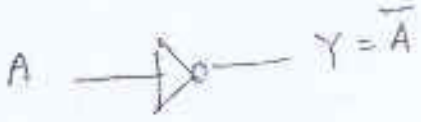
AND :-

Q) Design a ~~AND~~ NOT gate using NAND gate?

Ans :-



2 i/p NAND gate



Boolean Algebra :-

AND Law	OR Law
$A \cdot 0 = 0$ (Null Law)	$A + 0 = A$
$A \cdot 1 = A$ (Identity Law)	$A + 1 = 1$
$A \cdot A = A$	$A + A = A$
$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$

Commutative Law :-

Law-1 $A + B = B + A$, $A + B + C = B + C + A = C + A + B = B + A + C$

Law-2 $A \cdot B = B \cdot A$, $A \cdot B \cdot C = B \cdot C \cdot A = C \cdot A \cdot B = B \cdot A \cdot C$

Associative Law :-

The associative laws allows grouping of the variables. There are two associative law.

Law-1 $(A + B) + C = A + (B + C)$

Law-2 $(A \cdot B) C = A (B \cdot C)$

$A (BCD) = (ABC) D = (AB) (CD)$

Distributive Law :-

Law-1 : $A (B + C) = AB + AC$

The distributive laws allows factorising or multiplying out of expressions.

$ABC(D+E) = ABCD + ABCE$.

$AB(CD+EF) = ABCD + ABEF$

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

Law-2 :-

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

Redundant Literal Rule (RLR) :-

Law-1 : $A + \bar{A}B = A + B$

Law-2 :- $A(\bar{A}+B) = AB$

Idempotence laws :-

Law-1 :- $A \cdot A = A$

Law-2 : $A + A = A$

Absorption law :-

Law-1 : $A + A \cdot B = A$

Law-2 : $A(A+B) = A$

→ $A + A \cdot$ any term $= A$

→ $A($A + Any term$) = A$

Induced factor Theorem :-

Theorem 1 : $AB + \bar{A}C + BC = AB + \bar{A}C$

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

This theorem can be extended to any number of variables.

eg :- $(A+B)(\bar{A}+C)(B+C+D) = (A+B)(\bar{A}+C)$.

Transposition Theorem :-

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

Demorgan's Theorum :-

Law -1 : $\overline{A+B} = \bar{A}\bar{B}$

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

Law -2 : $\overline{AB} = \bar{A}+\bar{B}$

$\overline{ABCD\cdots} = \bar{A}+\bar{B}+\bar{C}+\bar{D}+\cdots$

1. Complement the entire given function
2. Change all the ANDs to ORs and all the ORs to ANDs.
3. Complement each of the individual variables
4. Change all 0s to 1's and 1's to 0's.

The procedure is called demorganisation or complementation of switching expressions.

$$f(A, B, C, \cdots, 0, 1, +, \cdot)_c = f(\overline{A}, \overline{B}, \overline{C}, \cdots 1, 0, \cdot, +)$$

Shannon's Expansion Theorem :-

$$f(A, B, C, \cdots) = A \cdot f(1, B, C, \cdots) + \overline{A} f(0, B, C \cdots)$$

$$f(A, B, C, \cdots) = [A + f(0, B, C \cdots)] + [\overline{A} + f(1, B, C, \cdots)]$$

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

Q) $f = \overline{AB}(CD + \overline{EF})\overline{(\overline{AB} + \overline{CD})}$

Q) Reduce the expression

$$f = \overline{\overline{AB} + \overline{A} + AB}$$

Q) Reduce the expression

$$f = A[B + \overline{C}\ \overline{(AB + A\overline{C})}]$$

Boolean functions and their representation :-

1. SOP (Sum of products) form.
2. POS (Product of Sum) form
3. Truth Table form.
4. Standard sum-of-products form
5. Standard product-of Sum form.
6. Venn diagram form.
7. Octal Designation.
8. Karnaugh map.

(1) SOP :-

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

(2) POS :-

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

Canonical Sum of product form,

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

→ The product term which contains all the variables of the function either in complemented or uncomplemented form is called a minterm

→ The minterms are often denoted as m_0, m_1, m_2

$f(A,B,C)$

for a 3-variable function $m_0 = \bar{A}\bar{B}\bar{C}$

$m_1 = \bar{A}\bar{B}C$

$m_2 = \bar{A}B\bar{C}$

$m_3 = \bar{A}BC$

$m_4 = A\bar{B}\bar{C}$

$m_5 = A\bar{B}C$

$m_6 = AB\bar{C}$

$m_7 = ABC$

e.g:- $f(A,B,C) = m_1 + m_2 + m_3 + m_5$

→ These are the decimal codes of the ~~function~~ for which minterm for which $f = 1$.

$$f(A,B,C) = \sum m(1,2,3,5)$$

where $\sum m$ represents the sum of all the minterm whose decimal codes are given in the parenthesis.

→ It is also called as Expanded ~~form~~ product of sums form or Canonical Products of Sum form.

→ This is derived by considering the combinations for which $f = 0$.

→ Each term is a sum of all variables

→ A variable appears in uncomplemented form if it has a value of 0 in the combination and appears in complemented

form if it has a value of 1 in the combination

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

$\bar{A} = 0$

$A = 1$

$B = 0$

$C = 0$

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

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

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

The sum term which contains each of 'n' variables in either complemented or uncomplemented form is called maxterm.

→ The product of maxterm corresponding to the row for which $f = 0$.

This is standard or canonical SOP form.

→ Maxterm represented as $M_0, M_1, M_2, M_3 \ldots$

$f(A,B,C) = M_0 \cdot M_4 \cdot M_6 \cdot M_7$

$f(A,B,C) = \prod M(0,4,6,7)$

$\prod$ represents the product of all maxterms whose decimal code is given within the parenthesis

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

$$M_0 = (\overline{A}+\overline{B}+\overline{C}) = 0$$

$$M_4 = (\overline{A}+\overline{B}+\overline{C}) = 0$$

$$0 + 0 + 0$$

$A = 0$

$B = 1$

$C = 1$

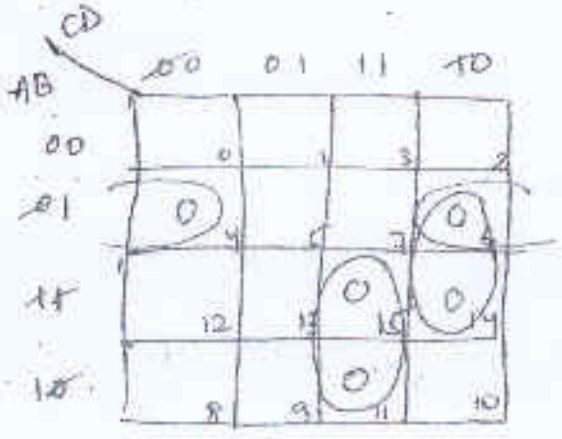
$$M_0 = \overline{A}\,\overline{B} \qquad m_1 = \overline{A}B$$

$$M_0 = (A+B) \qquad M_1 = A+\overline{B}$$

$A=0, B=0$

K- map (POS) :-

$$f = \Pi M(4,6,11,14,15)$$



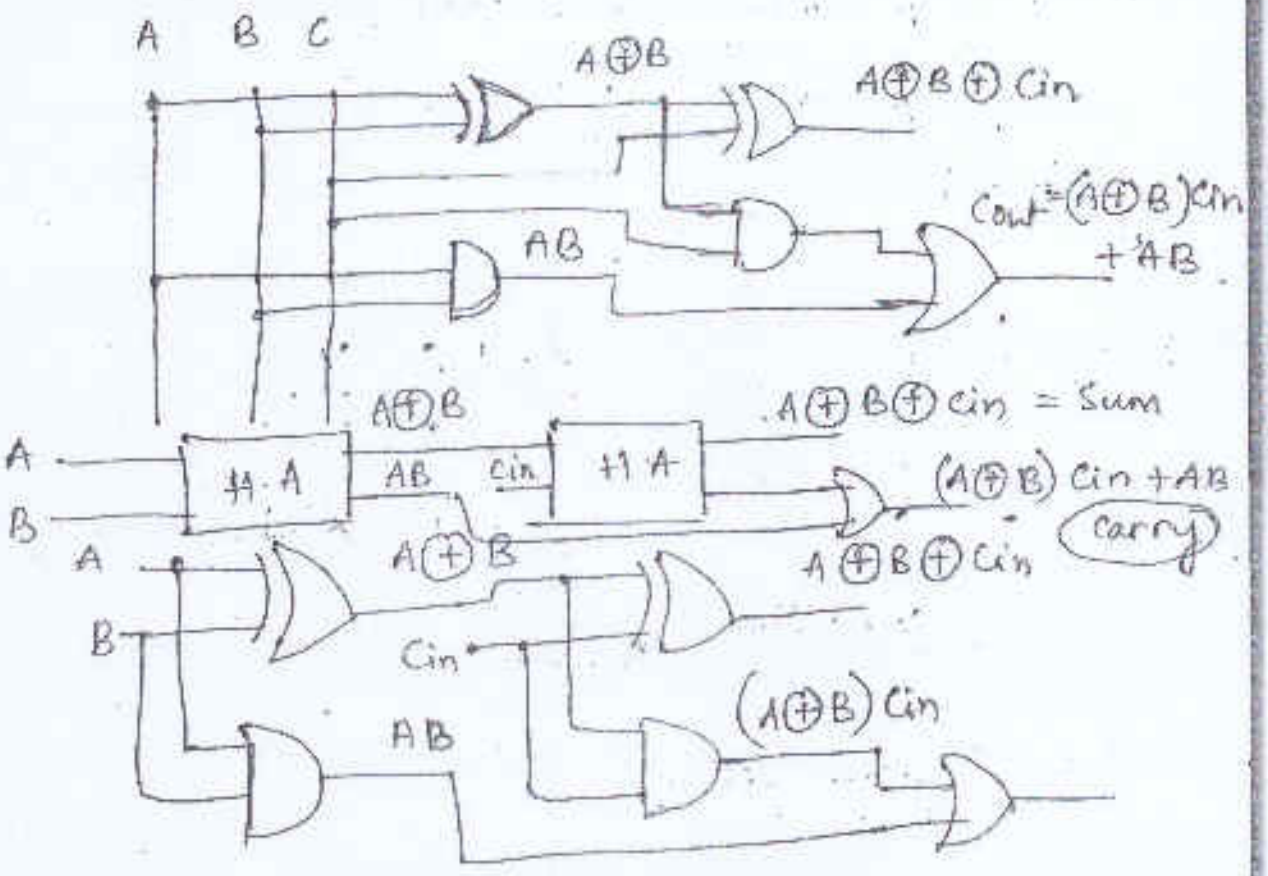
$\overline{A}B\overline{D} + AC\overline{D} + BC\overline{D}$

Design full adder using two half adder an OR gate & write truth table.

full Adder :-

Sum = $A \oplus B \oplus C$

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



Operation of multiplexur :- (Data Selector)

→ multiplexur means sharing.

→ 2 types of multiplexing → ① time multiplex
② frequency multiplex

Def^n :- A multiplexer (MUX) or data selector is a logic circuit that accepts several data inputs and allows only one of them

at a time to get through to the o/p.

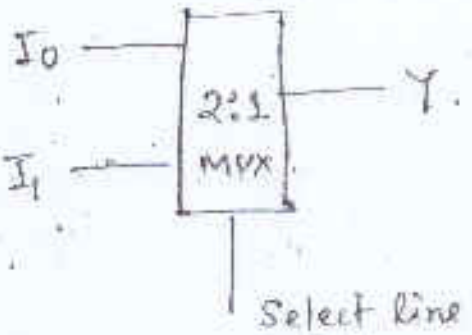
→ The routing of the desired data i/p to the o/p is controlled by selector i/p (Sometimes referred to as Address i/ps).

→ MUX acts like a digitally controlled multiposition switch.

① 2 i/p mux (2:1 MUX)

② 4 i/p mux (4:1 MUX)

2:1 (mux)



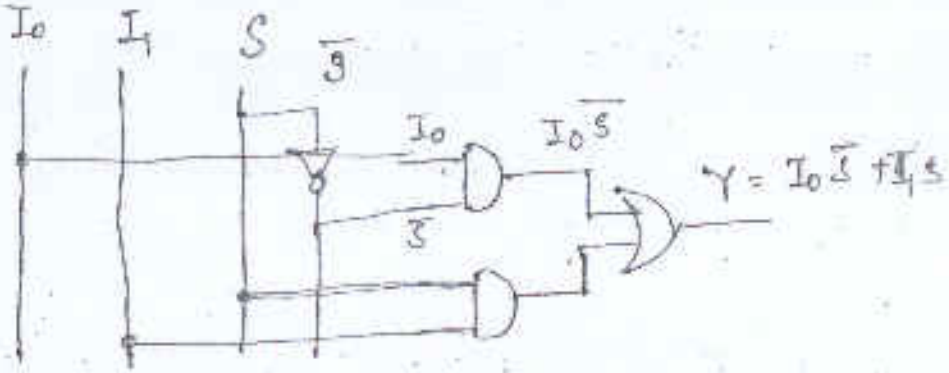
Truth Table:

i/p	S	Y
I_0	0	I_0 X
I_1	1	X I_1

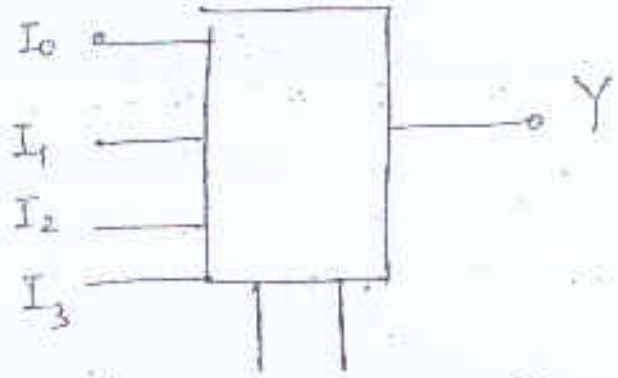
Expression

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

→ Logic Digram design :-



(ii) 4:1 MUX :-



i/p	S_1	S_0	Y
I_0	0	0	I_0
I_1	0	1	I_1
I_2	1	0	I_2
I_3	1	1	I_3

T.T

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

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

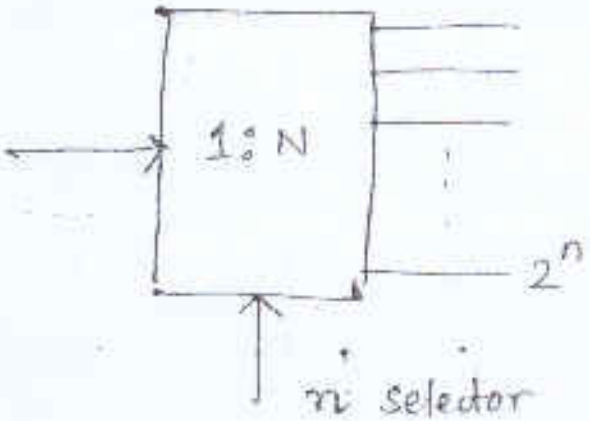
Logic Design :-

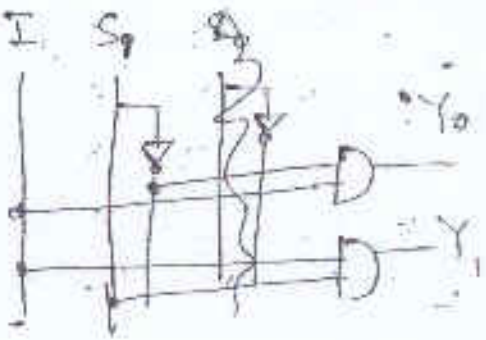
Demultiplexer (DEMUX):

(Data Distributors)

Defⁿ:- It takes a single i/p and distributes it over several outputs.

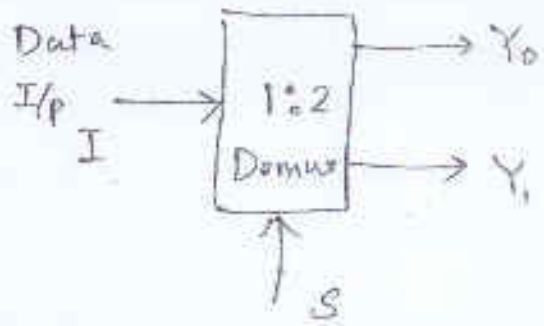
→ Demux is a $1:N(2^n)$ device.





1. 1:2 DEMUX
2. 1:4 Demux
3. 1:8 Demux

1:2 DEMUX :-



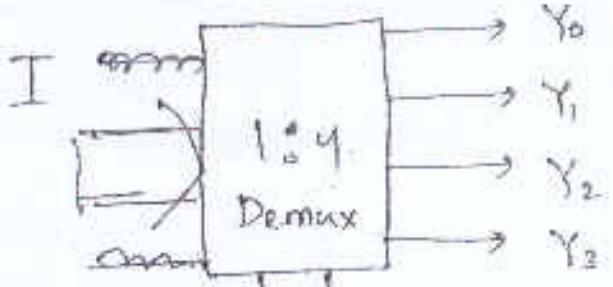
Truth Table

I	S	Y_0	Y_1
↓	0	I	X
	1	X	I

$Y_0 = I\bar{S}$

$Y_1 = IS$

1:4 Demux :-



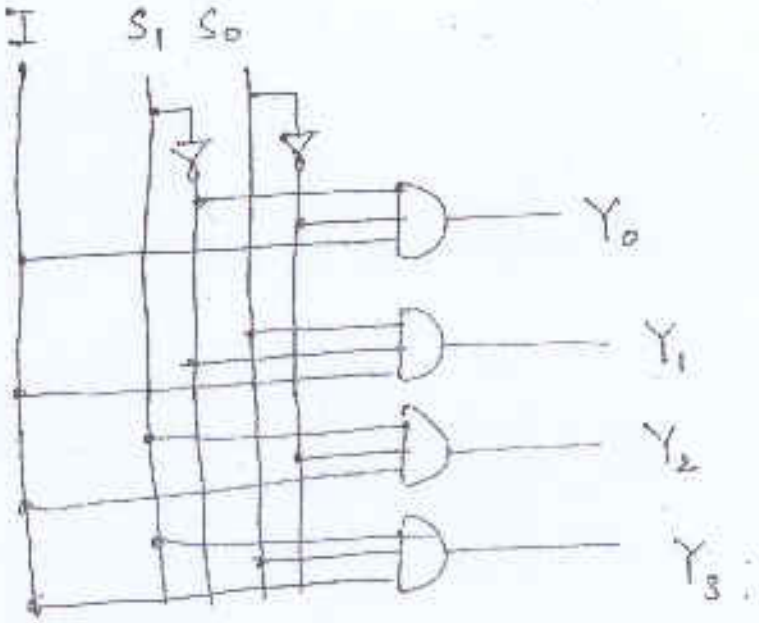
$Y_0 = \bar{S_1}\bar{S_0}I$

$Y_1 = \bar{S_1}S_0I$

$Y_2 = S_1\bar{S_0}I$

$Y_3 = S_1S_0I$

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



Logic Diagram.

Electrical Engineering Material

1. Conducting materials
2. Semiconducting materials
3. Insulating materials
4. Dielectric materials
5. Magnetic materials
6. Material for special purposes.

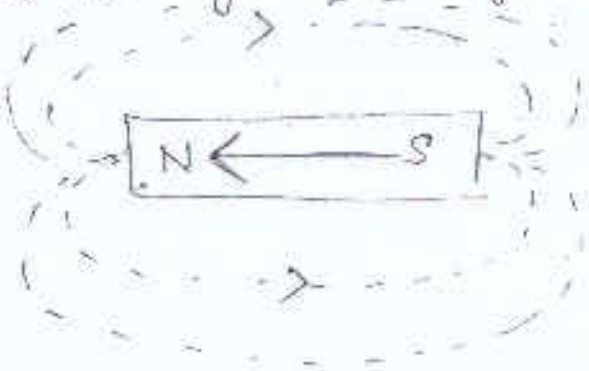
Magnetic Materials

1. Introduction
2. classification
3. Diamagnetism
4. Para magnetism
5. ferromagnetism
6. Magnetisation curve
7) Hystereris
8) Eddy currents
9) Curie Point
10) Magneto-striction.
11) Soft & Hard magnetic Materials
(a) Soft magnetic material
(b) Hard magnetic material

Magnets

① Permanent Magnet ② Electromagnet (current induced magnetism)

Magnetic dipoles :-

- Magnetic dipoles are found to exist in magnetic materials.
- A magnetic dipole is a small magnet composed of north & south poles instead of positive & negative charges.
- 

- Magnetic forces are generated by moving electrically charged particles. These forces are in addition to any electrostatic forces that may already exist
- → Magnetic forces ~~as~~ present in distributed field, which is represented by imaginary lines. These lines also indicates the direction of the force.

Magnetic field :-

- Magnetic field is generated by passing current 'I' through a coil of length 'l' and number of turns 'n', then the

magnetic field Strength, H (unit A/m)

$$H = \frac{nI}{\ell}$$ unit → Amp/meter

$$\delta H = \frac{1}{4\pi r^2} i\, \delta\ell \times \hat{r}$$

Magnetic flux Density :-

$$B = \frac{\phi}{A}$$ unit - wb/m^2 (or) Tesla.

$$B = \mu H$$

μ → permeability

↳ It is a specific property of the medium.

unit → $\frac{wb}{A \cdot m}$ (or) Henry/meter.

- Relative magnetic permeability,

$$\mu_r = \frac{\mu}{\mu_0}$$

$$\Rightarrow \mu = \mu_r \mu_0$$

μ_0 → magnetic permeability of Vacuum.

μ_r →

Magnetic Susceptibility :-

How much a material will become magnetized in an applied in an applied magnetic field

Susceptibility = $(\chi) = \frac{M}{H}$; $B = \mu_0 H + \mu_0 M$

$= \mu_0 \mu_R H$

$\chi_m = \mu_R - 1$.

Types of Magnetism :-

- A material is magnetically characterized based on the way it can be magnetized
- This depends on the material's magnetic susceptibility

There are 3 basic magnetisms,

1. Dia-magnetism
2. Para-magnetism
3. Ferro-magnetism

Dia-magnetism :-

- Very weak; exists only in presence of an external field, non-permanent
- The induced magnetic moment is very small and the magnetisation (M) direction is

1) Design of a 3-bit Binary to Gray code converter.

Design :-

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

$G_3 = B_2$.

$G_2 = B_3 \oplus B_2$

$G_1 = B_2 \oplus B_1$.

ray

Design of a 4-bit Gray-to-Binary code converter :-

→ The i/p is a 4-bit gray code & o/p is a 4-bit binary.

→ There are 16 possible combinations of 4-bit gray input and all of them are valid

Gray				Binary			
G_3	G_2	G_1		B_3	B_2	B_1	
0 ⊕ 0 ⊕ 0			→	0	0	0	
0 ⊕ 0	1		→	0	0	1	
0 ⊕ 1	0		→	0	1	1	
0 ⊕ 1	1		→	0	1	0	
1 ⊕ 0 ⊕ 0			→	1	1	1	
1 ⊕ 0 ⊕ 1			→	1	1	0	
1 ⊕ 1	0		→	1	0	0	
1	1	1	→	1	0	1	

4bit gray - Binary

1 ⊕ 0 ⊕ 0
↓ ↓ ↗ ↓
1 1 1

- $B_3 = G_3$
- $B_2 = G_3 \oplus G_2$
- $B_1 = G_3 \oplus G_2 \oplus G_1$
- $B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0$

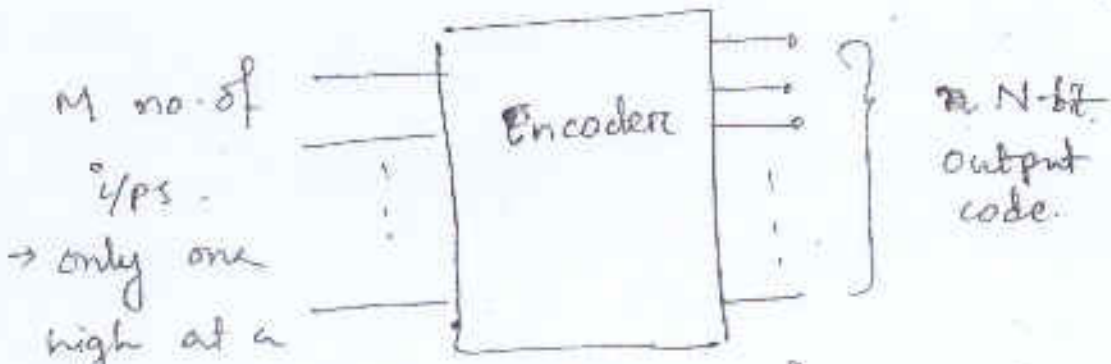
1) Design of 4 bit Binary to Gray.
2) Design of 4 bit Gray to Binary.
3) Design of 4bit binary to Ex-3.
4) Design of 4bit Ex-3 to binary.

Encoder:-

An encoder is a device whose inputs are decimal digits or alphabetic character and whose outputs are the coded representation of those inputs.

→ Combinational logic circuit that performs the 'reverse' operation of the decoder.

→ The opposite of decoding process is called encoding



Block diagram of Encoder.

Octal to Binary Encoder :-

→ This a 8:3 line encoder. It accepts 8 input lines and produces a 3 bit output code.

Truth Table

Octal Digits	Binary		
	A_2	A_1	A_0
D_0	0	0	0
D_1	0	0	1
D_2	0	1	0
D_3	0	1	1
D_4	1	0	0
D_5	1	0	1
D_6	1	1	0
D_7	1	1	1

Logic Diagram

$A_2 = D_4 + D_5 + D_6 + D_7$

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_0 = D_1 + D_3 + D_5 + D_7$

Decimal to BCD Encoder :-

(BCD → o/p)

Decimal i/ps	Binary A	B	C	D
D_0	0	0	0	0
D_1	0	0	0	1
D_2	0	0	1	0
D_3	0	0	1	1
D_4	0	1	0	0
D_5	0	1	0	1
D_6	0	1	1	0
D_7	0	1	1	1
D_8	1	0	0	0
D_9	1	0	0	1

Truth Table

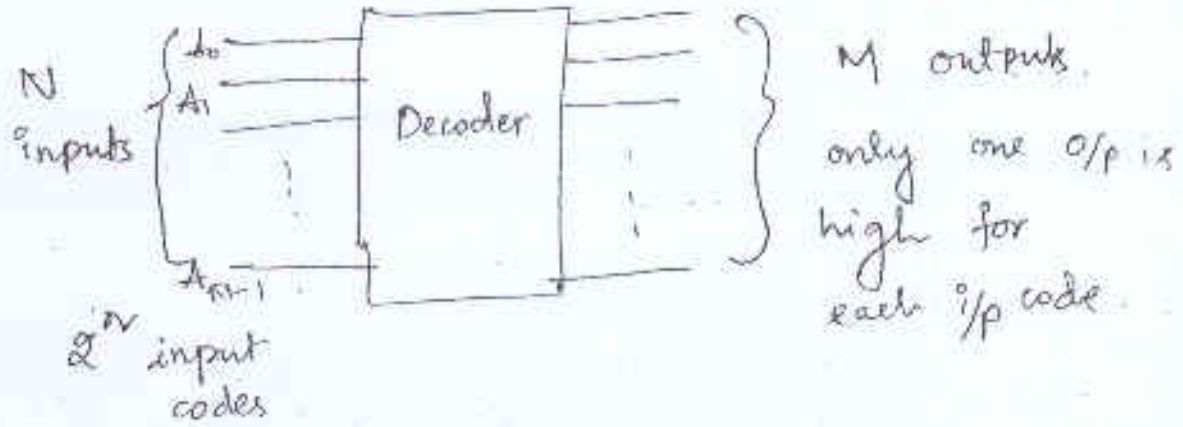
$A = D_8 + D_7$, $B = D_4 + D_5 + D_6 + D_7$

$C = D_2 + D_3 + D_6 + D_7$, $D = D_1 + D_3 + D_5 + D_7 + D_9$

→ There is no explicit input for a decimal 0. The BCD output is 0000 when the decimal inputs 1-9 are all '0'.

Decoder :-

A decoder is a digital circuit that converts an N-bit binary i/p code into M-o/p lines such that only one o/p line is activated for each one of the possible combination.



(1) 2:4 line decoder

(2) 3:8 line decoder

(1) 2:4

A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$D_0 = \bar{A}\bar{B}$, $D_1 = \bar{A}B$, $D_2 = A\bar{B}$, $D_3 = AB$.

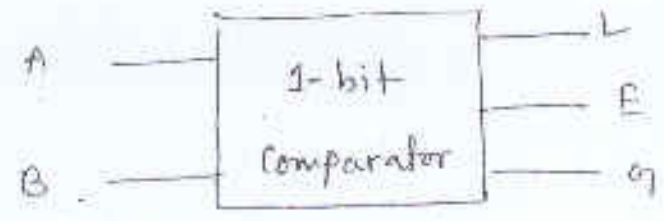
(3) 3:8 Decoder :-

i/ps			o/ps							
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
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								

Logic Design

Comparator :-

A comparator is a logic circuit used to compare the magnitude of two binary number.



A_0	B_0	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

2 bit magnitude Comparator :-

Let the two 2-bit numbers be $A = A_1 A_0$ and $B = B_1 B_0$

(1)

A		B				
A_1	A_0	B_1	B_0	L	E	G
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			

$A > B \longrightarrow G = A\overline{B}$

$A < B \longrightarrow L = \overline{A}B$

$A = B \longrightarrow E = A \odot B$

2-bit magnitude Comparator :-

Let the 2 bit numbers $A = A_1 A_0$ and $B = B_1 B_0$

1. $A_1 = 1$ and $B_1 = 0$, then $A > B$ or
2. If A_1 and B_1 coincide and $A_0 = 1$ and $B_0 = 0$, then $A > B$. So the logic expression for $A > B$ is $G = A_1\overline{B_1} + (A_1 \odot B_1) A_0 \overline{B_0}$

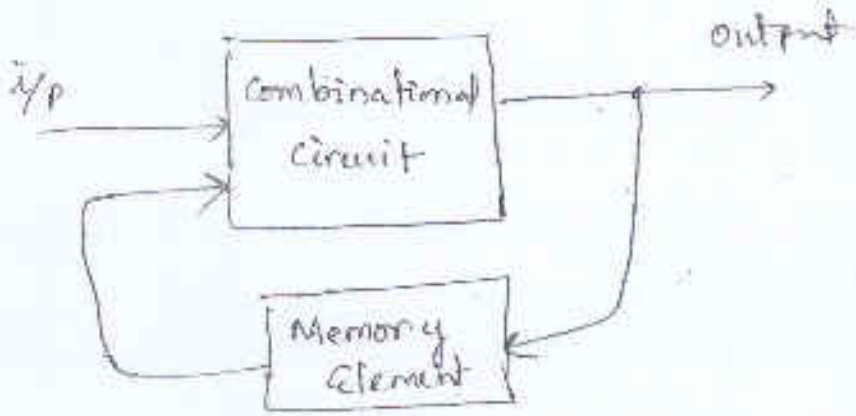
1. If $A_1 = 0$ and $B_1 = 1$, then $A < B$ or

2. If A_1 and B_1 are equal and $A_0 = 0$ and $B_0 = 1$, then $A < B$ is

$$A<B : L = \bar{A}_1 B_1 + (A_1 \odot B_1)\bar{A}_0 B_0$$

1. If A_1 and B_1 coincide and if A_0 & B_0 coincide then $A = B$. So the expression for $A = B$; $E = (A_1 \odot B_1)(A_0 \odot B_0)$

Sequential Circuit



Block diagram of sequential circuit

The output of the sequential circuit depends on not only the present i/p but also the past o/p & i/p of the system.

e.g :- Counter, shift registers

Combinational ckt	Sequential Circuit
(1) In combinational ckt, the output variables at any instant of are dependent only on the present i/p variables	(1) In sequential ckt, the o/p variables at any instant of time are dependent not only on the present i/p variables, but also on the past history of the system.
(2) memory unit is not required in combinational circuit.	(2) Memory unit is required to store the past history of the input variables in sequential circuit
(3) Combinational ckt are faster in response because the delay b/w the i/p & o/p is due to propagation delay of gates only.	(3) Sequential ckt are slower than combinational ckt
(4) combinational ckt are are easy to design.	(4) Sequential ckt are comparatively harder to design.

Classification of Sequential circuits :-

(1) Synchronous sequential circuit

(2) Asynchronous " "

Synchronous Seq. ckt :-

The seq. ckt which are controlled by a clock are called synchronous sequential ckt. These ckts will be active only when clock signal is present.

Asynchronous Sequential ckt :-

The seq. ckt which are not controlled by a clock are called asynchronous sequential ckt

(1) Latches
   (a) SR
   (b) D

(2) flip flops.
   (a) SR
   (b) JK
   (c) D
   (d) T.

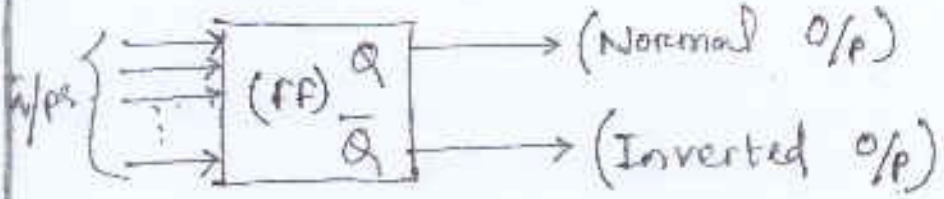
flip flop :-

→ flip-flop are the basic building block of the sequential circuit.

→ A flip flop i/p has to pulsed momentarily to cause a change in the flip-flop o/p. and the o/p will remain in that new state even after the i/p pulse has been removed. This is the flip flop's memory characteristics.

→ flip-flop acts as a storage device. It stores '1' when 'Q' o/p is 1; stores '0' when Q o/p is 0.

→ flip flops are the fundamental components of shift Registors & counters.



→ A flip flop can have one or more i/ps. The i/p signals which command the flip flop to change state are called excitation

→ A flip flop has two outputs, labelled Q & $\overline{Q}$.

Q → normal o/p.

$\overline{Q}$ → Inverted o/p.

→ The state of the flip-flop always refers to the state of the normal o/p (Q).

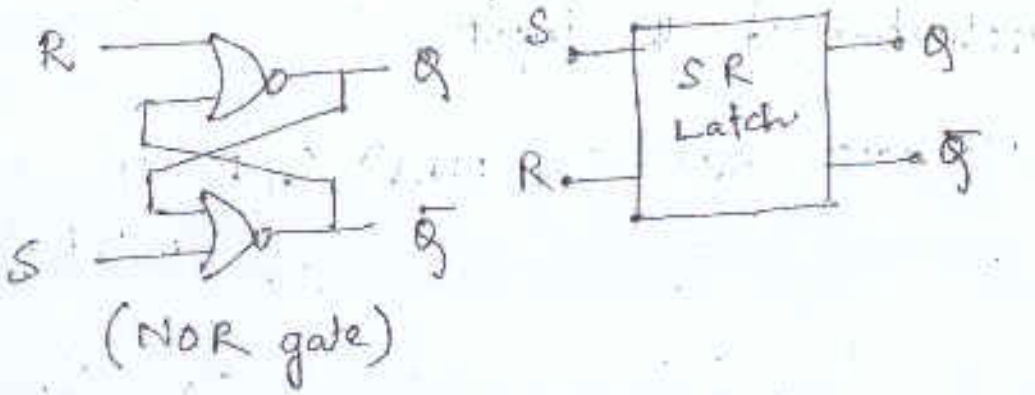
⇒ when Q = 1, The ff is said to be in "high" state or logic 1 state or SET state.

⇒ when Q = 0, The ff is said to be in LOW state or logic 0 state or RESET state or CLEAR state.

Latches :-

→ It has no clock pulse.

(1) SR Latch :-



→ It can be constructed using either two cross coupled NAND gates or NOR gates.

→ using NOR gates, an active high SR Latch can be constructed.

→ using two NAND gates an active LOW SR Latch can be constructed.

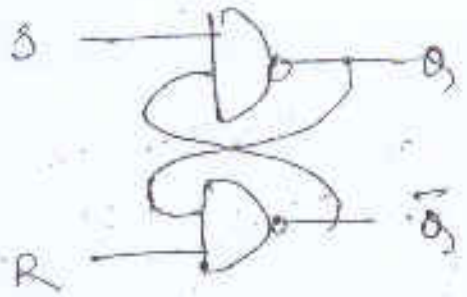
→ The name of the latch, SR (set-Reset).

S	R	Q	$\bar{Q}$
0	0	previous state. (no-change in state)	
0	1	0	1
1	0	1	0
1	1	Indetermined or Invalid.	

→ SR Latch is also called (SET-CLEAR) latch.

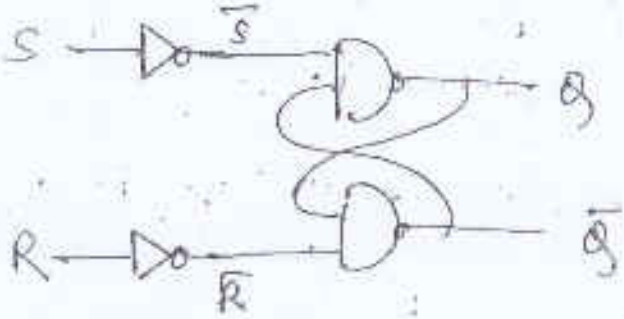
→ In more complex flip-flops, called gated Latches, the changes of state does not take place immediately after the application of the inputs.

The NAND gate (S-R Latch) (Active: low S-R Latch):-



S	R	Q_n	$\bar{Q}_n$
0	0	Indetermined	
0	1	Set	
1	0	Reset	
1	1	No change	

$\bar{S}$-$\bar{R}$ Latch (active high NAND Latch):-



Gated Latches (Clocked flip flops):-

asynchronous latches:- The o/p can change state any time the i/p conditions are changed.

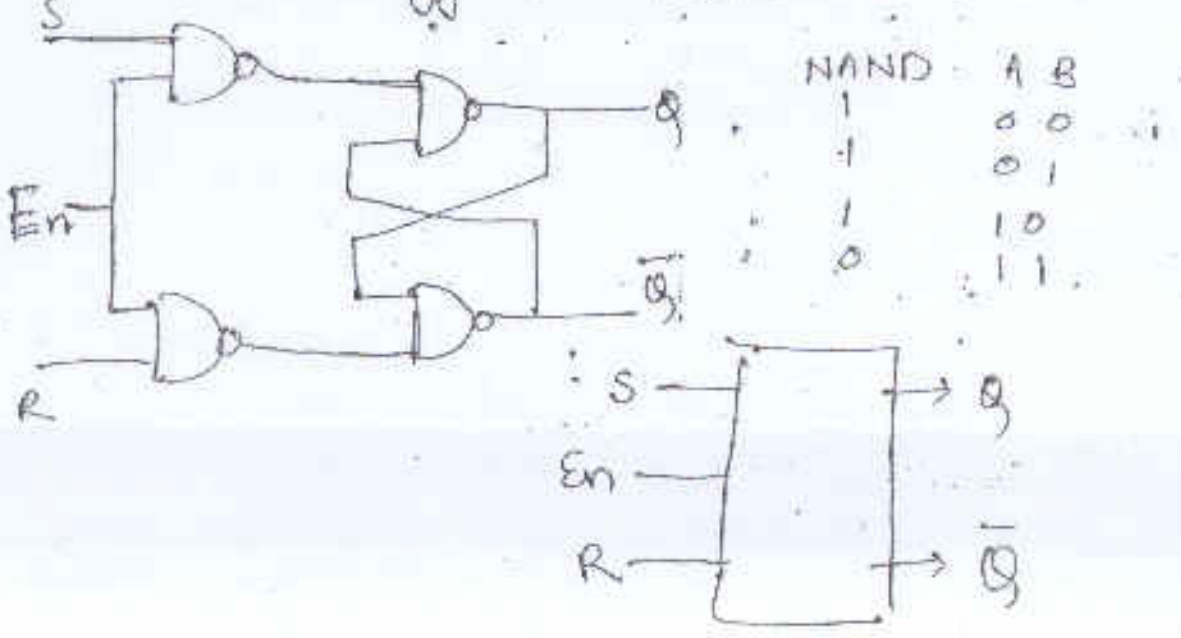
→ A gated S-R latch requires an Enable (EN) i/p. It's S & R i/ps will control the state of the flip flop only when the enable is High.

→ When, the enable is LOW, the i/ps become ineffective and no changes of state can take place.

→ The enable i/p may be clock.

So a gated S-R latch is also called a clocked S-R latch or Synchronous S-R Latch.

→ This types of flip-flops are respond to the changes in inputs only as long as the clock is HIGH. These types of flip-flops are also called level triggered flip-flops.

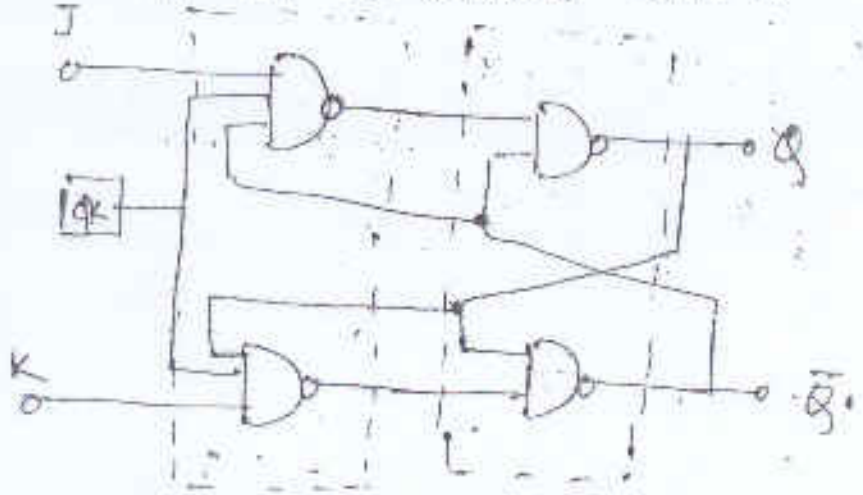


Truth Table :-

En	S	R	Qn	Qn+1	State
1	0	0	0	0	No change
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	x	Invalid
1	1	1	1	x	
0	x	x	0	0	No change
0	x	x	1	1	

JK - flip flop :-

→ JK flip flop is called as universal flip flop because the flip flop like T-ff, D-ff, SR-flip flop can be derived from it



Truth Table

clock	J	K	Q_{n+1}
0	x	x	Q_n.
1	0	0	Q_n → Hold.
1	0	1	→ 0 → Resent
1	1	0	→ 1 → Set
1	1	1	→ $\bar{Q}_n$ → Toggle State } → Race Around Condition

characterstics table

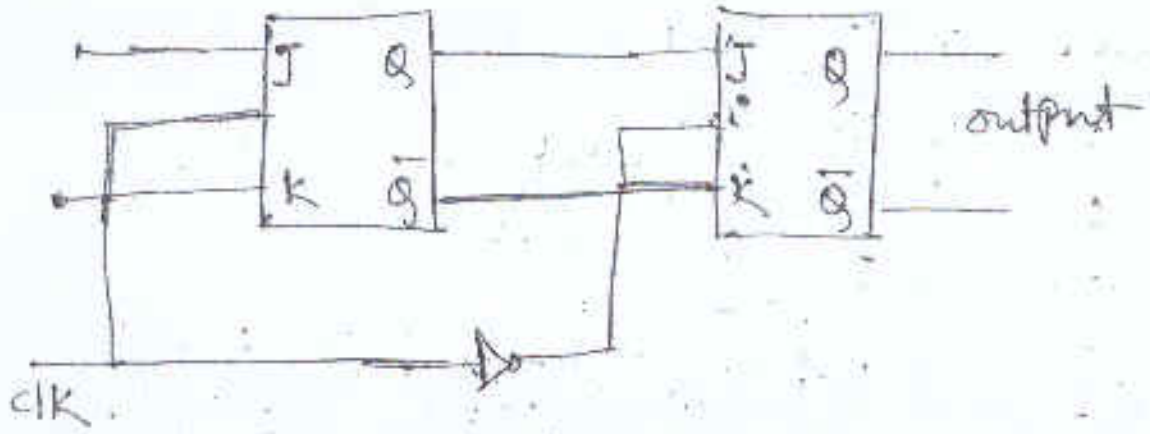
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Lip flop

S R-flop

$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$ → characterstics Equation.

Master-Slave JK Flip-flop :-



State the necessity of clock and give the concept of level clocking and edge triggering.

→ Clock helps to synchronise the ~~input and output~~ Sequential circuit to get a single external signal.

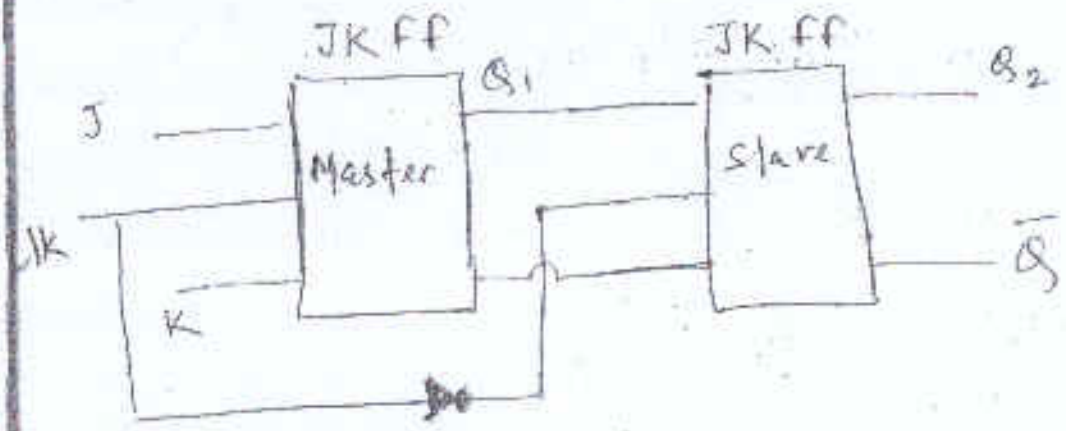
→ It prevents undesired echos in the output.

master - slave flip flops :-

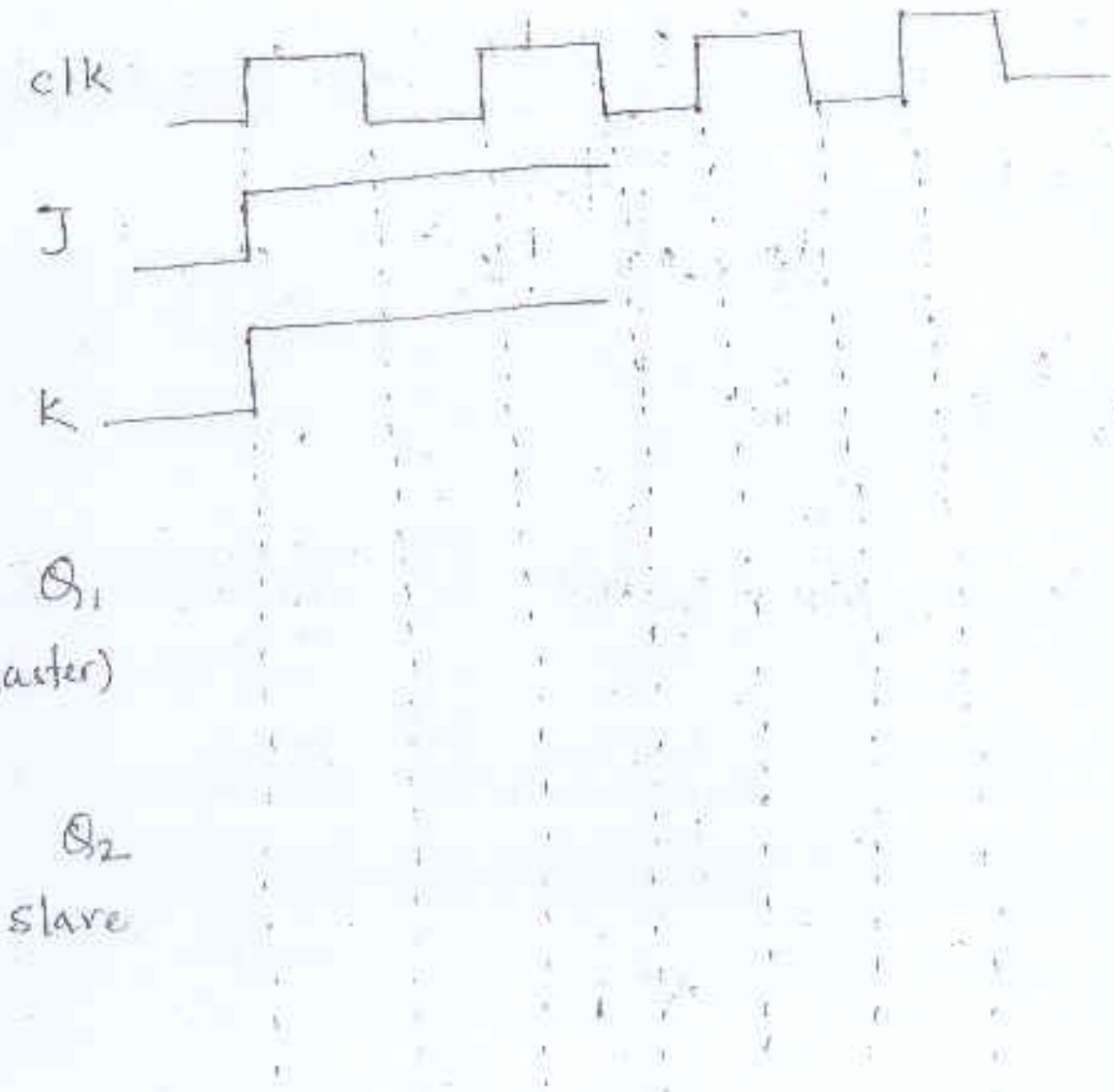
→ master-slave (or) pulse triggered flip flop contains two flip flops

→ Racing is a uncontrolled phenomenon

Toggling is a controlled phenomenon



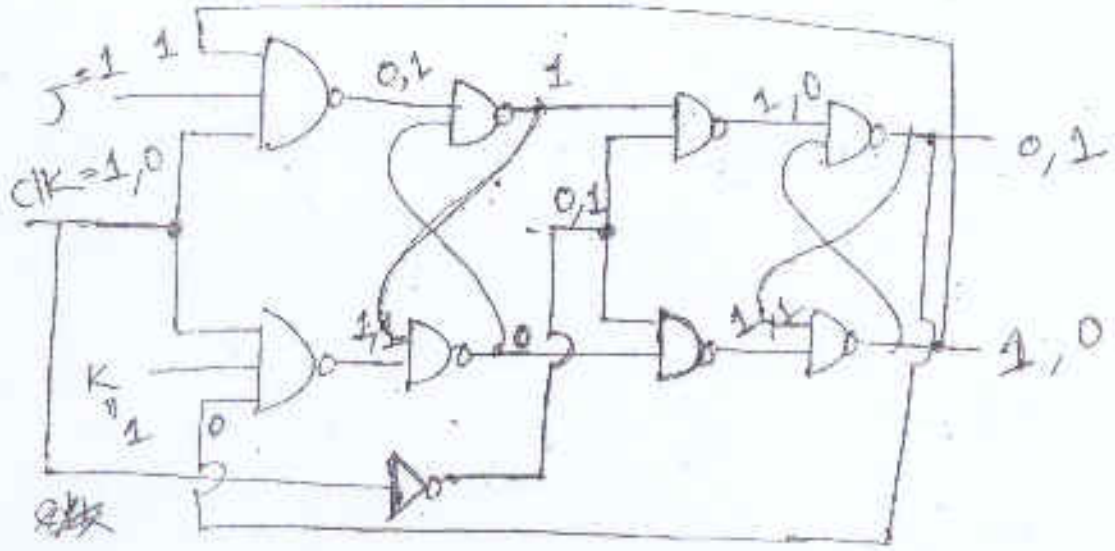
Timing diagram :-



(*) Master slave flip flop is same as -Ve Edge triggered flip flop.

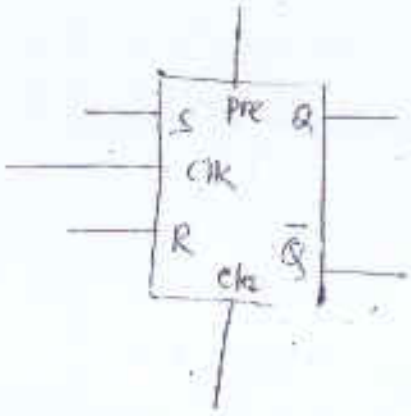
→ Racing is due to feed back.

→ We add another level.



→ master slave ff is same as -Ve edge triggered ff.

Clocked SR - flip flop with preset & clear inputs :-



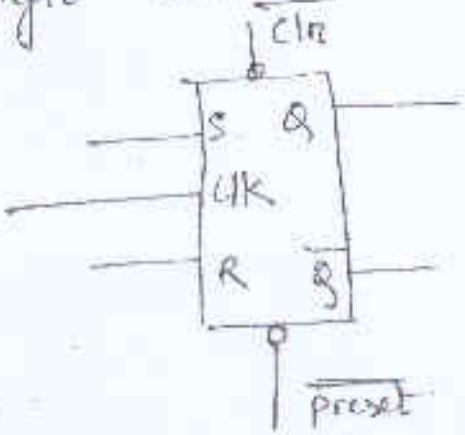
present is used to set the output Q i.e to 1.

clear is used to clear the output(Q) to '0'.

→

Clear	preset	Q
0	0	no change
0	1	1
1	0	0
1	1	no effect

If the clear & preset is activated on -Ve logic then the block diagram will be,



Application of flip flop :-

1. used for data storage
2. transfer of data
3. frequency Division.
4. counting
5. Parallel to Series & Series to Parallel Conversion

Parallel Data Storage :-

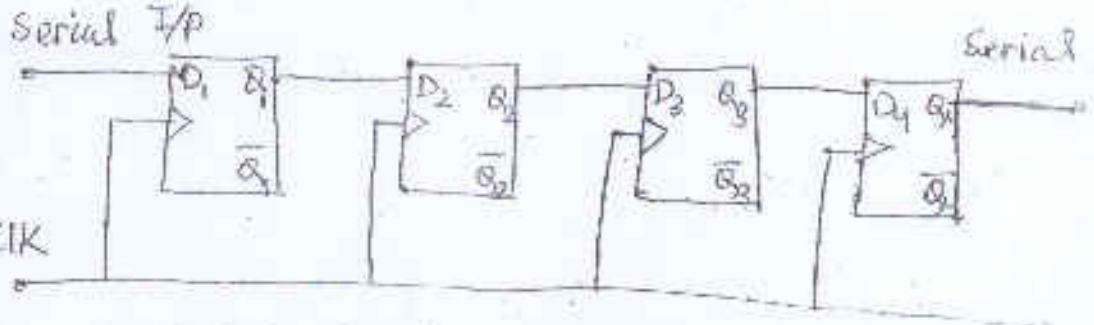
→ A group of flip flop is called a register. To store a data of N-bit, N flip flops are required.

→ Since the data is available in parallel form i.e all bits are present at a time.

Shift Register :-

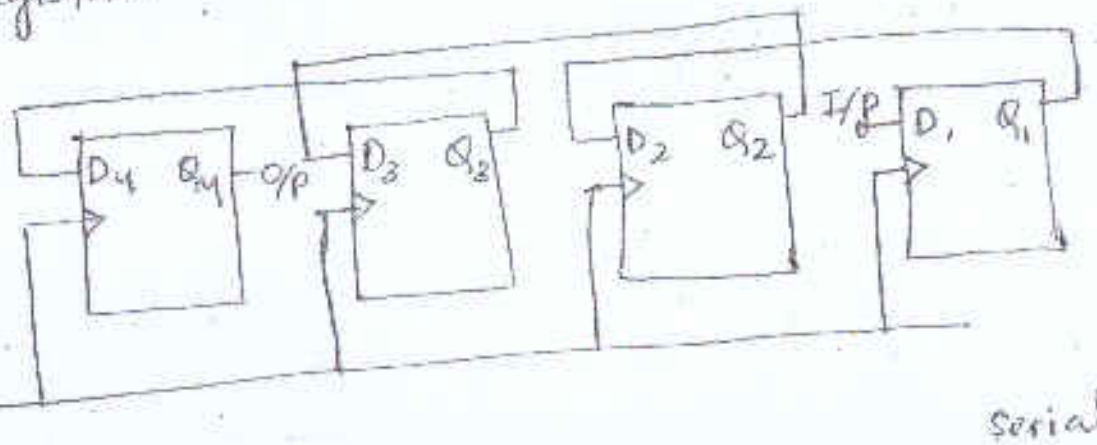
(1) SISO
(2) SIPO
(3) PISO
(4) PIPO

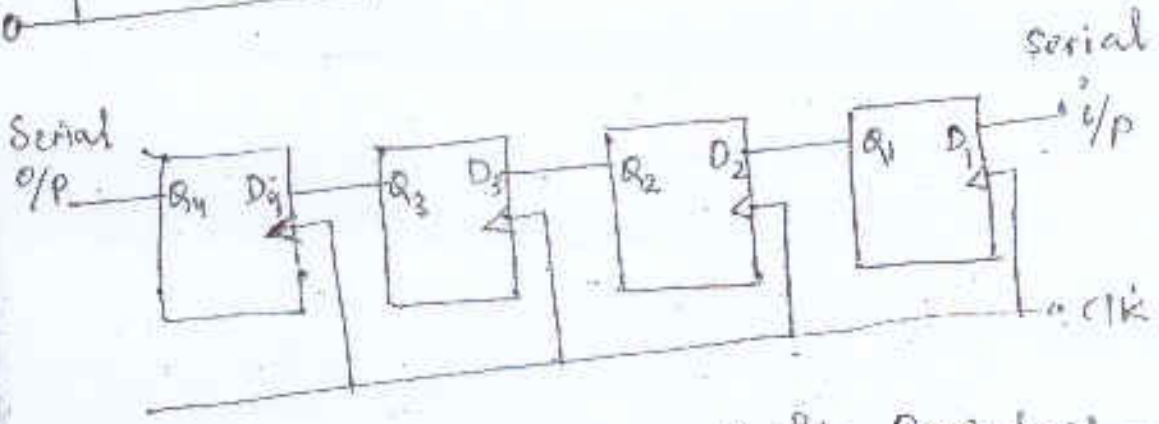
SISO :- (Serial in Serial out)



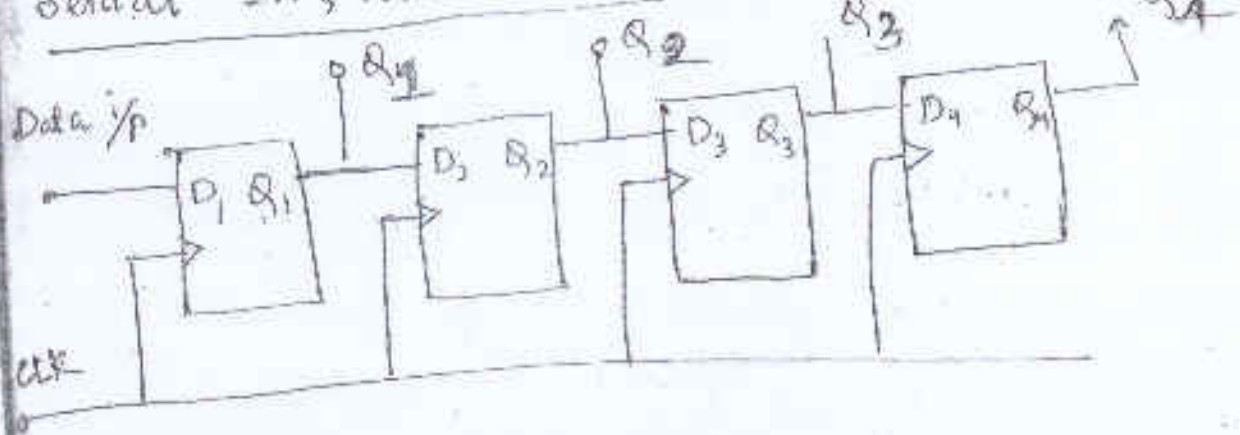
4-bit Serial-in, serial out, shift right Shift register.

4-bit serial in, serial-out, shift-left, shift register.



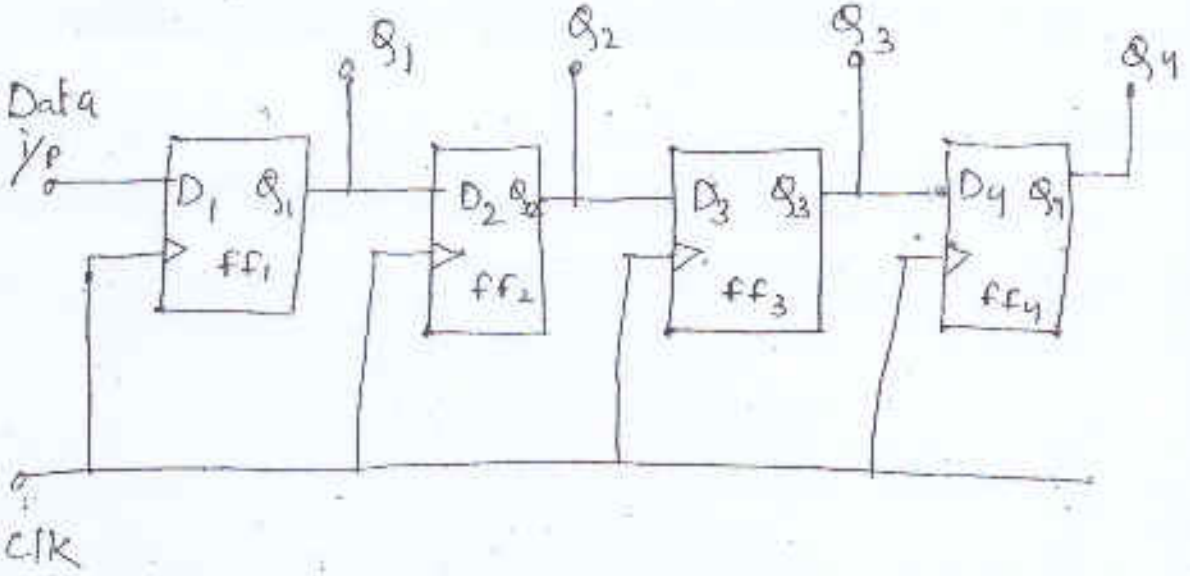


Serial-In, Parallel-Out, Shift Register:-



SIPO :-

In this type of register, the data bits are entered into a register serially, but the data stored in the register is shifted out in parallel form.



4-bit Serial in, Parallel-out

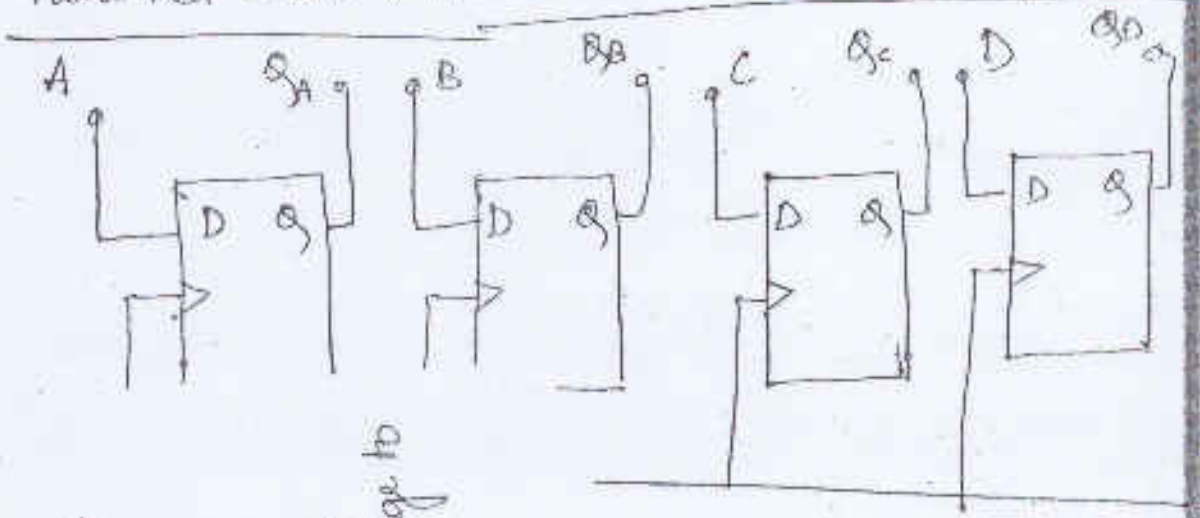
Parallel In, Serial-out (PISO) Shift register :-

In a parallel in, serial-out shift register the data bits are entered simultaneously into their respective stages on parallel lines, but the data bits are transferred out of the register serially, i.e bit by bit basis over a single line.

When Shift/$\overline{LOAD}$ = 0, gates G_2, G_4 are enabled allowing the data input to appear at the D inputs of the respective ffs., therefore data is inputted in one step.

→ The OR gates allows either the normal shifting operation or the parallel data entry depending on which AND gates are enabled by the level on the shift/$\overline{LOAD}$ input

Parallel-in, Parallel-out shift Register:



4-bit Parallel in - Parallel

the data
their respectiv
but the dat
of the register

Counters

1. Asynchronous counter / Ripple Counter.
2. Synchronous Counter.

Asynchronous Counters	Synchronous Counter
1. In this type of counter FFs are connected in such a way that the output of 1st ff drives the clock for the 2nd ff, the o/p of the 2nd the clock of the third & so on.	1. In this type of counter there is no connection b/w the o/p of the 1st FF and clock input of next ff & so on.
2. All the ffs are not clocked simultaneously.	2. All the ffs are clocked simultaneously.
3. Design & implementation is very simple even for more number of states.	3. Design & implementation becomes tedious and complex as the number of states increases.
4. Main drawback of these counter is their low speed as the clock is propagated through a number of ffs before it reaches the last FF.	4. Since clock is applied to all the ffs simultaneously the total propagation delay is equal to the propagation delay of only one ff. Hence they are faster.

Counter → (i) UP Counter
→ (ii) DOWN Counter

(i) UP counter :-

An up-counter is a counter which counts in the upward direction i.e

0, 1, 2, 3 ---- N.

(ii) DOWN Counter :-

A down-counter is a counter which counts in the downward direction, i.e

N, N-1, N-2, N-3, ---- , 1, 0

State :-

Each of the counts of the counter is called the state of the counter.

Modulus of the counter :-

The number of states through which the counter passes before returning to the starting stage is called the modulus of the counter.

→ So the modulus of the counter is equal to the total number of distinct states (count

(i) 2-bit counter

(ii) 3-bit counter

(iii) 4-bit counter.

(iv) 5-bit counter.

(i) 2-bit counter :-

→ It has 4-states, it is called mod-4 counter.

→ It requires 2 ffs.

→ The number of states = $2^2 = 4$ states

→ It divides the i/p clock signal frequency by 4, therefore, it is also called a divided by 4-counter.

(ii) 3-bit counter :-

→ A 3-bit counter uses 3 ffs and has $2^3 = 8$ states.

→ The number of states = 8.

→ It divides the i/p clock frequency by 2^3 i.e 8.

Note :- In general, an n-bit counter will have a n ffs and 2^n states, & divides the i/p frequency by 2^n. Hence it is a divide by-2^n counter.

→ e.g :- MOD-2 counter

How many ~~bit's~~ ffs it require?

MOD-4 counter, mod-5, mod-10

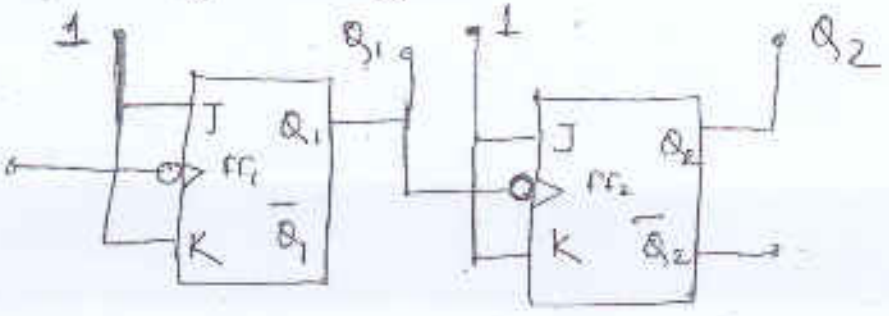
How many ffs it require?

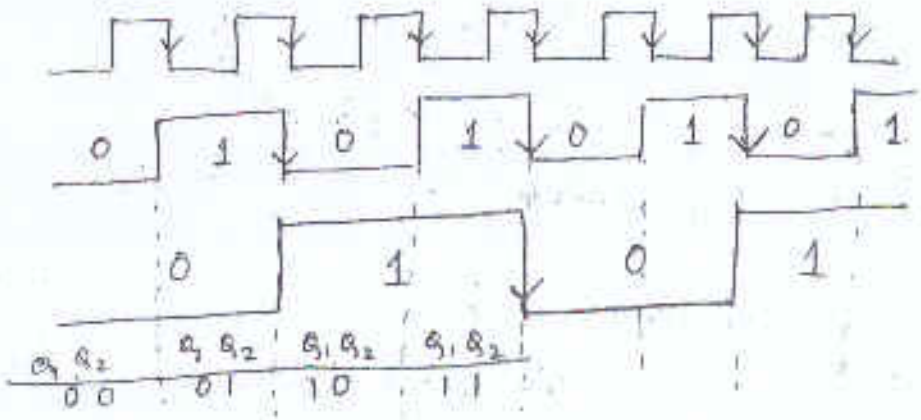
→ mod-2 counter & mod-5 counter can be combined to get a mod-10 counter.

→ mod-5 counter & a mod-4 counter can be combined to get a mod-20 counter.

Asynchronous Counter :-

(1) Two bit Ripple UP-COUNTER using -Ve Edge-triggered Flip-flops.



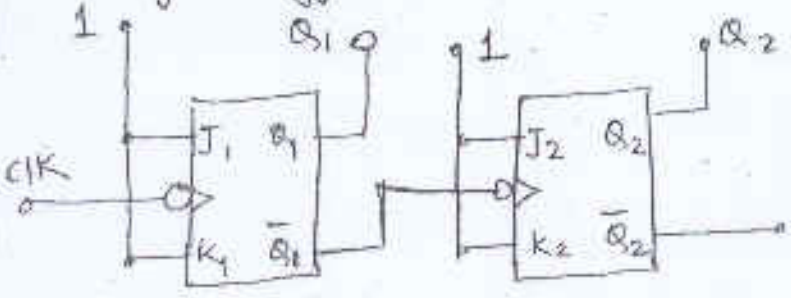


(ii) 2-bit Ripple Down-Counter using -Ve Edge triggered flip-flops :-

→ 2-bit down counter counts in the order.

0, 3, 2, 1, 0, 3, 2, 1, 0 i.e 00, 11, 10, 01, 00, 11 .... so on.

→ 2-bit ripple down-counter using negative edge triggered J-K ffs.

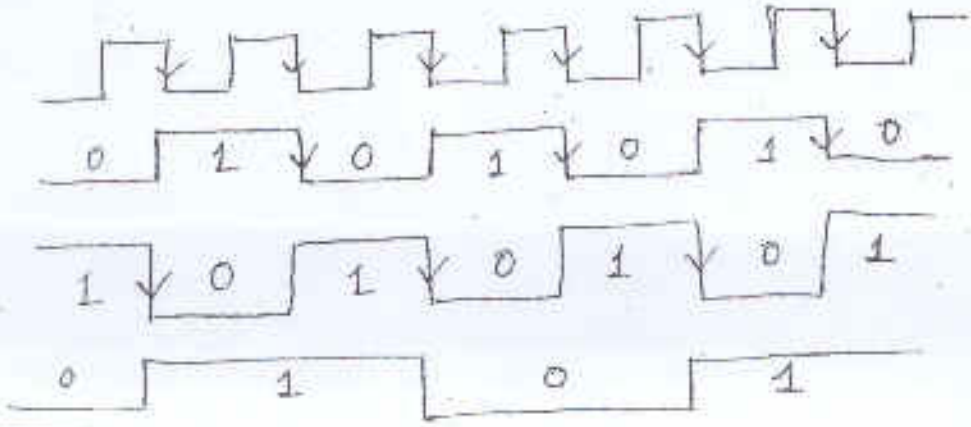


Logic diagram

Q_2Q_1 $\bar{Q}_2\bar{Q}_1$

00, 11, 10, 01, 00

Timing Diagram



Combinational logic Circuits

→ A digital circuit is combinational if it's output is depending on inputs (i.e the present i/p).

→ The combinational logic circuit is memory less.

→ This logic circuit deals with the method of combining basic gates to get the desired solution.

→ Elements of Combinational logic :-

(1) Literal
(2) Product term
(3) Sum term
(4) Sum of products
(5) Products of sum
(6) Minterms
(7) Maxterms
(8) Canonical forms,
(9) Canonical sum of products
(10) Canonical Product of sums
(11) Sum of minterms.
(12) Product of maxterms.

Combinational logic Circuits :-

2.1) Give the concept of combinational logic circuits

2.2) Half adder circuit and verify its function using truth table

2.3) Realize a Half-adder using NAND gates only and NOR gates only

2.4) full adder circuit and explain its operation with T.T

2.5) Realize full adder using two half adder and an OR-gate & write T.T.

2.6)
2.7) Operation of 4:1 mux & 1:4 demux.
2.8) Working of Binary - Decimal Encoder & 3x8 decoder.
2.9) Working of two bit magnitude comparator.

Chapter-1 :-

1) K-map for 2,3,4 variable, simplification of SOP and POS Logic Expression using K-MAP.

Digital Electronics

Q.1) Name 4 types of number system and write their respective bases?

2) 3) 4) 5) Find the 2's complement of $(10101101)_2$

3) Design Ex-OR gate using NAND gate

4) Write which gates are the basic gates & write it's truth Table with symbol

5) Define demorgan's theorem & write it's expression in 2 & 3-variable form

Scan

1) Design ~~Half~~ Adder with neat logic diagram.

2) Design 4:1 multiplexer with neat logic diagram

3) Reduce. ~~Solve~~ the given expression by using K-map and also draw it's logic diagram.

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