Unit - 3

Digital Electronics

Number System Representation :-

→ Decimal system :-

* This system is very simple & easy for us to perform any mathematical operation as the name Deci means ten & it has ten different levels or symbols which are 0, 1, 2, ... 9. Any number greater than '9' is represented by combination of these ten numbers.

Example :- $(1240)_{10}$

* The number '10' outside the bracket indicates the base or radix which represent the number given within the bracket is in decimal form or decimal number.

* The number within the bracket, each number of symbol has positional value as well as absolute value.

ie,

3	2	1	0
1	2	4	0

$= 1\times10^3 + 2\times10^2 + 4\times10^1 + 0\times10^0$

Generally we can represent,

$$[A_n, A_{n-1} \ldots A_0, A_{-1}, A_{-2}, \ldots A_{-m}]_r$$

Where $r \rightarrow$ radix or base of the number

$A_n \rightarrow$ Most significant bit [MSB]

$A_m \rightarrow$ Least significant bit [LSB]

The corresponding value will be,

$$A_n r^n + A_{n-1} r^{n-1} + A_{n-2} r^{n-2} \ldots A_0 r^0 + A_{-1} r^{-1} + A_{-2} r^{-2} + \ldots + A_{-r}$$

→ Binary Number System :-

* The binary number system is equivalent to electrical switch since switch has only two states either close or open similarly binary system has only two values either '1' (logic 1) or '0' (logic 0).

* Hence binary no system has a radix of '2'.

* Generally it is represented as,

$b_n, b_{n-1}, b_{n-2} \ldots b_0, b_{-1}, b_{-2}, b_3 \ldots b_{-m}$

→ This binary no can be converted into decimal equivalent by,

$$b_n 2^n + b_{n-1} 2^{n-1} + b_{n-2} 2^{n-2} + \dots b_0 2^0 + b_{-1} 2^{-1} + b_{-2} 2^{-2} + \dots b_m 2^{-m}$$

Eg :- $(1101.0101)_2$

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

1 1 0 1 . 0 1 0 1

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

$$= 1\times8 + 1\times4 + 1 + 0 + 1\times\frac{1}{4} + 1\times\frac{1}{16}$$

$$= 13 + 0.25 + 0.0625$$

$$= (13.3125)_{10}$$

Eg ②:- $(11001.110)_2$ Convert it into decimal form

$$1\times2^4 + 1\times2^3 + 1 + 1\times\frac{1}{2} + 1\times\frac{1}{4} + 0$$

$$= 16 + 8 + 1 + 0.5 + 0.25$$

$$= (25.75)_{10}$$

* Octal Number System :-

→ The octal no system has total of eight possible

or symbols from 0, 1, ... 7.

→ Since it has eight states hence it is represented by base or radix '8'.

→ Generally the octal numbers are represented as,

$$O_n, O_{n-1}, O_{n-2} \ldots O_0, O_{-1}, O_{-2} \ldots O_{-m}$$

→ The octal no can be converted into decimal equivalent by,

$$O_n 8^n + O_{n-1} 8^{n-1} + O_{n-2} 8^{n-2} + \ldots O_0 8^0 + O_{-1} 8^{-1} + O_{-2} 8^{-2} + \ldots O_{-m} 8^{-m}$$

Eg :- Convert $(2026)_8$ to decimal

$8^3\ 8^2\ 8^1\ 8^0$

2 0 2 6

$= 2 \times 8^3 + 0 + 2 \times 8^1 + 6 \times 8^0$

$= 1024 + 0 + 16 + 6$

$= (1046)_{10}$

Hexadecimal Number System :-

→ This is an important number system & it is widely used in Microprocessors, µcontrollers & Computers.

As the name indicates it is Hex + decimal (6+10), hence radix or base is '16' for this number system.

→ This number system has '16' possible values of symbols which are 0, 1, 2 .... 9 & A to F (ie, 10 to 15 is represented by English capital letters A to F).

→ Generally it is represented as,

$$H_n, H_{n-1}, H_{n-2}, \ldots H_0, H_{-1}, H_{-2} \ldots H_{-m}$$

→ The Hexadecimal number is converted into decimal as,

$$H_n 16^n + H_{n-1} 16^{n-1} + H_{n-2} 16^{n-2} + \ldots + H_0 16^0 + H_{-1} 16^{-1} + H_{-2} 16^{-2} + \ldots + H_{-m} 16^{-1}$$

Eg:- Convert $(1ABCD)_{16}$ to decimal

$$\begin{matrix} 4 & 3 & 2 & 1 & 0 \\ 1 & A & B & C & D \end{matrix}$$

$$= 1 \times 16^4 + A \times 16^3 + B \times 16^2 + C \times 16^1 + D \times 16^0$$

$$= (109517)_{10}$$

A => 10 C => 12
B => 11 D => 13

Decimal Number (10)	Binary Number (2)	octal Number (8)	Hexadecimal Number (16)
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Binary Coded Decimal (BCD):-

→ It is a numeric code in which each digit of decimal no is represented as a seperate group of binary bits.

→ It is also called as weighted binary code 8-4-2-1 in which each BCD no is represented by 4-bit binary no.

Decimal no	3rd bit 8	2nd bit 4	1st bit 2	0th bit 1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	1	0	0	0
5	0	0	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Eg:- ① $(87)_{10}$ = (0101 1000 0111) BCD

② $(6)_{10}$ = $(0110)_{BCD}$

③ (60) = $(0110\ 0000)_{BCD}$

* Number Base Conversion :-

→ Binary to Decimal Conversion :-

Procedure :-

1. Count the no of binary digits before fractional, binary or radix point & put the weight of each digit as 0, 1, 2 from right hand side to left hand side.

Let 'N' is the number of bits or digits, the last digit has weight of N-1.

2. Count the no of binary digits after the ~~xxx~~ radix or binary point & put the weight of each digit as -1, -2, -3 from left hand side to right hand side.

Let 'M' is the no of bits, last bit or digit has weight equal to 2^{-M}.

3. Write the binary bit stream is generalized Eqn given earli

4. Finally add all the products, which gives the number equivalent to the decimal number system.

amples:-

1. Convert $(1010110.1101)_2$ into decimal system.

$$\overset{6\ 5\ 4\ 3\ 2\ 1\ 0\ -1\ -2\ -3\ -4}{1\ 0\ 1\ 0\ 1\ 1\ 0\ .\ 1\ 1\ 0\ 1}$$

$$= 1\times2^6 + 0 + 1\times2^4 + 0 + 1\times2^2 + 1\times2^1 + 0 + 1\times2^{-1} + 1\times2^{-2} + 0 + 1\times2^{-4}$$

$$= 64 + 16 + 4 + 2 + \frac{1}{2} + \frac{1}{4} + \frac{1}{16}$$

$$= 86 + 0.5 + 0.25 + 0.0625$$

$$= (86.8125)_{10}$$

2. $(\overset{5\ 4\ 3\ 2\ 1\ 0}{1\ 0\ 1\ 0\ 1\ 0})_2 = (?)_{10}$

$$= 1\times2^5 + 0 + 1\times2^3 + 0 + 1\times2^1 + 0$$

$$= 32 + 8 + 2$$

$$= (42)_{10}$$

3. $(\overset{0\ -1\ -2\ -3}{0.1\ 1\ 0})_2 = (?)_{10}$

$$= 0\times2^0 + 1\times\frac{1}{2} + 1\times\frac{1}{4} + 0$$

$$= 0 + 0.5 + 0.25$$

$$= (0.75)_{10}$$

* Decimal to Binary System :-

1. Consider the given decimal number before decimal point & divide successively by '2' until the no is not divisible by '2' & at the same time write remainder for every division at the RHS side.

2. Write all the remainders from bottom to top gives the binary equivalent no of that given decimal no before decimal point.

3. For the conversion of fractional part of decimal no multiply it by 2 & write the carry or if there is any digit goes beyond fractional pt take out the carry & multiply it untill it reaches zero or perform this step four to five times.

4. Write all the carries generated from number of multiplication procedure from top to bottom to get the binary equivalent of decimal fractional part.

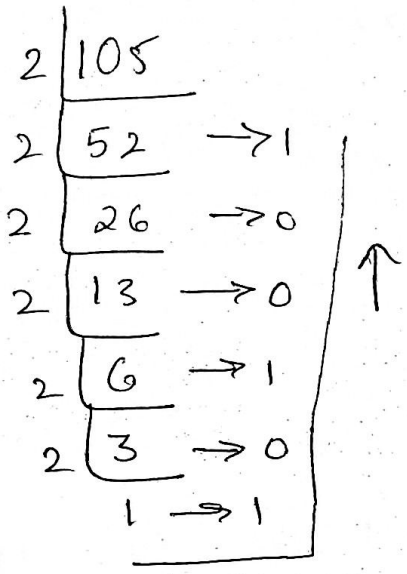
mal point

mples :-

1. $(105.202)_{10} = (?)_2$

105 is before fractional point ($\div$ by 2)

202 is after fractional point ($\times$ by 2)



$(105)_{10} = (1101001)_2$

0.202×2 Carry

$0.404 \rightarrow 0$

$\times 2$

$0.808 \rightarrow 0$

$\times 2$

$1.616 \rightarrow 1$

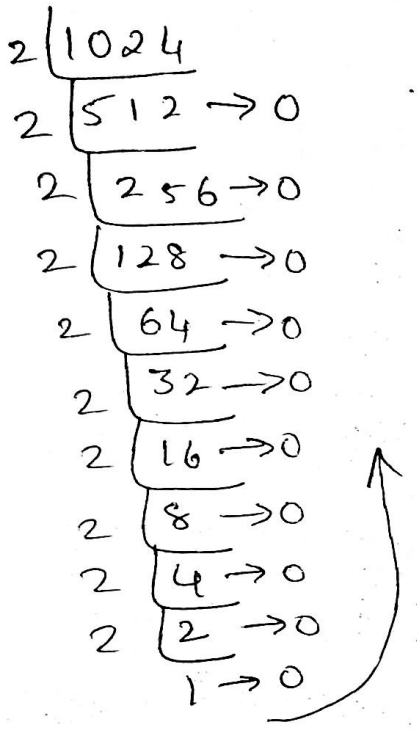
0.616×2

$0.232 \rightarrow 1$

$0.202 = (.0011)_2$

$\therefore (105.202)_{10} = (1101001.0011)_2$

② $(1024.625)_{10} = (?)_2$



0.625 × 2

0.250 → 1

× 2

0.500 → 0

× 2

0.000 → 1

1.00

$(1000000000.101)_2$

③ $(555)_{10} = (?)_2$

$(1000101011.1101)_2$

$(010110101001.10101100)_2 = (?)_8$

010 110 101 001 . 101 011 000 ← post fixed '0'

2 6 5 1 . 5 3 0

$= (2651.530)_8$

* Octal to Binary System :-

$(3565)_8 = (?)_2$

3 5 6 5

011 101 110 101

$(3565)_8 = (011101110101)_2$

(2) $(2651.53)_8 = (?)_2$

010 110 101 001 . 101 011

$(010110101001.101110)_2$

* Binary to Octal :-

The octal no system has the base of '8' & the bi

no has the base of '2'.

The third power of base of binary gives the octal base i.e, $2^3 = 8$, ∴ the power '3' indicates the no of binary digits required to represent octal no's.

Example :- ① $(111011101)_2 = (?)_8$

← Grouping direction

Prefixed 0 → 0 111 011 101

= 011 101 110 101

= 3 5 6 5

= $(3565)_8$

② $(.101011)_2 = (?)_8$

.101 011

5 3

$(0.101011)_2 = (0.53)_8$

* Binary to Hexadecimal :-

$2^4 = 16$

Eg:- $(1011010101001.101011)_2 = (?)_{16}$

'0' prefixed — 00 Post fixed

Prefixed

0101 1010 1001 . 1010 1100 — Post fixed

5 A 9 . A C

$(5A9.AC)_{16}$

(2) $(0.10000100)_2 = (?)_{16}$

$(0.84)_{16}$

* Hexadecimal to Binary Conversion :-

Eg:- $(ABC)_{16} = (?)_2$

A B C

1010 1011 1100

$(101010111100)_2$

② $(90AB.1C)_{16} = (?)_2$

1001 0000 1010 1011 . 0001 1100

$(1001000010101011.00011100)_2$

③ $(.CDF)_{16} = (?)_2$

$(0.1100\ 1101\ 1111)_2$

* Decimal to octal Conversion :-

① $(576)_{10} = (?)_8$

~~$(608^2 \cdots)$~~

$(1100)_8$

8 | 576
8 | 72 → 0
8 | 9 → 0
1 → 1

② $(985.85)_{10} = (?)_8$

$(1731)_8$

8 | 985
8 | 123 → 1
8 | 15 → 3
1 → 7

$0.85 \times 8 = 6.80 = 6$
$0.80 \times 8 = 6.40 = 6$
$0.40 \times 8 = 3.20 = 3$
$0.20 \times 8 = 1.60 = 1$
$0.60 \times 8 = 4.80 = 4$

$(985.85)_{10} = (1731.66314)_8$

* Decimal to Hexadecimal Conversion :-

① $(988.86)_{10} = (?)_{16}$

$0.86 \times 16 = 13.76 = (13)_{10} = (D)$

$0.76 \times 16 = 12.16 = (12)_{10} = (C)$

$0.16 \times 16 = 2.56 = (2)_{10} = (2)_{16}$

$0.56 \times 16 = 8.96 = (8)_{10} = (8)_{16}$

16 | 988

16 | 61 → 12 (C)

3 → 13 (D)

$\therefore (988.86)_{10} = (3DC.DC28)_{16}$

② $(124)_{10} = (?)_{16}$

16 | 124

7 → 12 (C)

$\therefore (124)_{10} = (7C)_{16}$

* Hexadecimal to Decimal Conversion :-

① $(ABC.CD)_{16} = (?)_{10}$

$\overset{2}{A}\ \overset{1}{B}\ \overset{0}{C} . \overset{-1}{C}\ \overset{-2}{D}$

$A \times 16^2 + B \times 16^1 + C \times 16^0 + C \times 16^{-1} + D \times 16^{-2}$

$= 10 \times 16^2 + 11 \times 16^1 + 12 \times 16^0 + 12 \times \frac{1}{16} + 13 \times \frac{1}{64}$

$= (2748.80)_{10}$

* Octal to Decimal Conversion :-

① $(1234.56)_8 = (?)_{10}$

$= 1\times8^3 + 2\times8^2 + 3\times8^1 + 4\times8^0 + 5\times\frac{1}{8} + 6\times\frac{1}{64}$

$= 512 + 128 + 24 + 4 + 0.625 + 0.09375$

$= (668.71875)_{10}$

② $(0.625)_8 = (?)_{10}$

$= 0 + 6\times\frac{1}{8} + 2\times\frac{1}{64} + 5\times\frac{1}{512}$

$= (0.7910)_{10}$

③ $(1722)_8 = (?)_{10}$

$= 1\times8^3 + 7\times8^2 + 2\times8^1 + 2\times8^0$

$= (978)_{10}$

$(9CF1)_{16} = (?)_{10}$

$9\times16^3 + C\times16^2 + F\times16^1 + 1\times16^0$

$= (40177)_{10}$

* Octal to Hexadecimal Conversion :-

① $(726.625)_8 = (?)_{16}$

$2^4 = 16$

7 2 6 . 6 2 5

000 111 010 110 . 110 010 101 000 prefixed

post fixed

Group the binary - not into 4-bits

0001 1101 0110 . 1100 1010 1000

$= (1D6.CA8)_{16}$

② $(21056.375)_8 = (?)_{16}$

2 1 0 5 6 . 3 7 5

0010 001 000 101 110 . 011 111 101 000

$= (222E.7E8)_{16}$

* Hexadecimal to octal conversion :-

(1) $(1DG.CA8)_{16} = (?)_8$

= 0001 1101 0110 . 1100 1010 1000

← →

3 bit grouping

$= (0726.6250)_8$

(2) $(222E.7E8)_{16} = (?)_8$

2 2 2 E . 7 E 8

omit 000 010 010 010 1110 011 110 000

post final

$= (21056.375)_8$

Examples :-

(1) $(ABCD)_{16} = (?)_2 = (?)_8$

(2) Find the decimal value of $(204.2)_8$ or $(132.25)_{10}$

(3) Convert → $(532.65)_{10} = (?)_{16} = (?)_2$

$(ABCD)_{16} = (?)_{10}$ ~~$= (?)_8$~~

Binary Addition :-

The binary addition is performed with the help of following rules :-

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 =$ '0' sum bit '1' carry bit

Ex:- ① Add $(1010)_2$ & $(1111)_2$ in binary form

Carry from previous bit → 1 1

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

↑ Carry

* Binary subtraction :-

The binary subtraction is based on following rules:-

a. $0 - 0 = 0$

b. $0 - 1 = 11$ '1' is borrow bit, '1' is subtraction bit

c. $1 - 0 = 1$

d. $1 - 1 = 0$

e. $10 - 1 = 1$

Eg:-

```
  1 0 1 0
    1 1 1 1
  ---------
1 | 1 0 1 1
  ↑
Borrow bit
```

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

* Complements :-

→ 1's Complement

→ 2's Complement

→ 1's Complement representation :-

* The 1's Complement of a binary no is represented by replacing all 1's in a given binary no by 0's & replacing all 0's by 1's resulting no is 1's complement of a given binary no.

Eg:- Find 1's Complement of $(11011)_2$

```
  1 1 0 1 1
  0 0 1 0 0  ← 1's Complement form
```

$(10111011)_2$

$= 01000100$ ← 1's C form.

* 2's Complement representation :-

→ 2's C of a binary no is represented by taking 1's complement of given binary number & add '1' to the result.

ie, 2's Complement = 1's Complement + 1

Example :- ① $(10011)_2$ find 2's Complement

```
 10011
 01100 ← 1's C form
     1
 -----
 01101 ← 2's C form
 =====
```

② $(101000011)_2$

```
 010111100 ← 1's C form
       1
 ---------
 010111101 ← 2's C form
 =========
```

* Binary subtraction using 1's & 2's Complements

→ 1's Complement subtraction :-

It has two cases :-

a. Subtraction of smaller number from larger no
b. Subtraction of larger no from smaller no

a. Subtraction of smaller no from larger no :-

* Determine the 1's complement of a smaller no.
* Add 1's Complement of smaller no to the larger no.
* Add the carry to the result. The carry generated from the addition is called end around carry.

Eg:- ① Subtract $(10101010)_2$ from $(11100010)_2$
170 226

```
    10101010
step1: 01010101  ← 1's C

step2: 01010101
       11100010
       --------
      (1)00110111
       |
       ↓
   end around
   carry
```

Step 3: 0 0 1 1 0 1 1 1 ← Result without carry bit

\+ 1 ← Adding carry bit

0 0 1 1 1 0 0 0

Verification :-

2 2 6

-1 7 0

5 6

2 | 56
2 | 28 → 0
2 | 14 → 0
2 | 7 → 0
2 | 3 → 1
1 → 1

$(111000)_2$

(2) Subtract $(11010)_2 - (10111)_2$ using 1's C method.

1 0 1 1 1 ← smaller no

0 1 0 0 0 ← 1's C

1 1 0 1 0

0 1 0 0 0

1 0 0 0 1 0

1

0 0 0 1 1

b. Subtraction of larger number from smaller no.

Subtract

* Determine i's C of larger no or subtrahend.
* Add 1's C of larger no to the smaller no.
* The added result is in 1's C form. In order to get true answer take 1's C of the result & put a -ve sign for the true answer.

Ex:- (1) Subtract $(111\ 00010)_2$ from $(10101010)_2$ using 1's C method.

step 1:

$$\begin{array}{rl} 11100010 & \leftarrow \text{larger no} \\ 00011101 & \leftarrow \text{1's C} \end{array}$$

step 2:

$$\begin{array}{r} 10101010 \\ 00011101 \\ \hline 11000111 \end{array} \leftarrow \text{Result}$$

step 3:

$$00111000 \leftarrow \text{1's Complement of the result}$$

$$\therefore -(111000)_2$$

no.

Subtract $(10111)_2 - (11010)_2$ using 1's C method.

11010 ← larger no

00101 ← 1's C

$$\begin{array}{r} 10111 \\ +\ 00101 \\ \hline 11100 \end{array}$$ ← Result

$-(00011)_2$

* 2's Complement subtraction :-

It has too cases :-

→ Subtraction of smaller no from larger no

→ Subtraction of larger no from smaller no.

a. Subtraction of Smaller no from larger number :-

* Determine 2's complement of smaller no or Subtrahend.

* Add 2's C of smaller no to the larger no.

* The carry is generated, neglect the carry to get the true answer.

Subtraction

Eg:- ① Subtract $(101011)_2$ from $(111001)_2$ using 2's C method.

step 1: 101011 ← smaller no

010100 ← 1's C

1 ← Adding '1'

010101 ← 2's C

step 2:

010101

111001

(1)001110

↑ Carry

Neglect the Carry for true answer.

$(001110)_2$

② Subtract $(11010)_2$ from $(11101)_2$ using 2's C method.

step 1: 11010 ← smaller no

00101

1

00110

step 2:

00110

11101

(1)00011

↑ Carry

neglect the Carry to get true answer

$(00011)_2$

Subtraction of larger Number from smaller Number :-

* Determine 2's C of larger no or Subtrahend.

* Add 2's C of larger no to smaller no.

* The added result is in 2's C form in order to get true answer, take 2's C of added result & put -ve sign.

Example :- (1) Subtract $(111001)_2$ from $(101011)_2$ using 2's C method.

step 1: 111001 ← larger no

000110 ← 1's C

1 ← adding '1' to get 2's C

000111

step 2:

000 111
101 011
110010 ← result

step 3: 001101 ← 1's C of result

1 ← adding '1' to get 2's C

001110

$-(001110)_2$

② Subtract $(11101)_2$ from $(1010)_2$ using 2's C

step 1:

$$\begin{array}{r} 11101 \leftarrow \text{larger no} \\ 00010 \\ \quad\quad 1 \\ \hline 00011 \end{array}$$

step 2:

$$\begin{array}{r} 00011 \\ 11010 \\ \hline 11101 \end{array}$$

step 3:

$$\begin{array}{r} 00010 \\ \quad\quad 1 \\ \hline 00011 \end{array}$$

$= -(00011)_2$

Example:- ① perform (a) 1011 − 1001 using 1's C

(b) 1011 − 1001 using 2's C

② Subtract $(1010)_2$ from $(1100)_2$

③ Subtract the decimal no +28 & −19 using 2's C metha

soln: $(11100)_2$

2|28
2|14 → 0
2|7 → 0
2|3 → 1
1 → 1

$(10011)_2$

2|19
2|9 → 1
2|4 → 0
2|2 → 0
1 → 0

3

Introduction to digital logic :-

There are two types of signals namely

→ Analog

→ Digital

* Analog signals are continuously varying its amplitude w.r.t time.

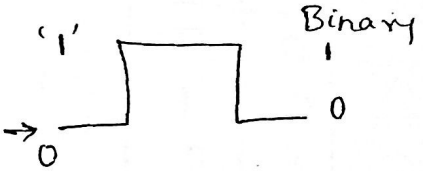
* Digital signals are sampled at a definite interval of time, which can be represented by only two possible amplitude levels which are high level (+5V) & low level (0V).

* There are two types of systems :-

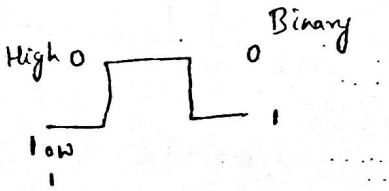
→ positive logic

→ Negative logic

Positive logic → High = 1 = 5V
Low = 0 = 0V



Negative logic → High = 0 = 5V
Low = 1 = 0V



* Boolean Algebra :-

It is used to simplify / rearrange Boolean Equati

to make simple logic circuit.

Properties of Boolean Algebra :-

→ Commutative property

→ Associative property

→ Distributive property

→ Duality property

* Commutative law or commutative property :-

Law-1 : A + B = B + A

Proof :-

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

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

The truth tables are identical. Therefore A OR B is same as B OR A.

Law 2: $AB = BA$

Proof:-

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

B	A	BA
0	0	0
0	1	0
1	0	0
1	1	1

Truth tables are identical ∴ A And B is same as B AND A

NOTE: Commutative law can be extended to any no of variables.

* Associative law :-

law 1: $A + (B + C) = (A + B) + C$

law 2: $A \cdot (BC) = (AB) \cdot C$

* Distributive law :-

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

* Duality :- It state that "for any given boolean relation it is possible to derive an another boolean relation by,

→ Changing (+) sign to an (·) (OR to AND)

→ Changing (·) sign to an (+) (AND to OR)

ie, $A + 0 = A$

$A \cdot 1 = A$

* DeMorgan's Theorem :-

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

ie, Complement of a product is equal to the sum of the complements.

Proof :-

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

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

ie, the complement of a sum is equal to the product of the complements.

Proof :-

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

les in Boolean Algebra :-

→ Any single variable or a function of several variables can have either 1 or 0 value.

→ The complement of a variable is represented by "bar" over the letter ie, $\overline{A}$ or sometime represented by a prime symbol (A').

* The logical AND function of two variables is represented by writing a "dot" ie, A.B b/n two letters or simply writing the two letters (AB).

"+" sign for OR fun (A+B) : '+' sign b/n the variables.

* Addition in Boolean algebra involves having either a binary '0' or binary '1' ie,

$$\left.\begin{aligned} 0+0 &= 0 \\ 0+1 &= 1 \\ 1+0 &= 1 \\ 1+1 &= 1 \end{aligned}\right\} \text{ OR function}$$

* Binary Multiplication involves,

$$\left.\begin{aligned} 0 \cdot 0 &= 0 \\ 0 \cdot 1 &= 0 \\ 1 \cdot 0 &= 0 \end{aligned}\right\} \text{ AND function}$$

* Other basic rules :-

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + A = A$
6. $A + \bar{A} = 1$
7. $A \cdot A = A$
8. $A \cdot \bar{A} = 0$
9. $\bar{\bar{A}} = A$
10. $A + AB = A$
11. $A + \bar{A}B = A + B$
12. $(A+B)(A+C) = A + BC$

Proof :- 10. $A + AB = A$

$LHS = A + AB$

$= A(1 + B)$ → 1

$= A$ = RHS

11. $A + \bar{A}B = A + B$

$LHS = A + \bar{A}B$ ∵ $A + AB = A$

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

$= A + B(A + \bar{A})$ → 1

$= A + B$ = RHS

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

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

$= AA + AC + BA + BC$

$= A + AC + AB + BC$

$= A + AB + BC$

$= A(1 + B) + BC$

$= A + BC$

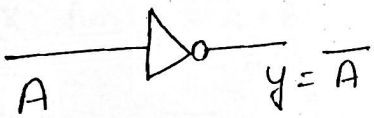
Logic Gates :-

→ The logic gates are the basic elements for designing of digital system. The operation of logic gates can be understood with the help of truth table.

→ The truth table is a tabulation of all possible i/p combinations & the corresponding o/p.

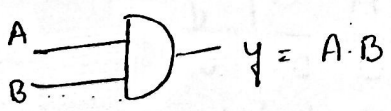
→ AND, OR & NOT are the basic gates & remaining gates are designed with combination of these gates.

* NOT gate :- (Inverter)



A	y
0	1
1	0

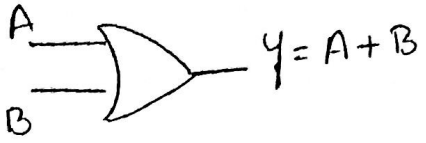
* AND gate :-



$2^2 = 4$

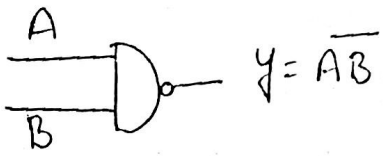
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

* OR gate :-



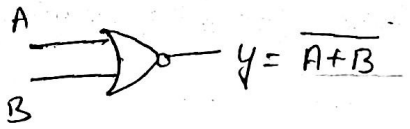
A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

* NAND gate :-



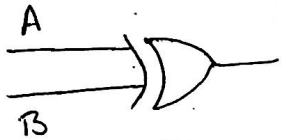
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

* NOR gate :-



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

* Ex-OR gate :-

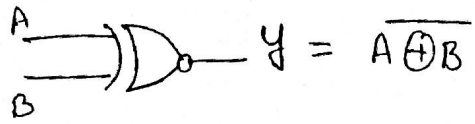


$Y = A \oplus B$

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

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

EX- NOR :-



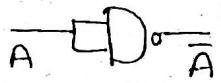
A	B	y
0	0	1
0	1	0
1	0	0
1	1	1

→ Universal gates :- NAND & NOR gate are universal gates because using these gates the basic fun like AND, OR, NOT & any Boolean fun can be implemented.

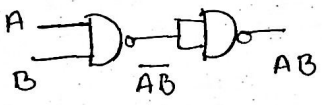
* NAND gate as Universal gate :-

→ NOT function using NAND :- $\bar{a}b = \bar{a} \cdot \bar{b}$

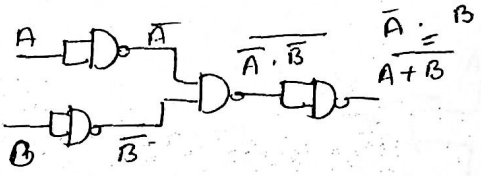
$\bar{A}b = a \cdot b$
$= \overline{\overline{a+b}}$
$= \overline{\bar{a} \cdot \bar{b}}$



→ AND fun



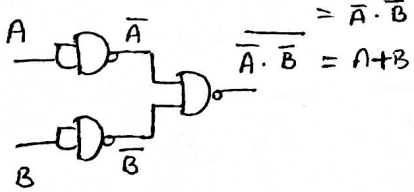
→ NOR fun



→ OR fun $y = A+B$

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

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



→ X- OR fun

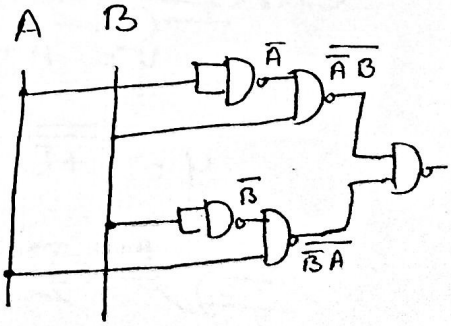
$Y = A \overline{\oplus} B$

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

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

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

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



→ X- NOR fun :-

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

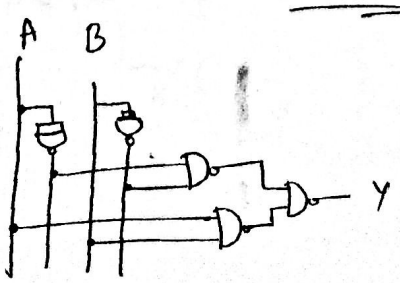
$= \bar{A}\bar{B} + AB$

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

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

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

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



* NOR as Universal gate :-

(i) NOT function :-

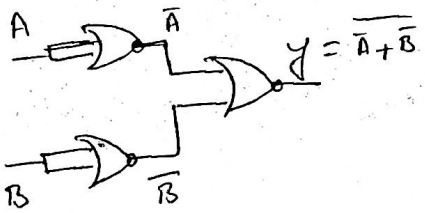
(ii) AND function :-

$y = A * B$

Apply De-Morgans theorem

$y = \overline{\overline{A * B}}$

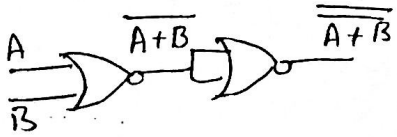
$= \overline{\overline{A} + \overline{B}}$



(iii) OR function :-

$y = A + B$

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

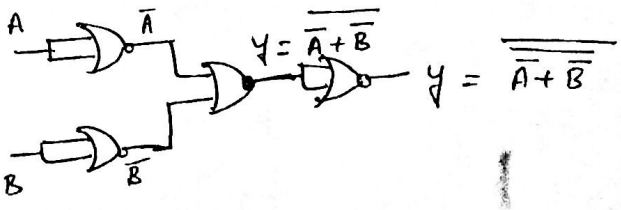


(iv) NAND :-

$y = \overline{AB}$

$y = \overline{\overline{AB}}$

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



X-OR fun using NOR :-

$$Y = A \oplus B$$

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

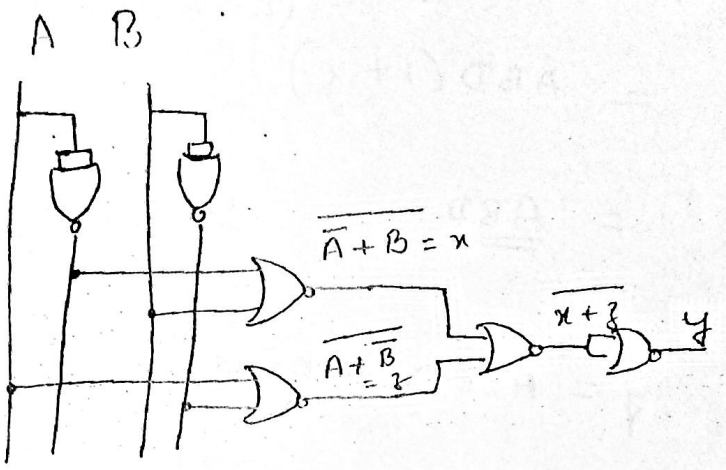
$$= \overline{\bar{A}B + A\bar{B}}$$

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

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

$$= \overline{\underbrace{A + \bar{B}}_{x}} + \overline{\underbrace{\bar{A} + B}_{z}}$$

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



(vi) XNOR using NOR :-

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

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

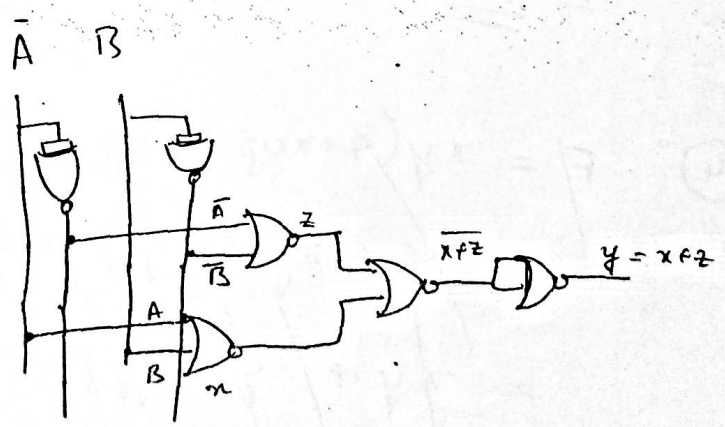
$$= \overline{\bar{A}\bar{B} + AB}$$

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

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

$$\underset{x+z}{} \quad = \overline{(A + B) \cdot (\bar{A} + \bar{B})}$$

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



Simplify the following expressions:-

① $Y = ABCD + ABD$

$= ABD(1 + C)$

$= \underline{ABD}$ $\qquad 1 + C = 1$

② $Y = ABCD + A\bar{B}CD$

$= ACD(B + \bar{B})$ → 1

$= \underline{ACD}$

③ $Y = AB + ABC + ABD + ABE$

$= AB(1 + C + D + E)$ → 1 $\qquad \because$ anything $+ 1 = 1$

$= \underline{AB}$

④ $F = xy + xyz + xy\bar{z} + \bar{x}yz$

$= xy + xy(z + \bar{z}) + \bar{x}yz$

$= xy + xy + \bar{x}yz$

$= xy +$

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

$= xy(1 + z) + xy\bar{z} + \bar{x}yz$

$= xy + xy\bar{z} + \bar{x}yz$

$= xy(1 + \bar{z}) + \bar{x}yz$

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

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

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

$$= \bar{C}$$

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

Apply DeMorgans theorem

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

$$= A[\bar{A} + \bar{A}B + \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{B} + \bar{B}\bar{C} + \bar{A}\bar{C} + B\bar{C} + \bar{C}]$$

$$= A[\bar{A} + \bar{A}B + \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{B} + \bar{B}\bar{C} + B\bar{C} + \bar{C}]$$

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

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

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

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

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

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

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

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

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

$y = \bar{A}BC + AC + AB$

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

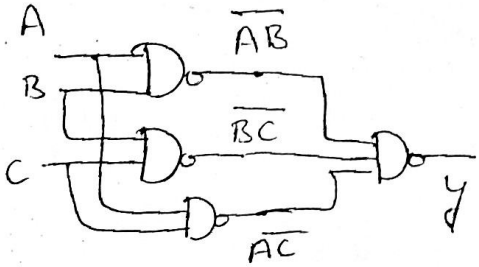
$= C(A + B) + AB$

$= AC + AB + BC$

$= AB + BC + AC$

$y = \overline{AB + BC + AC}$

$= \overline{AB} \cdot \overline{BC} \cdot \overline{AC}$

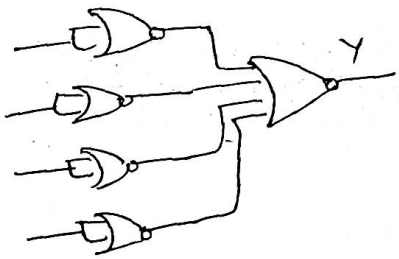


$y = ABCD$

Take double complement

$= \overline{\overline{ABCD}}$

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



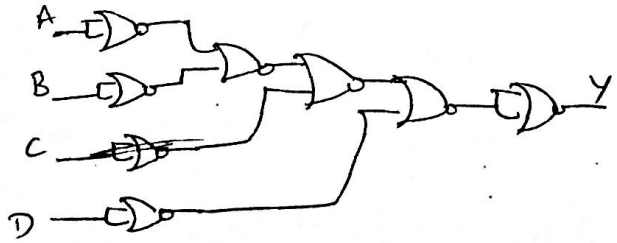
$y = \overline{\overline{AB}C\,D}$

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

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

Take double complement

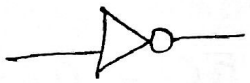
$= (\bar{A} + \bar{B}) + \bar{C} + \bar{D}$



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

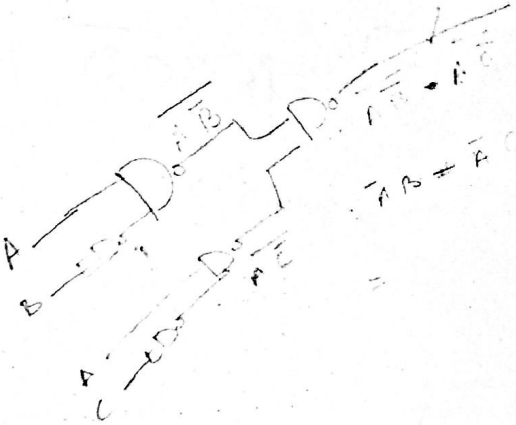
$$= \bar{A}\bar{C}(\bar{B}+B) + A\bar{C}(\bar{B}+B)$$

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



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

$$= \bar{C}$$



(2) $$y = A\left(\overline{ABC} + A\bar{B}C\right)$$

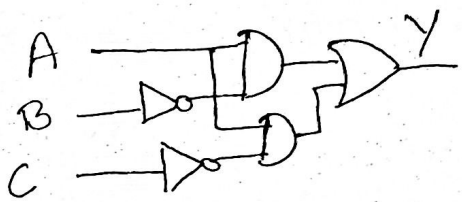
$$= A(\bar{A}+\bar{B}+\bar{C}) + A \cdot A\bar{B}C$$

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

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

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

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



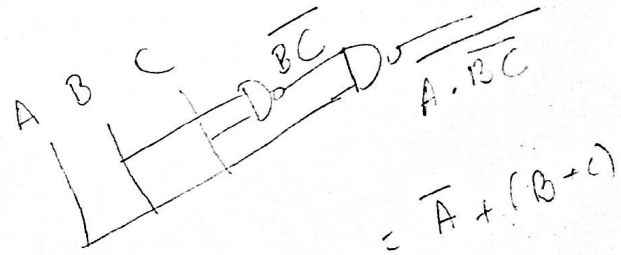
(3) $$\overline{A\bar{B} + \bar{A} + AB}$$

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

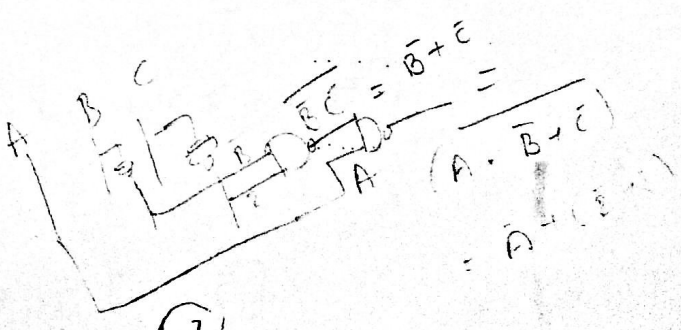
$$= \overline{\bar{A} + \bar{B} + B}$$

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

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



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



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

(4) $AB + \overline{AC} + A\overline{B}C(AB+C)$

(5) $\overline{ABC} + \overline{A}B\overline{C} + \overline{A}BC$

(6) $ABC + A\overline{B}C + AB\overline{C} = A(C+B)$

(7) $\overline{A}BC\overline{D} + BC\overline{D} + B\overline{CD} + B\overline{C}D$

Realize using basic, NAND & NOR gates

Prove:

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

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

$= [A\bar{B} + \bar{A}C][ABC]$

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

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

$= 0$

Simplify the following examples & realize using basic gates :-

(i) $y = (C + DE)(CE + DF)$ that is $\overline{ab} = \bar{a} + \bar{b}$

apply De-morgan's theorem to above expression until the complement is only to a single variables.

$$y = \overline{(C + DE) \cdot (CE + DF)}$$

Soln:

$$y = \overline{(C + DE) \cdot (CE + DF)}$$

$= (\bar{C} \cdot \overline{DE}) + (\overline{CE} \cdot \overline{DF})$

$= \bar{C}(\bar{D} + \bar{E}) + [(\bar{C} + \bar{E}) \cdot (\bar{D} + \bar{F})]$

$= \bar{C}\bar{D} + \bar{C}\bar{E} + [\bar{C}\bar{D} + \bar{D}\bar{E} + \bar{C}\bar{F} + \bar{E}\bar{F}]$

$= \bar{C}\bar{D} + \bar{C}\bar{E} + \bar{D}\bar{E} + \bar{C}\bar{F} + \bar{E}\bar{F}$

$= \bar{C}[\bar{D} + \bar{E} + \bar{F}] + \bar{E}[\bar{D} + \bar{F}]$

$= xy + \bar{x} yf$

$= y(x + \bar{x} f)$ $\quad a + \bar{a} b = a + b$

$= y(x + f)$

$= \underline{xy + yf}$

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

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

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

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

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

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

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

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

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

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

$= \underline{\bar{C}}$

three i/p Combinational ckt, the o/p is at logic '1' or logic high only when even no of 1's in the i/p combination. Design & realize using NAND gates.

Soln:-

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

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

double complement

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

$$= \overline{\overline{\bar{a}bc} \cdot \overline{a\bar{b}c} \cdot \overline{ab\bar{c}}}$$

(2) A four i/p combinational ckt, the o/p is at logic '1' or logic High when the number of 1's in a i/p combination is greater than or equal to three. Design a combinational ckt & realize using basic gates.

(3) perform the following statements, indicate the logic gates used

a) o/p is high if & only if all i/ps are high.

b) o/p is low if & only if all i/p's are high.

c) o/p is low if & only if all the i/p's are low

a)

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

b)

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

c)

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

* Half Adder :-

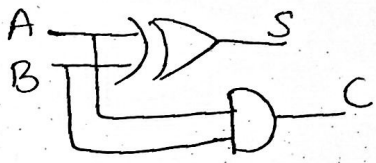
→ A combinational ckt which performs addition of two bits is called Half adder.

→ Since it has two binary i/ps & it provides two o/ps are Sum (S) & Carry (C).

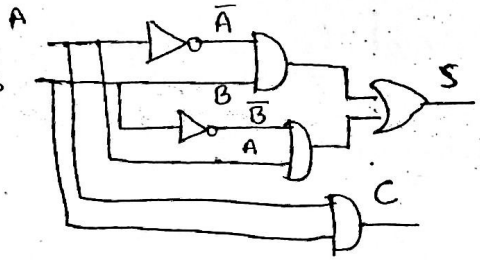
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

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

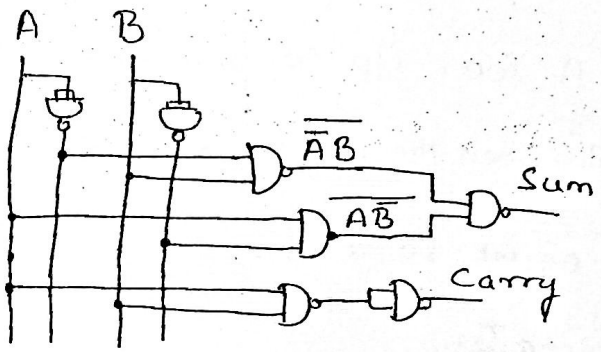
$C = AB$



Using basic gates :-



Using NAND gates :-



Using NOR gates :-

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

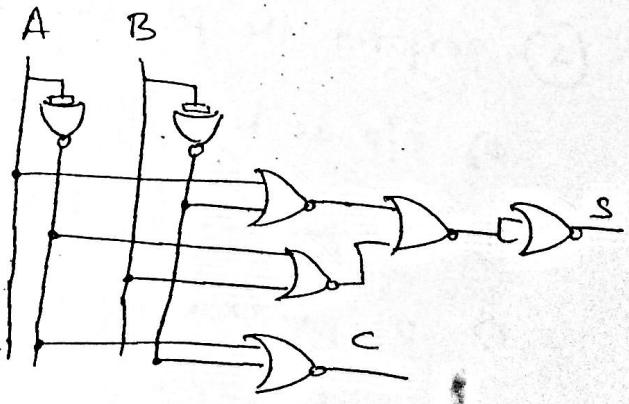
$= \overline{\overline{\bar{A}B} \cdot \overline{A\bar{B}}}$ $= \overline{\bar{A} + \bar{B}}$

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

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

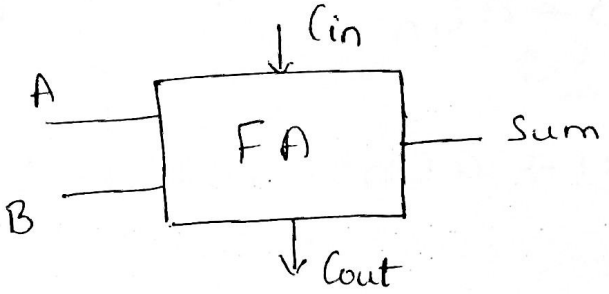
taking double complement

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



Full Adder :-

→ It is a Combinational ckt that forms the arithmetic sum of three i/p bits & 2 o/ps.



Block diagram

a	b	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = \bar{a}\,\bar{b}\,C_{in} + \bar{a}\,b\,\bar{C_{in}} + a\,\bar{b}\,\bar{C_{in}} + a\,b\,C_{in}$

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

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

$= C_{in}\,\overline{(a \oplus b)} + \bar{C_{in}}(a \oplus b)$

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

$C_{out} = \bar{a}\,b\,C_{in} + a\,\bar{b}\,C_{in} + a\,b\,\bar{C_{in}} + a\,b\,C_{in}$

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

$= \bar{a}\,b\,C_{in} + a\,\bar{b}\,C_{in} + ab$

$= \bar{a} b C_{in} + a(b + \bar{b} C_{in})$ $\quad \because b + \bar{b} C_{in} = b + C_{in}$

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

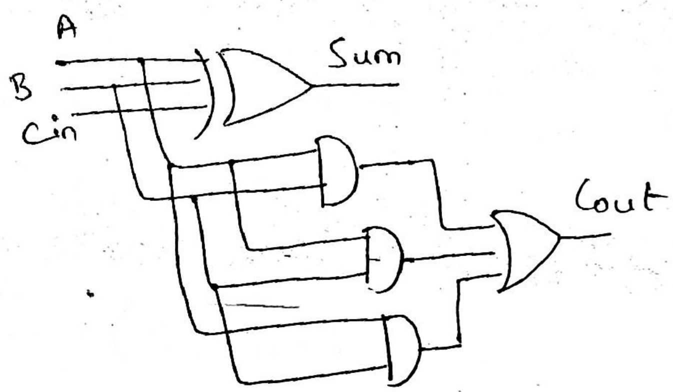
$= \bar{a} b C_{in} + ab + a C_{in}$

$= b(\bar{a} C_{in} + a) + a C_{in}$ $\quad (a + \bar{a} C_{in} = a + C_{in})$

$= b(a + C_{in}) + a C_{in}$

$= ab + b C_{in} + a C_{in}$

9845070670
→ Kiran



9448069430
Manohar

9845091022
— Jagnath

9731939966
— Vinod

logic diagram for full adder

9980829627
— Savitha

7349366046
— Revanna

NOR :-

$S = \overline{\overline{(A + B + \bar{C}_{in})} + \overline{(A + \bar{B} + C_{in})} + \overline{(\bar{A} + B + C_{in})} + \overline{(\bar{A} + \bar{B} + \bar{C}_{in})}}$

$C_{out} = \overline{(\bar{A} + \bar{B})} + \overline{(\bar{B} + \bar{C}_{in})} + \overline{(\bar{A} + \bar{C}_{in})}$

9741842064
— Srinath

9448158194
— Sriprakash

009715566169/8
— Muizz

9663539922

UNIT IV

FLIP FLOPS

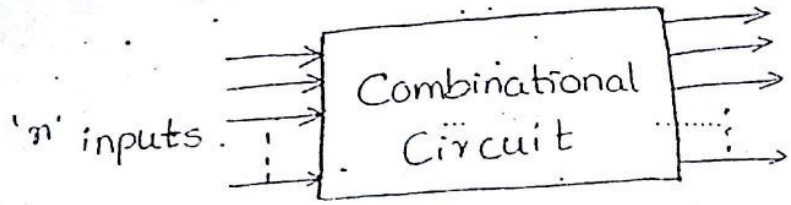
Manjunath Lakkanhavar
Assistant Professor
Dept. of Electronics and Communication
M.S. Ramaiah Institute of Tech
BANGALORE-560 054.

Introduction to Flip Flops.

→ Logic circuits for digital systems may be combinational or sequential.

→ A combinational circuit consists of logic gates whose outputs at any time are determined from the present combination of inputs.

→ A combinational ~~circuit~~ consists of input variables, logic gates, and output variables.

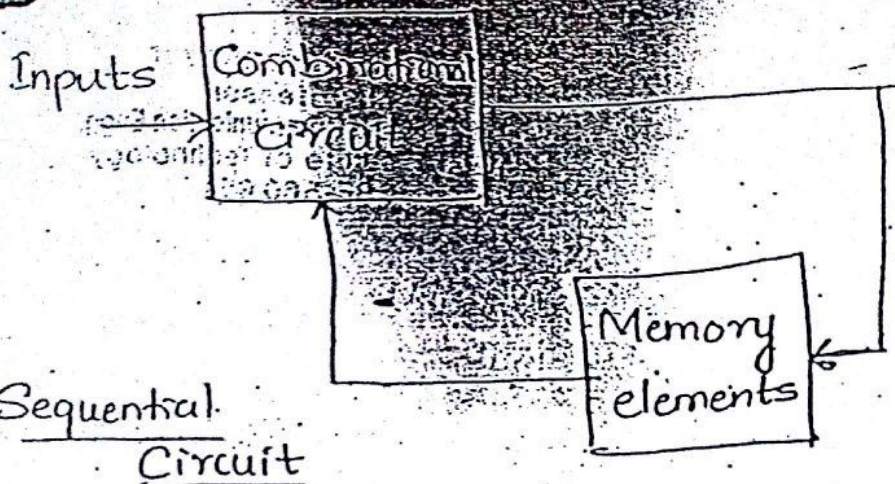


Combinational Circuit

→ A sequential circuit consists of combinational circuit, to which storage elements are connected to form a feedback path.

→ ~~The storage elements are~~ devices capable of storing binary information.

→ A logic circuit designed to produce a specified output for certain combination of input variables with storage facility, the resulting circuit is called sequential circuit shown below.



Sequential Circuit

→ The outputs to be generated that are not only dependent on the present input combinations or conditions but they also depend upon the past history of these inputs. The past history is provided by feedback from the output back to the input.

Comparision between Combinational & Sequential Circuits

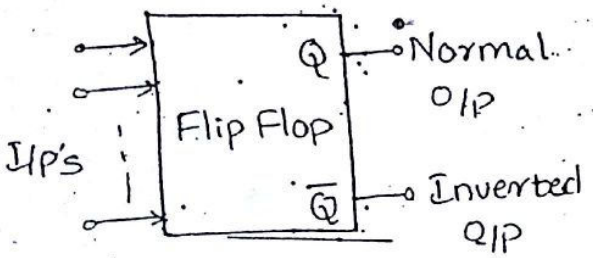
Combinational Circuits	Sequential circuits
1) The o/p variables dependent only on the combination of i/p variables.	(1) The o/p variables dependent not only on present i/p variables but also dependent on previous o/p variables.
2) Memory unit is not required	(2) Memory unit is required to store previous o/p's.
3) It is faster in speed.	(3) It is slower in speed.
4) It is easy to design.	(4) It is harder to design
5) eg: Parallel Adder	(5) Eg: Serial Adder.

Manjunath Lakkannavar
Assistant Professor
Dept. of Electronics and Communication Engg.
M.S. Ramaiah Institute of Technology,
BANGALORE-560 0..

The storage elements used in sequential circuits are called FLIP FLOPS.

→ A flip-flop is a binary storage device capable of storing one bit of information.

→ A sequential circuit may use many flip flops to store as many bits as necessary.



Flip-Flop Symbol

Output states

$Q = 1, \overline{Q} = 0$; It is called HIGH state also called SET state.

$Q = 0, \overline{Q} = 1$: It is called LOW or '0' state : also called RESET state.

Latch

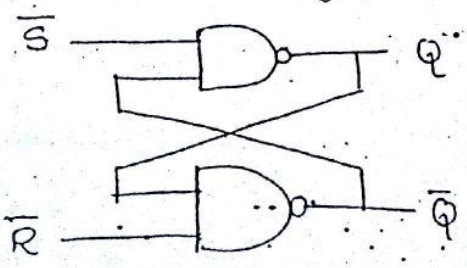
A Latch is the most basic type of flip flop circuit. It can be constructed using NAND or NOR gates. According the latch are of two types:

1) NAND gate latch
2) NOR gate latch.

Manjunath Lakkannavar
Assistant Professor
Dept. of Electronics and Communication Engg.
M.S. Ramaiah Institute of Technology.
BANGALORE-560 054.

1) NAND gate latch

→ As NAND gates are called as "Active Low" devices, i.e., it will be active only for low i/ps.

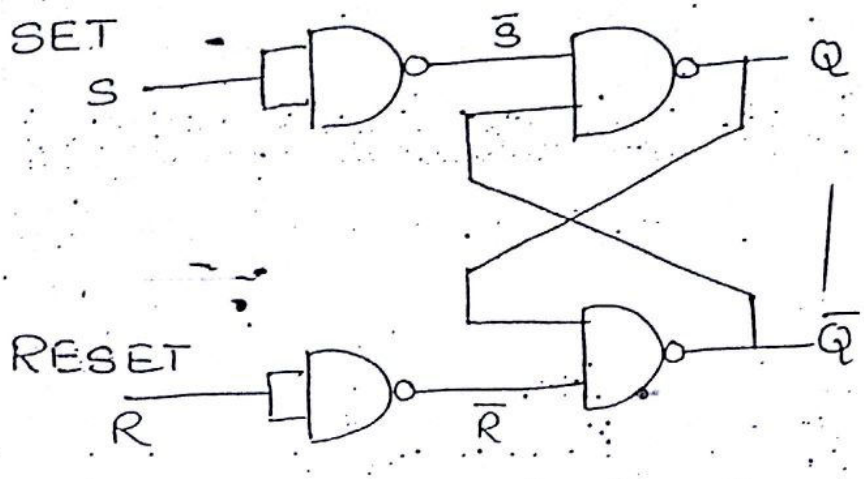


$\overline{S}\overline{R}$ Latch

Latch constructed from two cross-coupled NAND Gates

The $\bar{S}\bar{R}$- latch is given as shown above.

→ The alternative diagram for $\bar{S}\bar{R}$ latch can be constructed using true values of 'S' & 'R' as shown below & it is called as S.R. latch.



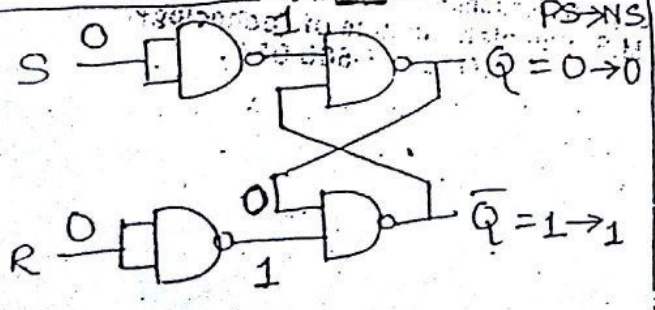
Manjunath Lakkannavar
Asst. Professor
Dept. of Electronics & Communication Engg.
M.S. Ramaiah Institute of Technology,
BANGALORE-560 054.

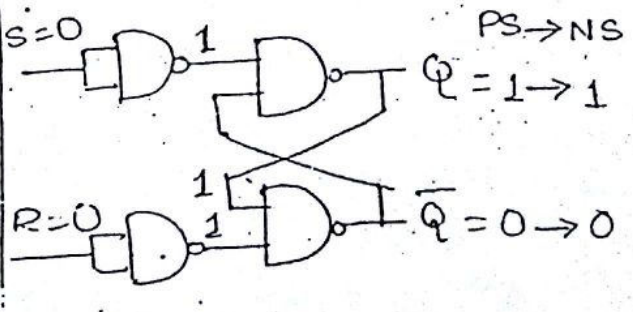
SR latch using NAND gates

Now let us see the working operation of SR latch using NAND gates.

→ Let us assume the previous o/ps are $Q=0$ & $\bar{Q}=1$, simultaneous will see for previous o/ps are $Q=1$ & $\bar{Q}=0$ for all combinations of i/ps.

Case 1: S=R=0



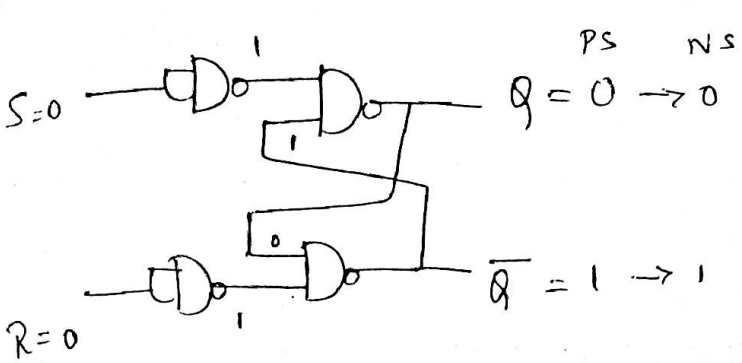


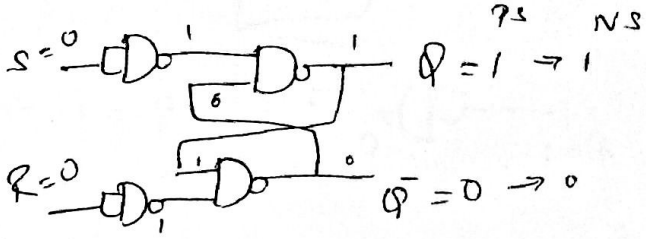
From the above diagrams; it is seen that for i/p combination S=0 & R=0, there is no change in the o/p's

→ It means, it is storing the same value of previous o/p's.

Let us assume the previous o/p's are $Q = 0$ & $\bar{Q} = 1$ (or) $Q = 1$ & $\bar{Q} = 0$.

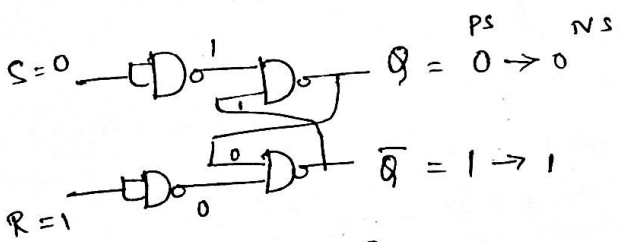
Case 1: $S = R = 0$

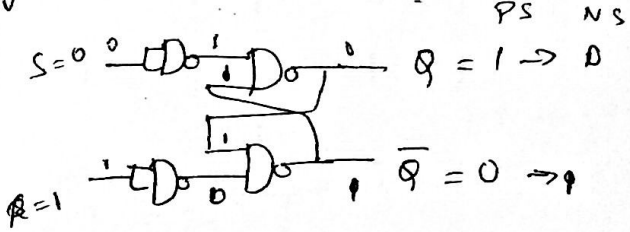




→ From the above diagram, it is seen that for i/p combination $S = R = 0$, there is no change in the o/p's it means it is storing the same value of previous o/p's.

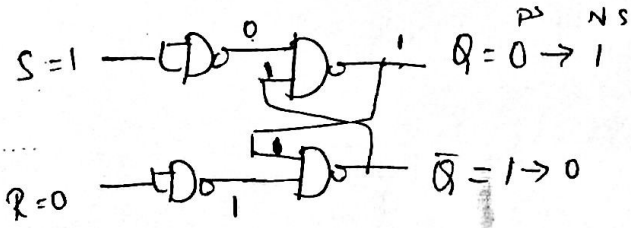
Case 2: $S = 0, R = 1$ (Reseting)

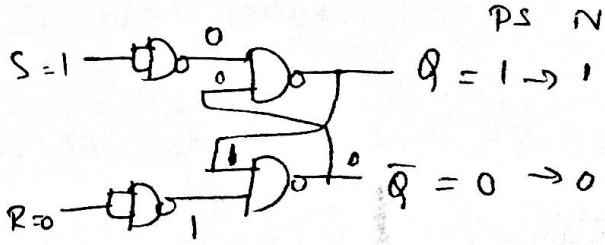




(start from Q)

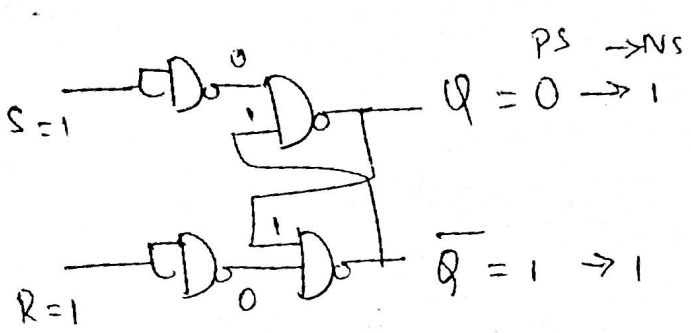
→ Case 3: $S = 1, R = 0$ (Setting)

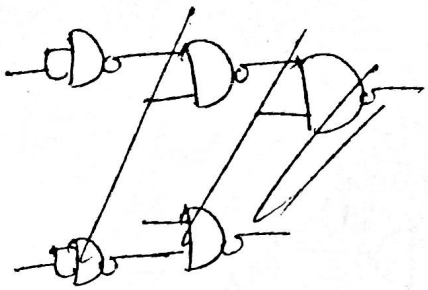


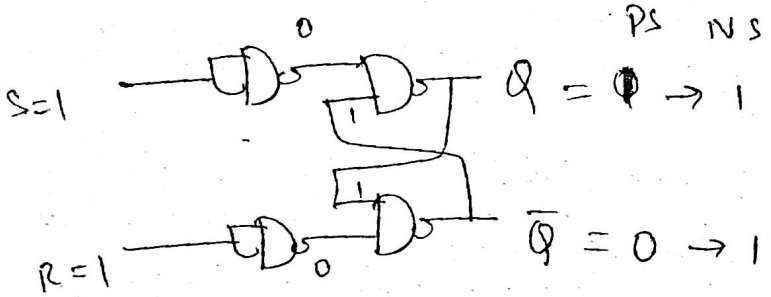


(start from $\bar{Q}$)

Case 4: S = R = 1



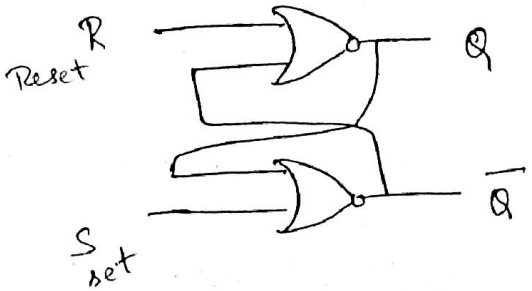


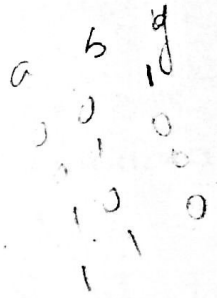


S	R	Q	Q̄	State
0	0	Q	Q̄	No change
0	1	0	1	Reset
1	0	1	0	set
1	1	1	1	Invalid

NOR gate latch :-

→ As NOR gates are called as "Active high" devices, it will be active only for high i/ps.

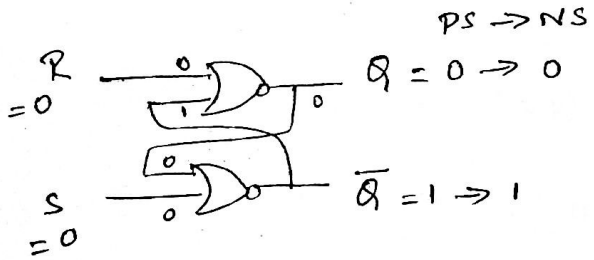


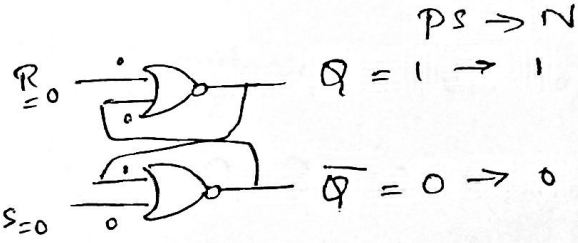


* NOR gate latch is constructed from two cross coupled NOR gates.

→ let us assume the previous o/p's as $Q = 0$ & $\bar{Q} = 1$ simultaneously will observe for $Q = 1$ & $\bar{Q} = 0$ for all combination of i/p's.

Case 1 :- $S = R = 0$

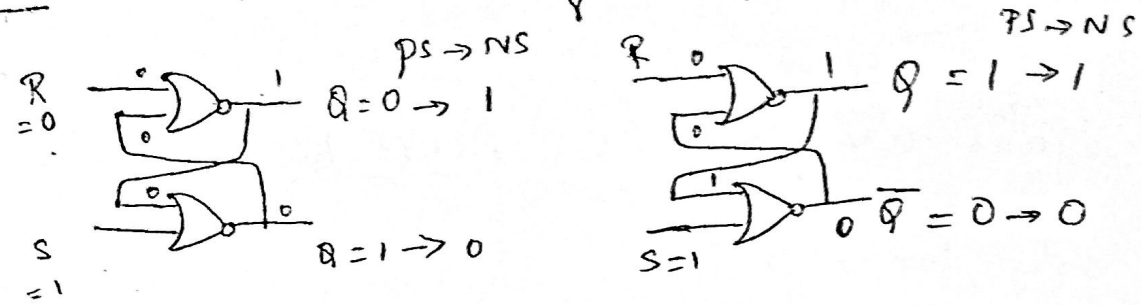




When $S = R = 0$ there is No Change in o/p's.

→ It means it is storing the same value of previous o/p's.

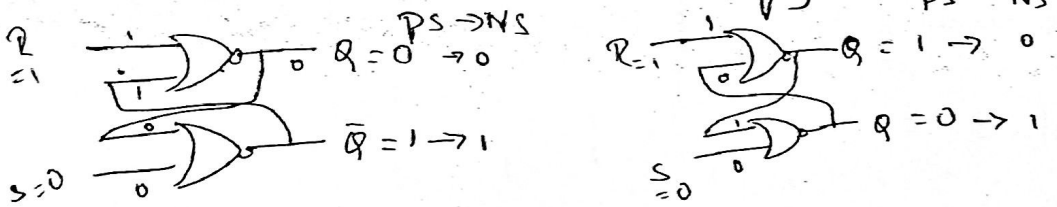
Case 2: $R=0$; $S=1$ (Setting Condition)



For i/p combination $R=0$ & $S=1$, if the normal o/p $Q=0$ set to $Q=1$, while if $Q=1$ remains as $Q=1$ ie, the latch is said to be in set state

[For setting the latch start from Q]

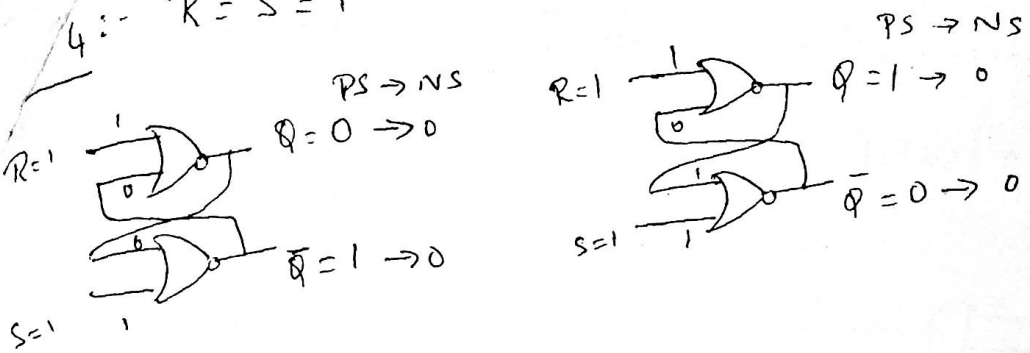
Case 3: $R=1$; $S=0$ [Resetting]



For i/p combination $R=1$ & $S=0$, if the normal o/p $Q=0$ remains as $Q=0$ while if $Q=1$ resets to $Q=0$ ie, The latch is said to be in Reset state.

Note: For resetting the latch start from $\bar{Q}$.

4:- R = S = 1



For i/p combination R = S = 1, the o/p's are no longer complements to each other ie, both Q = Q̄ = 0, which is invalid or indeterminate.

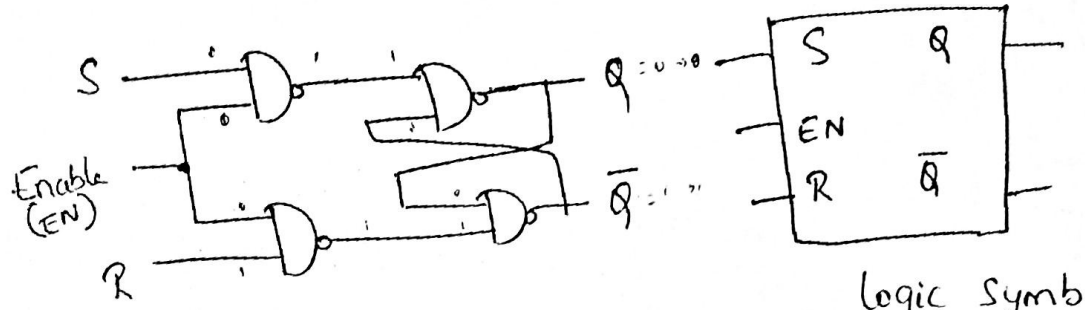
R	S	Q	Q̄	Status
0	0	Q	Q̄	No change
0	1	1	0	Set
1	0	0	1	Reset
1	1	0	0	Invalid

* The Gated SR Flip-Flop :-

→ In the latches discussed so far, the o/p responds immediately to change in the i/p.

→ However it can easily be modified to create a Flip flop that is sensitive to these i/ps only when an enable i/p is active.

→ Such a latch with enable i/p is known as gated SR flip-flop / Latch.



logic symbol

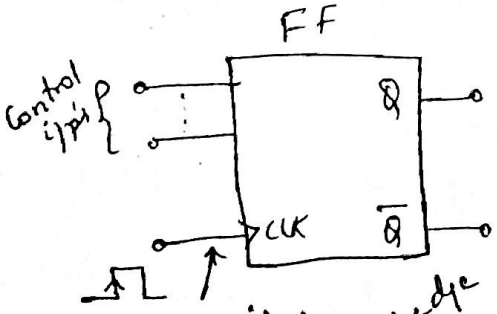
SR latch with Enable i/p using NAND gates

EN	S	R	Q	$\bar{Q}$	States
1	0	0	Q	$\bar{Q}$	No change
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	1	1	Invalid
0	x	x	Q	$\bar{Q}$	No change

→ When EN = 1, the ckt behaves like a SR latch

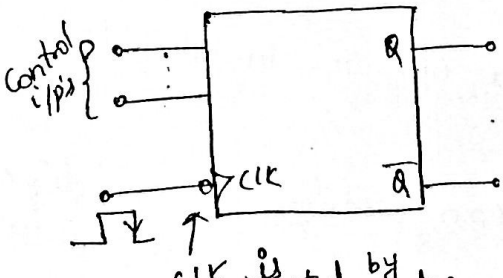
EN = 0, the ckt retains its previous state.

* Clocked Flip-flops :-



CLK is activated by a rising edge

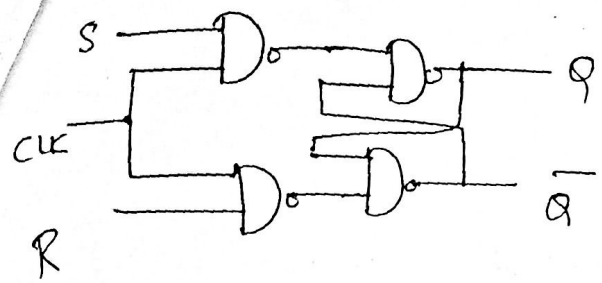
fig(a)

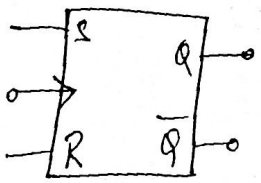


CLK is activated by a falling edge

fig(b)

Clocked SR ff :-





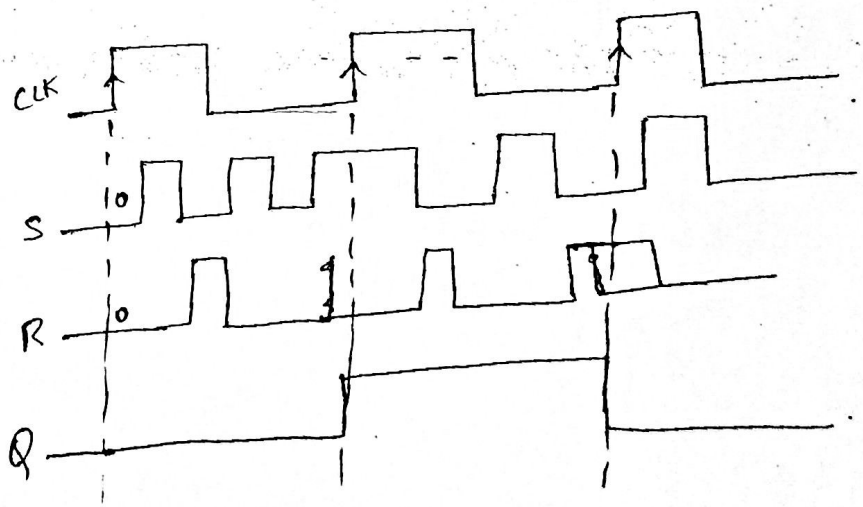
Logic Symbol

Clocked SR FF

→ The ckt shown above is similar to gated SR FF/latch except Enable signal is replaced by clock pulse/signal.

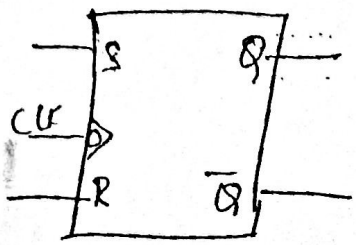
→ On the +ve edge of the clk pulse, the ckt responds to the S & R i/p's & behaves like a SR latch & retains its previous state when clk = 0.

CLK	S	R	Q	$\bar{Q}$	status
↑	0	0	Q	$\bar{Q}$	NC
↑	0	1	0	1	Reset
↑	1	0	1	0	set
↑	1	1	1	1	Invalid
0	x	x	Q	$\bar{Q}$	NC



Ily we have -ve edge triggered SR ff/Latch

CLK	S	R	Q	$\bar{Q}$	status
↓	0	0	Q	$\bar{Q}$	NC
↓	0	1	0	1	Reset
↓	1	0	1	0	set
↓	1	1	1	1	Invalid
1	x	x	Q	$\bar{Q}$	NC



Logic Symbol

* Introduction to Microcontroller :-

→ Microcontrollers are simple computing machines.

→ These are low power consuming processing machines.

Eg:- Security Systems

Camera, Toys, Laser printers ...

→ MC can be classified on the basis of their bits processed like 8-bit MC, 16 bit MC.

→ 8-bit MC called 8051 was introduced by Intel Corporation in 1981.

The main features of 8051 is as follows :-

1. 8-bit CPU with registers A & B both are 8 bits.
2. Sixteen bit program counter (PC) & data pointer (DPTR)
3. Eight bit program status word.
4. Stack pointer is a 8 bit register.
5. Internal ROM of 4KB & RAM of 128 bytes
6. Internal RAM of 128 bytes is divided into :-
   → Four register banks, each containing eight registers
   → Sixteen bytes of memory which is bit addressable