GOVERNMENT POLYTECHNIC DHENKANAL

NOTES ON

DIGITAL ELECTRONICS AND MICROPROCESSORS

PREPARED BY:- SMRUTIREKHA MOHANTY

1.1 Number Systems and Arithmetic

Signal:-

Signal is a physical quantity that depends on independent variable such as time, space, frequency etc.

→ The signal are of two types these are

(I) Analog signal

(II) Digital signal

(I) Analog Signal:-

The signal which is continous in nature that means the amplitude is change in every time.

(II) Digital signal:-

The signal which is discrete in nature that means for a particular time period the amplitude is constant.

→ The Digital signal are represented by the help of binary number system i.e '0 & 1'

→ The number system are of four types. These are

(I) Binary number system

(II) Decimal number system

(III) Octal number system

(IV) Hexadecimal number system.

(I) Binary number system

It is a type of number system in which the digital signal are represented by either "0" or "1".

(II) Decimal number system

It is a type of number system in which the digital signal are represented by 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

(III) Octal number system

It is a type of number system in which the digital signal are represented by 0, 1, 2, 3, 4, 5, 6, 7.

Hexadecimal number system :-

It is a type of number system in which the Digital signal are represented by 0, 1, 2, 3, 4 ...... 15.

→ In Hexadecimal number system 0 to 9 Digits are used and 10 is represented by A, 11 is represented by B, 12 is represented by C, 13 is represented by D, 14 is represented by E, and 15 is represented by F.

* The binary no. is reprented by $(\;)_2$
* The Decimal no. is reprented by $(\;)_{10}$
* The octal no. is reprented by $(\;)_8$
* The Hexadecimal no. is reprented by $(\;)_{16}$.

The steps for the converssion of Decimal no. into any base :-

Step-1

At the first step the given Decimal no. is converted into the base using sucessive devission by the base or radix by living its remainder.

Step-2

The sucessive devission of quotient can be repeated until the quotient is less than the base.

Step-3

After completion of devission collect the remainder from bottom to top.

Convert $(15)_{10}$ into the binary number ?

Ans:-

$$
\begin{array}{r|ll}
2 & 15 & \\
\hline
2 & 7 & 1 \\
\hline
2 & 3 & 1 \\
\hline
2 & 1 & 1 \\
\hline
 & 0 & 1
\end{array}
\uparrow
$$

$(15)_{10} = (1111)_2$

* Convert $(225)_{10}$ Decimal no. into the binary no.?

2	225	
2	112	1
2	56	0
2	28	0
2	14	0
2	7	0
2	3	1
2	1	1
	0	1

$$(225)_{10} = (11100001)_2$$

* Convert $(0.8125)_{10}$ into the binary no.

$$0.8125 \times 2 = 1.6250 \rightarrow 1$$

$$0.6250 \times 2 = 1.2500 \rightarrow 1$$

$$0.2500 \times 2 = 0.5000 \rightarrow 0$$

$$0.5000 \times 2 = 1.0000 \rightarrow 1$$

$$(0.8125)_{10} = (0.1101)_2$$

* Convert $(111.001)_2$ into Decimal no.

$(111.001)_2$

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

$= 4 + 2 + 1 + 0 + 0 + 0.125$

$= 7.125$

$$(111.001)_2 = (7.125)_{10}$$

* convert $(1101)_2$ into Decimal no.

Ans:- $(1101)_2$

$= 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$= 8+4+0+1$

$= 13$

* convert the binary no. $(1110\cdot110)_2$ into the Decimal no.

Ans:- $(1110\cdot110)_2$

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

$= 8+4+2+0+0.5+0.25+0$

$= (14\cdot75)_{10}$

$(1110\cdot110)_2 = (14\cdot75)_{10}$

* convert the Decimal no. into octal no.

Ans:- $(225)_{10} \rightarrow (\quad)_8$

8	225	
8	28	1
8	3	4
	0	3

$(225)_{10} = (341)_8$

* convert $(1225)_{10}$ into octal no.

Ans!-

8	1225	
8	153	1
8	19	1
8	2	3
	0	2

$(1225)_{10} = (2311)_8$

* convert $(73\cancel{1})_8$ into the Decimal no.

Ans!- $(731\cdot151)_8$

$= 7\times8^2 + 3\times8^1 + 1\times8^0 + 1\times8^{-1} + 5\times8^{-2} + 1\times8^{-3}$

$= 448 + 24 + 1 + 0\cdot125 + 0\cdot078 + 0\cdot001$

$= 473\cdot204$

$(731\cdot151)_8 = (473\cdot204)_{10}$

* Convert $(731)_8$ into the Decimal no.

Ans:- $(731)_8$

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

$= 448 + 24 + 1$

$= 473$

$(731)_8 = (473)_{10}$

* Convert the Decimal no. into the Hexadecimal no.

Ans:- $(1225)_{10} \rightarrow (\ \ )_{16}$

16	1225	
16	76	9
16	4	C
	0	4

$(1225)_{10} = (4C9)_{16}$

* Convert $(1225.125)_{10}$ into Hexadecimal no.

Ans:-

16	1225	
16	76	9
16	4	C
	0	4

$0.125 \times 16 = 2.000 \rightarrow 2$

$(1225.125)_{10} = (4C9.2)_{16}$

* Convert $(1225.125)_{10}$ into octal

Ans:-

8	1225	
8	153	1
8	19	1
8	2	3
	0	2

$0.125 \times 8 = 1.000 \rightarrow 1$

$(1225.125)_{10} = (2311.1)_8$

* Convert $(11C.1)_{16}$ into Decimal no.

Ans:- $(11C.1)_{16}$

$= 1\times16^2 + 1\times16^1 + C\times16^0 + 1\times16^{-1}$

$= 256 + 16 + 12 + 0.0625$

$= 284.0625$

$(11C.1)_{16} = (284.0625)_{10}$

Convert the binary no. into the octal no :-

step-1

First write the binary no.

step-2

Make groups of three bits starting from the binary points.

Step-3

If the last group contain less than three bits then we assume the remaining bits to be zero.

Step-4

for each three bits group find the octal digit.

Step-5

To get the result place the octal digit into the same order.

Octal	Binary
0 →	000
1 →	001
2 →	010
3 →	011
4 →	100
5 →	101
6 →	110
7 →	111

* Convert $(111001110)_2$ into octal no.

Ans:-

$$\underbrace{111}_{7}\,\underbrace{001}_{1}\,\underbrace{110}_{6}$$

$$(111001110)_2 = (716)_8$$

* Convert $(10111101 \cdot 11101)_2$ into octal no.

Ans:-

$$\underbrace{010}_{2}\,\underbrace{111}_{7}\,\underbrace{101}_{5}\cdot\underbrace{111}_{7}\,\underbrace{010}_{2}$$

$$(10111101 \cdot 11101)_2 = (275 \cdot 72)_8$$

conversion of binary no. into the Hexadecimal no.

step-1

first write the binary no.

step-2

Make groups of four bits starting from the binary points.

step-3

If the last group contains less than four bits then we assume the remaining bits to be zero.

step-4

For each four bits groups find the binary digit.

step-5

To get the result place the binary digit into the order.

Hexadecimal	Binary
0 →	0000
1 →	0001
2 →	0010
3 →	0011
4 →	0100
5 →	0101
6 →	0110
7 →	0111
8 →	1000
9 →	1001
A →	1010
B →	1011
C →	1100
D →	1101
E →	1110
F →	1111

* Convert $(1100101\cdot110)_2$ into Hexadecimal no.

Ans:- $(1100101\cdot110)_2$

$= \underbrace{0110}_{6}\ \underbrace{0101}_{5}\cdot\underbrace{1100}_{C}$

$(1100101\cdot110)_2 = (65\cdot C)_{16}$

* Conversion of Hexadecimal no. into the Binary

Ans:- $(FC\cdot B)_{16} \to (11111100\cdot1011)_2$

$(BF\cdot F)_{16} \to (10111111\cdot1111)_2$

1.8 Binary Addition

Rules :- 0+0 = 0

0+1 = 1

1+0 = 1

1+1 = 0 with carry 1

Eg :- $(1101\cdot101)_2 + (111\cdot011)_2$

$$\begin{array}{r} 1101\cdot101 \\ +\ 111\ 011 \\ \hline 10101\cdot000 \end{array}$$

Binary Subtraction

Rules :- 0-0=0

1-1=0

0-1=1 , with borrow of 1

1-0=1

$$\begin{array}{r} 1010\cdot010 \\ 0111\cdot111 \\ \hline 0010\cdot011 \end{array}$$

Binary Multiplication

Rules :-
$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

Eg:- $(101)_2 \times (101)_2$

```
    101
  x 101
  -----
    101
   000x
  101xx
  -----
  11001
```

Binary Division

Eg:- $(101101)_2 \div (110)_2 = ?$

```
110 ) 101101 ( 111.1
      110
      ----
      1010
       110
      ----
      1001
       110
      ----
      0110
       110
      ----
      0000
```

1.3 Complements

Complements is a Method which is used for Subtraction purpose.

→ usually complements are of two types

(I) $(r-1)$'s complement.

(II) r's complement.

Here $r \rightarrow$ radix

(I) $(r-1)$'s complement :-

In $(r-1)$'s complement all the digits are substracted from $(r-1)$

(II) r's complement :-

In r's complement first we have find out $(r-1)$'s complements.

→ Then we are adding '1' to get the r's complements

→ The r's complement can also find out by using the formula $\boxed{r^n - N}$.

Here r = The base

n = The number of digit

N = The given no.

Eg:-

* find out the 9's complement & 10's complement of $(79)_{10}$

Ans:- Here $r = 10$

$(r-1)$'s $= 9$

$$\begin{array}{r} 99 \\ 79 \\ \hline (20) \end{array} \text{ 9's complement}$$

$$\begin{array}{r} +1 \\ \hline 21 \end{array} \text{ 10's complement}$$

or 10's complement $= r^n - N$

$= 10^2 - 79$

$= 100 - 79$

$= 21$

* find out the 1's complement and 2's complement of $(1101)_2$

Ans:- Here $r = 2$

$r - 1 = 1$

1's complement $=$ 1111
$$\begin{array}{r} 1111 \\ 1101 \\ \hline 0010 \end{array}$$

2's complement $= (0010 + 1) = (0011)_2$

Or

2's complement $= r^n - N$

$= 2^4 - 1101$

$= 10000 - 1101$

$= (0011)_2$

* find out the 1's and 2's complement of $(0110)_2$

Ans:- Here $r = 2$

$r - 1 = 1$

1's complement $=$
$$\begin{array}{r} 1111 \\ 0110 \\ \hline 1001 \end{array}$$

2's complement $= (1001 + 1)_2 = (1010)_2$

Or

2's complement $= r^n - N$

$= 2^4 - 1001 = 10000 - 1001$

$= (1010)_2$

1.4 Substraction of binary number in 2's complement Method

step-1

First find out the 2's complement of substrahend

step-2

Then add the minuend with the 2's complement of substrahend

step-3

If the end carry is 1 then discard the carry and take the result as positive.

Step-4

If their is no carry then the result of the 2's complement of that no. is taken as negative.

* Eg. subtract $(1011)_2$ from $(1111)_2$ using 2's complement.

Ans:- 1111 → Minuend
1011 → Subtrahend

$$\begin{array}{r} 1111 \\ -\ 1011 \\ \hline 0100 \\ +\ \ \ \ 1 \\ \hline 0101 \end{array}$$

0100 → 1's
0101 → 2's

Add minuend + 2's complement ~~of subtrahend~~

$$\begin{array}{r} 1111 \\ +\ 0101 \\ \hline 10100 \end{array}$$

$(1111)_2 - (1011)_2 = (0100)_2$

* Subtract $(1111)_2$ from $(1010)_2$ using 2's complement

Ans:- 1010 → minuend
1111 → Subtrahend
2's complement

Subtraction of two decimal no. using 10's complements :-

Step-1

first find out the 10's complement of Substrahend

Step-2

Then add the minuend with the 10's complement of substrahend

Step-3

If the end carry is 1 then discard the carry and take the result as positive.

Step-4

If their is no carry then the result of the 2's complement of that no. is taken as negative.

* Eg. subtract $(53)_{10}$ from $(77)_{10}$

Ans:- 77 → minuend

53 → substrahend

10's complement

$$\begin{array}{r} 99 \\ 53 \\ \hline 46 \\ +1 \\ \hline 47 \end{array}$$

Add minuend + 10's complement

$$\begin{array}{r} 77 \\ +47 \\ \hline 124 \end{array}$$

$(77)_{10} - (53)_{10} = (24)_{10}$

1.5. Use of weighted and un-weighted Codes & write Binary equivalent number for a number in 8421, Excess-3 and Gray code and vice-versa.

Digital codes

If the Magnitude of decimal number is very large, Then it is difficult to find out the binary numbers using division.

→ To overcome this problem we can use code numbers for each decimal digits and the codes are called as digital codes.

→ The digital codes are of two types.

(i) weighted codes

(ii) unweighted codes

(i) weighted codes:-

The codes in which each digit position is assigned with a weight is called as weighted codes.

Eg:- BCD codes.

BCD codes

→ BCD stands for Binary coded Decimal code.

→ BCD codes are also called as 8421 code.

→ In this code each decimal digit is represented by a 4bit binary number.

→ In BCD codes for each decimal no. (0 to 9) a BCD code is assigned.

Decimal	BCD code
0 →	0000
1 →	0001
2 →	0010
3 →	0011
4 →	0100
5 →	0101
6 →	0110
7 →	0111
8 →	1000

* convert $(53)_{10}$ into BCD no.

Ans:- $(53)_{10} = (01010011)_2$

* convert $(99)_{10}$ into BCD no.

Ans:- $(99)_{10} = (10011001)_2$

* convert $(75)_{10}$ into BCD no.

Ans:- $(75)_{10} = (01110101)_2$

* convert the BCD $(1001\,0011\,0100)_2$ into Decimal.

Ans:- $(1001\,0011\,0100)_2 = (934)_{10}$

BCD Addition

Step-1

Add two BCD number using binary addition

Step-2

If the result is greater than 9, then add binary 6 with the four bit result to get a valid BCD no.

Step-3

If the result produces end carry then the binary 6 (0110) Must be added with each four bit BCD number.

Step-4

If the result is equal to or less than 9 then it is a valid BCD no.

* Add $(7)_{10}$ with $(3)_{10}$ using BCD codes?

Ans:- $(7)_{10} = 0111$
$(3)_{10} = 0011$

$$\begin{array}{r} 0111 \\ +\,0011 \\ \hline 1010 \\ +\,0110 \\ \hline \underbrace{0001}_{1}\underbrace{0000}_{0} \end{array}$$

$(7)_{10} + (3)_{10} = (00010000)_2$

* Add $(88)_{10}$ with $(88)_{10}$ using BCD codes?

Ans :- $(88)_{10}$ = 10001000

$(88)_{10}$ = 10001000

```
   10001000
 + 10001000
 ----------
  100010000
 + 01100110
 ----------
000101110110
```

0001 → 1

0111 → 7

0110 → 6

$(88)_{10} + (88)_{10} = (176)_{10}$

(ii) Unweighted code :-

The codes inwhich the digital bits does not have weighted position value is called as unweighted codes.

Eg:- Excess-3 codes
Gray codes

(i) Excess-3 Codes

The BCD numbers are converted into the excess-3 codes by adding "3" with each BCD codes.

Decimal	BCD	Excess-3
0 →	0000 →	0011
1 →	0001 →	0100
2 →	0010 →	0101
3 →	0011 →	0110
4 →	0100 →	0111
5 →	0101 →	1000
6 →	0110 →	1001
7 →	0111 →	1010
8 →	1000 →	1011
9 →	1001 →	1100

Gray code conversion binary no. into the Gray code:

Step-1

The MSB of binary code is same as gray code.

Step-2

To get the next gray code bit add the MSB of Binary with next bit of Binary.

Step-3

To get further next bit of Gray code add the consqutive next bit & previous bit. Repeat it until get LSB of gray code.

* Convert $(1111)_2$ into gray code

Ans:- (1111)

MSB → LSB

1111

↓↓↓↓

1000

* find the gray code of $(101101)_2$

Ans:- 101101

↓ MSB ↓ LSB

101101

↓↓↓↓↓↓

111011

→ Gray code is also known as cyclic code.

→ This code is used for error checking & correction in digital communication.

Conversion of gray code into the binary :-

Step-1

The MSB in gray code same as the MSB of binary.

Step-2

To get the next binary number add the binary MSB with the next significant bit.

Step-3

Record the result neglecting carry and continue the process unit LSB is

Step-4

The no. of binary bits same as the number of gray code bits.

\# Find the binary code for gray code $(10111)_2$

Ans :-

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

$$(10111)_2 = (11010)_2$$

ASCII code :-

ASCII is American standard code for information Interchange.

→ The ASCII code is seven bit code

1.6 Importance of Parity Bit

Parity bit is an extra bit that to be added with data bits to detect the errors appeare in the digital transmission.

→ A bit either 1 or 0 is added as a parity bit

→ parity bit is two types
(i) Even parity
(ii) odd parity

(i) Even parity

In Even parity the parity bit is selected as 0 if number of "1" present in the data is even, while the parity bit is selected "1" if number of "1" present in the data is odd.

(ii) odd parity

In odd parity the parity bit is selected as "0" if number of 1 is present in the data is odda, while the parity bit is selected "1" if number of "1" present in the data is even.

Data bit	Even parity	odd parity
0000	0	1
0001	1	0
0010	1	0
0011	0	1
0100	1	0
0101	0	1
0110	0	1
0111	1	0
1000	1	0
1001	0	1
1010	0	1
1011	1	0
1100	0	1

	Even	Odd
1101 →	1	0
1110 →	1	0
1111 →	0	1

1.7 Logic Gates:-

It is an electronics circuit having one or more than one input and only one output.

→ Logic gates are the basic building blocks of any digital system.

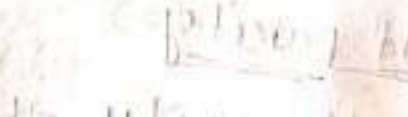
→ Usually 8 no. of logic gates are used. These are
1. NOT
2. AND
3. OR
4. NAND
5. NOR
6. EX-OR
7. EX-NOR
8. Buffer

NOT Logic gates

→ In NOT gates if the input is high the output is low and if the input is low output is high

→ If the input to the NOT gate is A then the output to the NOT gate is $y = \overline{A}$

→ The symbol of NOT gate is



Truth Table for NOT gate

A	$y = \overline{A}$
0	1
1	0

AND gate :-

→ In AND gate two inputs are used to get the output.

→ If both the input are high then the output is high otherwise output is 0.

→ The symbol of AND gate is 

→ suppose the input to the AND gate is A&B then the output of AND gate is A·B.

Truth Table of AND gate

A	B	Y = A·B
0	0	0
0	1	0
1	0	0
1	1	1

OR gate

→ In OR gate two inputs are used. If any of the input is high then the output is high Other.wise it is 0.

→ suppose the input to the OR gate is A & B the the output of OR gate is A+B

→ the symbol of OR gate is 

Truth Table of OR gate

A	B	y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

NAND gate

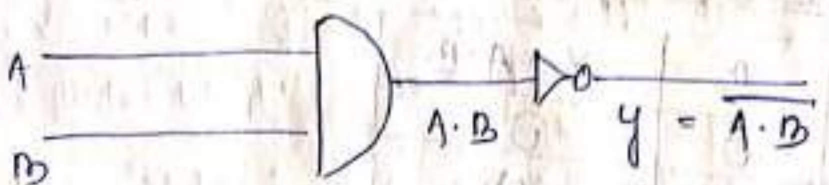
NAND gate = NOT gate + AND gate

→ In NAND gate we have two input & one output

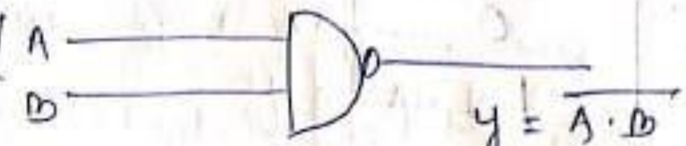
→ NAND logic gates if any of the input is low then the output is high.

→ If A & B are the inputs to the NAND gate then the output y is $(\overline{A \cdot B})$

→ The symbol of NAND gate is



⇒



Truth Table of NAND gate

A	B	A·B	$Y = \overline{A \cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

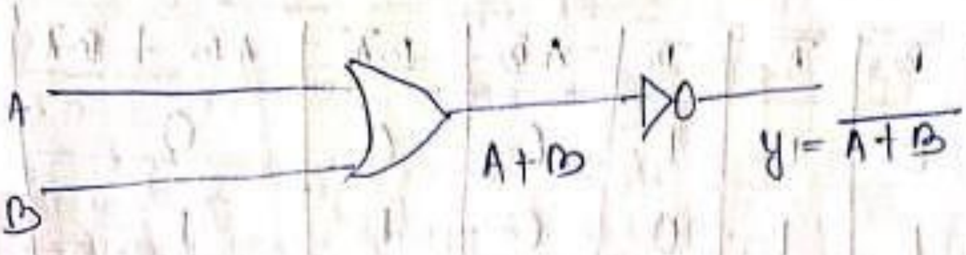
NOR gate:-

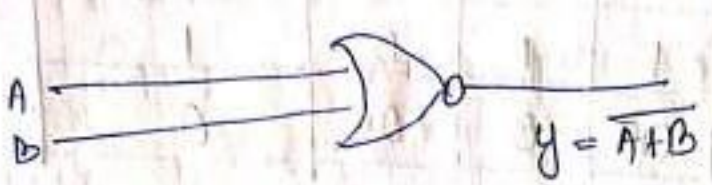
In NOR gate if both the inputs are low then the output is high.

→ In NOR gate we have two inputs and one outputs.

→ If the inputs are A & B then the output is $y = \overline{A + B}$

→ The symbol of NOR gate is





Truth Table for NOR gate

A	B	A+B	$y = \overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

EX-OR gate :-

In EX-OR gate if the either input is high then the output is high.

→ when the inputs are A&B then the output of the EX-OR gate is

$$y = A \oplus B$$

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

→ The symbol of EX-OR gate is

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

Truth Table for EX-OR gate:-

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

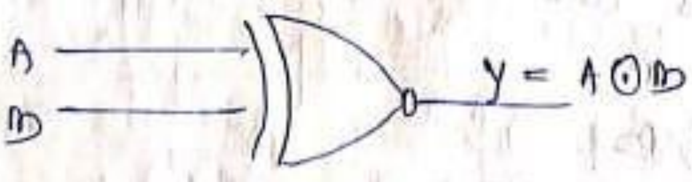
EX-NOR gate:-

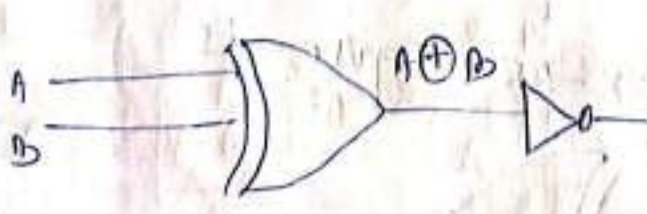
→ In EX-NOR gate, if both the inputs are equal then the output is high.

→ If the inputs are A&B then the output of the EX-NOR gate is

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

→ The symbol of EX-NOR gate is





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

Truth Table for EX-NOR gate :-

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

Buffer gate :-

→ In Buffer gate what ever inputs are given the same outputs are taken.

→ The symbol of Buffer gate is

A $\qquad y = A$

If $\bar{A}$ $\qquad y = \bar{A}$

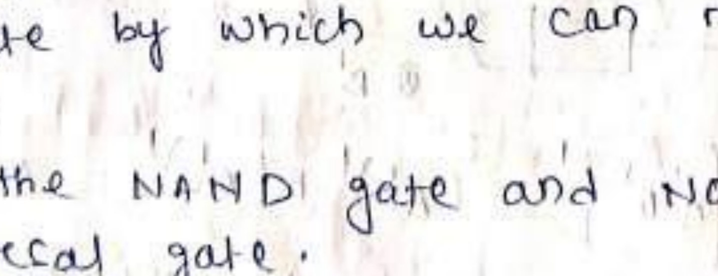
1.8 Realize AND, OR, NOT operations using NAND NOR gates.

Universal gate :-

→ It is a gate by which we can realised all the gates.

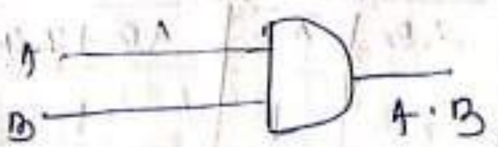
→ Usually the NAND gate and NOR gate are the universal gate.

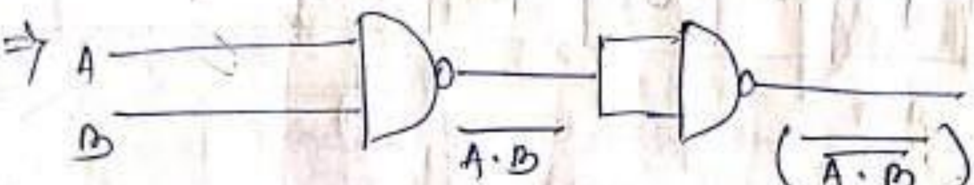
Realization of NOT gate using NAND gate :-



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

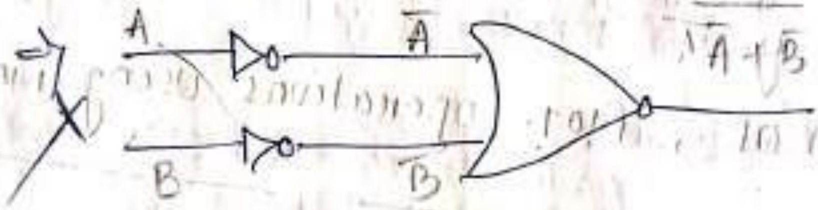
Realization of AND gate using NAND:-



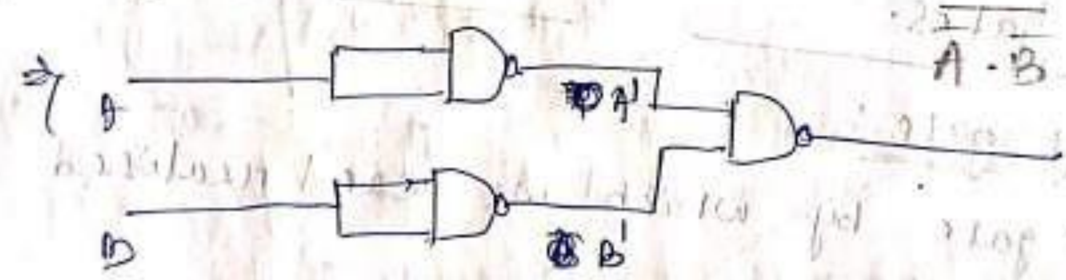


$$\left(\overline{\overline{A \cdot B}}\right) = A \cdot B$$

Realization of OR gate using NAND:-



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

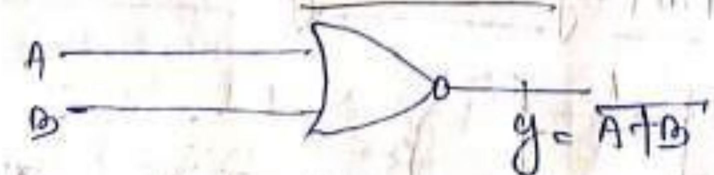


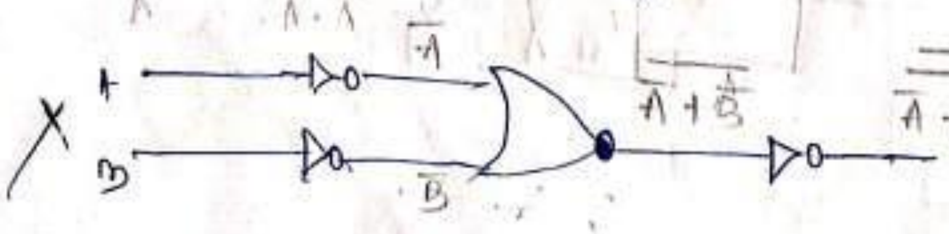
$$\overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A}} + \overline{\overline{B}}$$

$$= A+B.$$

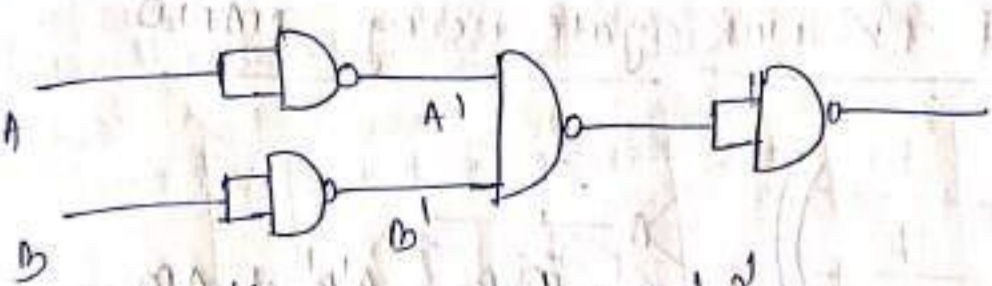
$$[A'B']' = (A')' + (B')' = A+B$$

Realization of NOR gate using NAND:-





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



$$\left[A'B'\right]' = (A')' + (B')'$$

$$= A + B$$

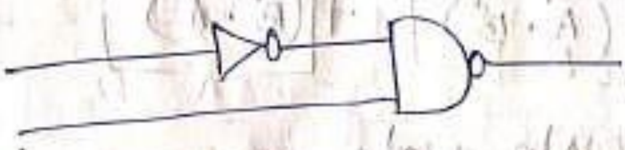
$$y = (A+B)'$$

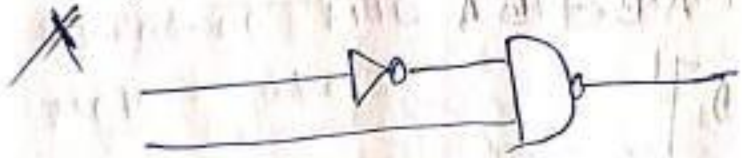
$$= \overline{A+B}$$

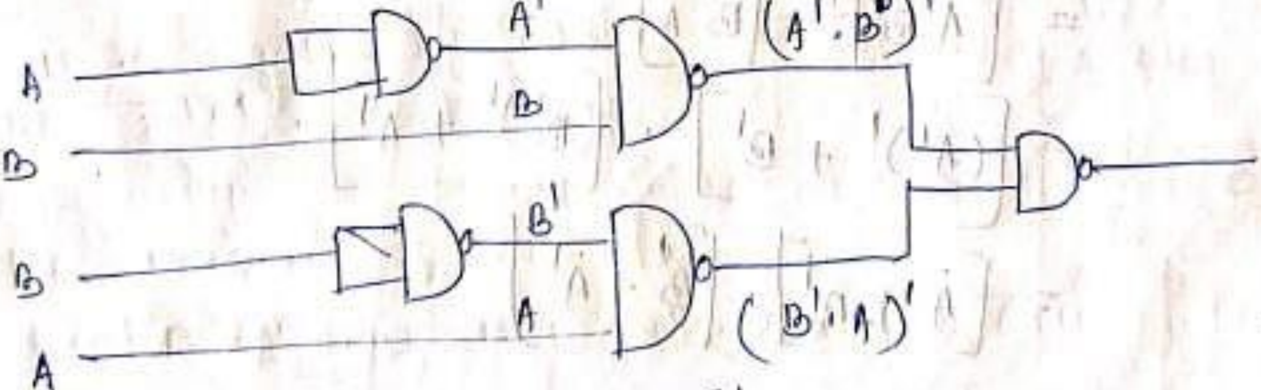
Realization of EX-OR gate using NAND

$A \oplus B = A'B + AB'$

$\Rightarrow$





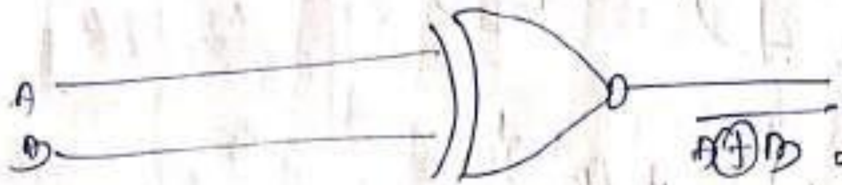


$$\left[(A'\cdot B)'\cdot(B'\cdot A)'\right]'$$

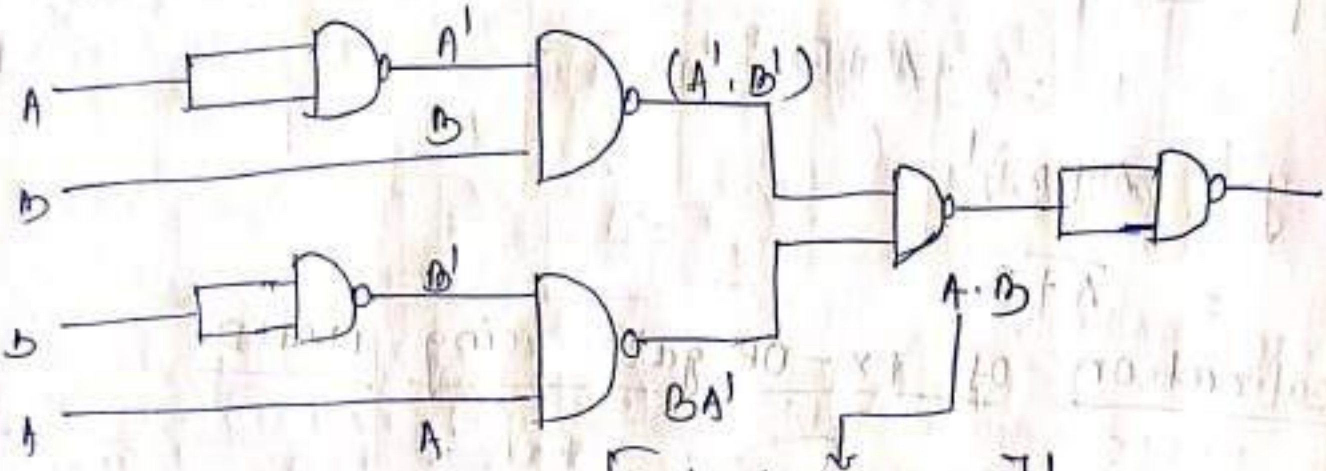
$$= \left[(A'\cdot B)'\right]' + \left[(B'\cdot A)'\right]'$$

$$= A'B + B'A$$

Realization of EX-NOR gate using NAND



$\overline{A \oplus B} = A'B' + AB$



$$\left[(A' \cdot B')' \cdot (B' \cdot A)'\right]'$$

$$= \left[(A' \cdot B)'\right]' + \left[(B' \cdot A)'\right]'$$

$$= A'B + B'A$$

$$y = \left[A'B + B'A\right]'$$

$$= \left[A'B\right]' \cdot \left[B'A\right]'$$

$$= \left[(A')' + B'\right] \cdot \left[(B')' + A'\right]$$

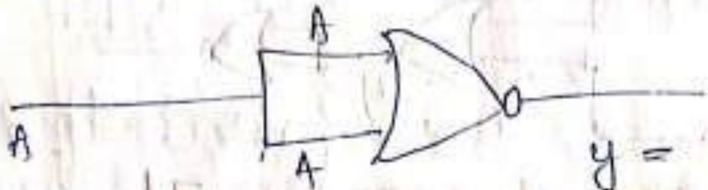
$$= \left[A + B'\right] \cdot \left[B + A'\right]$$

$$= A \cdot B + A \cdot A' + B' \cdot B + A' \cdot B'$$

$$= AB + A'B'$$

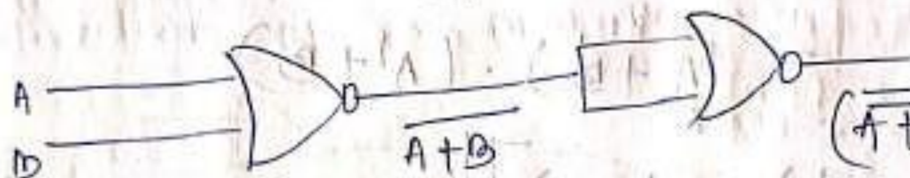
Realization of basic gate using NOR gate :-

NOT gate using NOR gate :-



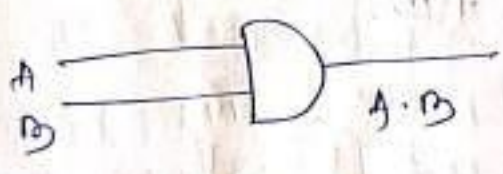
$y = \overline{A+A} = \overline{A}$

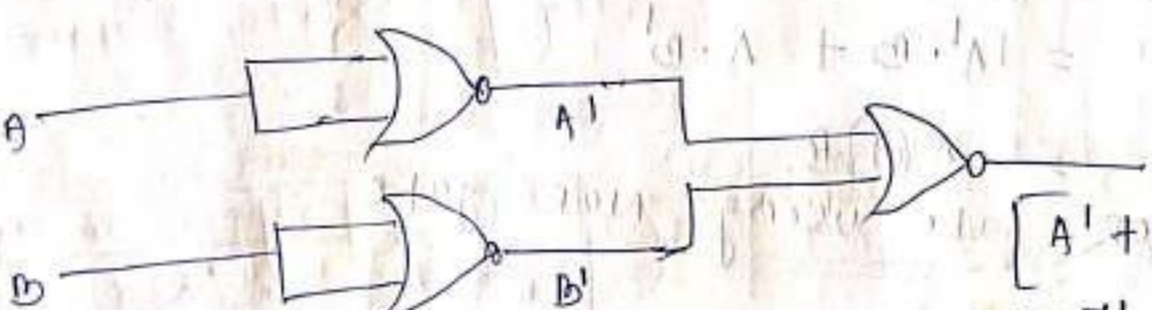
OR gate using NOR gate :-



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

AND gate using NOR gate :-





$[A'+B']'$

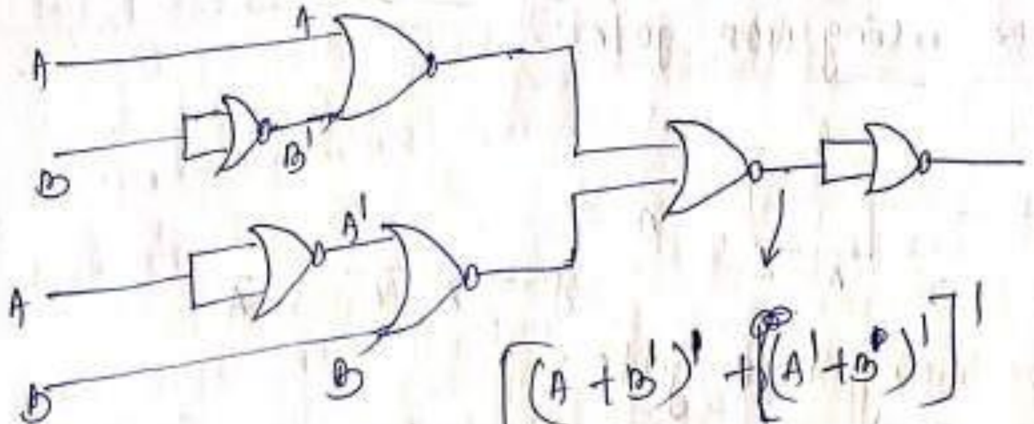
$= [A']' \cdot [B']'$

$= A \cdot B$

EX-OR gate using NOR gate :-

$A \oplus B$

$= A'B + AB'$



$$\left[(A+B')' + (A'+B)'\right]'$$

$$= \left[(A+B')'\right]' \cdot \left[(A'+B)'\right]'$$

$$= (A+B')\cdot(A'+B)$$

$$Y = \left[(A+B')\cdot(A'+B)\right]'$$

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

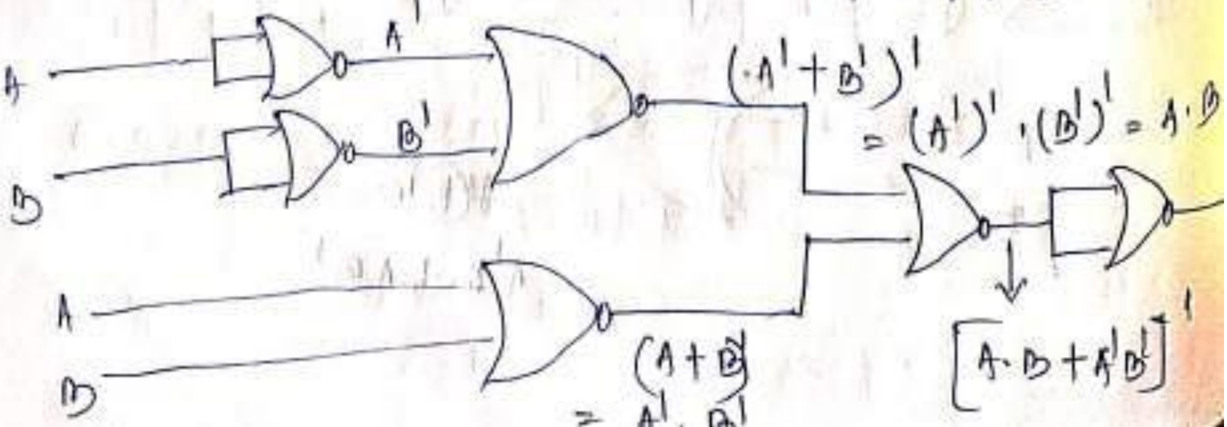
$$= (A+B')'$$

$$= A'\cdot B + A\cdot B'$$

$$= A \oplus B$$

EX-NOR gate using NOR gate

$$A \odot B$$
$$= A\cdot B + A'\cdot B'$$



$$(A'+B')' = (A')'\cdot(B')' = A\cdot B$$

$$(A+B)' = A'\cdot B'$$

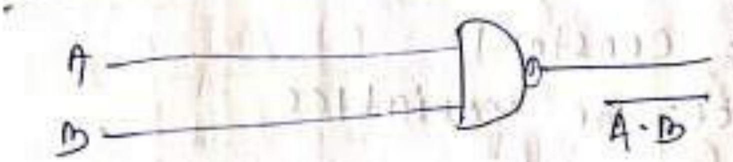
$$\left[A\cdot B + A'B'\right]'$$

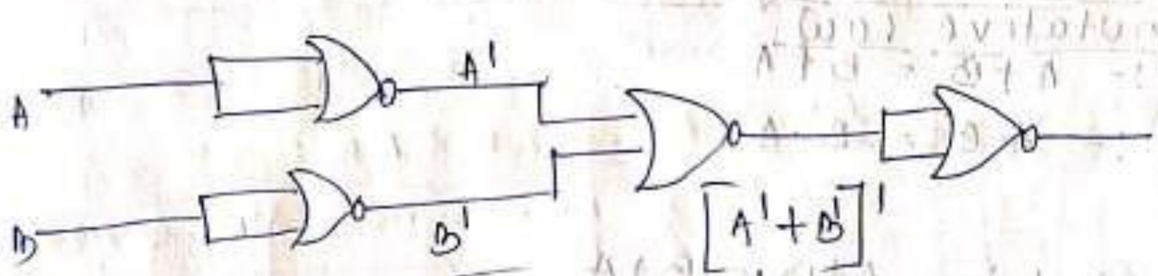
$$y = \left[\left[A\cdot B + A'B'\right]'\right]'$$

$$= A\cdot B + A'B'$$

$$= A \odot B$$

NAND gate using NOR gate





$$y = \left[\left[A' + B'\right]'\right]'$$

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

$$= (A\cdot B)'$$

1.9 Different Postulates and De-morgan's Theorem in Boolean Algebra

Boolean Algebra

The Boolean Algebra are used to represent the digital signal in terms of Algebra.

→ The Boolean Algebra consists of three parts

(i) constant
(ii) variable
(iii) function

Constant

In Boolean Algebra the constants are always same. ~~i.e it~~ The constant in Boolean Algebra may be 0 or 1.

Variable :-

In Boolean Algebra the variables are changes that means the output of the boolean algebra may contains different variables.

Function :-

In Boolean Algebra the functions contains more than one number of variables and constant.

Ex:- $F(A, B, C) = 1 + AB + BC$

Here 1 is the constant

A, B, C are the variables

Laws of Boolean Algebra

① Commutative law

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

Law 2 :- $A \cdot B = B \cdot A$

Proof

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

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

Law-2 :- $A \cdot B = B \cdot A$

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

B	A	B · A
0	0	0
0	1	0
1	0	0
1	1	1

$A \cdot B \cdot C = B \cdot C \cdot A = C \cdot A \cdot B = B \cdot A \cdot C$

(2) Associative laws :-

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

A	B	C	A+B	(A+B)+C
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

A	B	C	B+C	A+(B+C)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

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

A	B	C	AB	(A·B)C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

A	B	C	B·C	A(B·C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

If 4 no. of variables are present then

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

(3) Distributive Laws :-

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

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

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

A	B	C	B+C	AB	AC	AB+AC
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	1

A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Law :- 2

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

R.H.S $= (A+B)(A+C)$

$= AA + AC + AB + BC$

$= A + AC + AB + BC$

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

$= A(1 + B) + BC$

$= A \cdot 1 + BC = A + BC$

$\therefore$ L.H.S = R.H.S (Proved)

(4) AND Laws :-

$A \cdot 0 = 0$

$A \cdot 1 = A$

$A \cdot A = A$

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

(5) OR Laws

$A + 0 = A$

$A + 1 = 1$

$A + A = A$

$A + \overline{A} = 1$

(6) NOT operation (Inversion law) :-

$\overline{\overline{A}} = A$

(7) Absorption Law

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

$A + A \cdot B$

$= A(1 + B)$

$= A \cdot 1$

$= A$

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

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

$A(A + B)$

$= A \cdot A + A \cdot B$

$= A + AB$

$= A(1 + B)$

$= A \cdot 1$

$= A$

A	B	A+B	A(A+B)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

(8) Identity Law:-

$$A + A = A$$
$$A \cdot A = A$$

(9) Demorgan's Law:-

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

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

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

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

A	B	A·B	$\overline{A \cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

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

1.10 Use of Boolean Algebra for simplification of Logic Expression.

Logic Expressions are expressed by the help of Boolean Expression.

→ The expression may contains constants (0&1), variables & function.

→ Usually the Boolean expression are expressed in two ways. These are

(I) Sum of product (SOP)

(II) Product of sum (POS)

(I) <u>sum of product (SOP)</u>

It is an expression in which the product term are sum together

Ex:- $AB + A\bar{B}C + BC$

Here $A \cdot B$, $A \cdot \bar{B} \cdot C$, $B \cdot C$ are the product term. Then the product terms are solve together by summed ~~them~~.

(II) <u>product of sums:- (POS)</u>

It is an Expression in which the sum terms are product together.

Ex:- $F = (A + B + C) \cdot (A + B' + C)$

Here $A + B + C$ & $A + B' + C$ are the summing term, then the summing terms are solve together by producted ~~them~~.

<u>Min term:-</u>

Each individual term in the standard SOP form is called Minterm.

Eg:- $F = \underbrace{ABC}_{} + \underbrace{A\bar{B}\bar{C}}_{} + \underbrace{\bar{A}BC}_{}$ → Minterms

→ It is represented by "m_i"

<u>Max term:-</u>

Each individual term in the standard POS form is called Maxterm.

Eg:- $f = \underbrace{(A+B)}_{}\ \underbrace{(A+\bar{B})}_{}$ → Max terms

→ It is represented by "M_i"

A	B	Minterms m_i	Maxterms M_i
0	0	$\bar{A}\bar{B} \to m_0$	$A+B \to M_0$
0	1	$\bar{A}B \to m_1$	$\bar{A}+\bar{B} \to M_1$
1	0	$A\bar{B} \to m_2$	$\bar{A}+B \to M_2$
1	1	$AB \to m_3$	$\bar{A}+\bar{B} \to M_3$

SOP $\to A = 1,\ \bar{A} = 0 \to$ Minterm $\to$ product term

POS $\to A = 0,\ \bar{A} = 1 \to$ Maxterm $\to$ sum term

standard SOP

~~when the function is a three variable function~~

A	B	C	Minterms	Maxterms
0	0	0	$\bar{A}\bar{B}\bar{C}$ m_0	$A+B+C,\ M_0$
0	0	1	$\bar{A}\bar{B}C$	$A+B+\bar{C},\ M_1$
0	1	0	$\bar{A}B\bar{C}$	$A+\bar{B}+C,\ M_2$
0	1	1	$\bar{A}BC$	$A+\bar{B}+\bar{C},\ M_3$
1	0	0	$A\bar{B}\bar{C}$	$\bar{A}+B+C,\ M_4$
1	0	1	$\bar{A}\bar{B}C$	$\bar{A}+B+\bar{C},\ M_5$
1	1	0	$AB\bar{C}$	$\bar{A}+\bar{B}+C,\ M_6$
1	1	1	ABC	$\bar{A}+\bar{B}+\bar{C},\ M_7$

* $Y = \sum m(3, 6, 7)$

$= m_3 + m_6 + m_7$

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

* $Y = \pi(4, 5, 7)$

$= M_4 \cdot M_5 \cdot M_7$

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

Canonical form / standard form :-

Boolean expression where each term contains all Boolean variables in their true or complemented form.

+ These product terms are nothing but the minterms.

+ sum of all minterms of "f" for which "f" assumes 1 is called canonical SOP.

$$Ex:- f = \bar{x}\bar{y}z + \bar{x}yz + x\bar{y}z + xyz$$

$$f = \Sigma m(1,3,5,7)$$

$$= \Sigma(m_1 + m_3 + m_5 + m_7)$$

Conversion of SOP into canonical SOP form :-

Step-1

Determine the maximum variable.

Step-2

Multiply "1" where term is missing.

Step-3

Simplify the boolean expression using Boolean Theorm.

* Convert $AB + A'C$ into canonical form ?

Ans:- (i) A, B, C

(ii) $A \cdot B \cdot 1 + A' \cdot 1 \cdot C$

$$F = A \cdot B(C + \bar{C}) + A' \cdot (B + \bar{B}) \cdot C \quad \left(\because \left.\begin{matrix} C + \bar{C} \\ B + \bar{B} \end{matrix}\right\} = 1\right)$$

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

* Convert $AB + C$ into canonical form ?

Ans:- (i) A, B, C

(ii) $AB \cdot 1 + 1 \cdot 1 \cdot C$

(iii) $F(A, B, C) = AB \cdot 1 + 1 \cdot 1 \cdot C$

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

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

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

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

Conversion of POS into Canonical POS form :-

Step-1

first write the boolean expression and determine the maximum number of variables.

Step-2

Add 0 where the variables terms are missing.

Step-3

Simplify the boolean expression using Boolean Theorm.

* Convert the $F = (A'+B')(B'+C)(A+C)$ Boolean expression into canonical POS form.

Ans:- $F = (A'+B')(B'+C)(A+C)$

A, B, C

$$F = (A'+B'+0)\,(0+B'+C)\,(A+0+C)$$

$$= (A'+B'+C \cdot C')\,(A \cdot A'+B'+C)\,(A+B \cdot B'+C)$$

$$= (A'+B'+C)\,(A'+B'+C')\,(A+B'+C)\,(A'+B'+C)$$

$$(A+B+C)\,(A+B'+C)$$

Conversion of SOP into the canonical POS :

Step-1

first write the boolean expression and determine the Maximum variable.

Step-2

Take the complement of given expression and ~~step~~ simplify.

Step-3

Take once again complement.

* convert $F = AB' + BA'$ into canonical POS form

Ans:- $F = AB' + BA'$

$F' = [AB' + BA']'$

$= (AB')' \cdot (BA')'$

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

$(F')' = F = [(A' + B) \cdot (B' + A)]'$

$= [A'(B' + A) + B(B' + A)]'$

$= [A'B' + \cancel{A'A} + \cancel{B'B} + AB]'$

$= [A'B' + AB]'$

$= [A'B']' \cdot [AB]'$

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

~~* conversion of pos into sop~~

K-Map (Karnaugh Map)

→ To simplifying the boolean expressions K-map method is used.

→ K-map is the graphical representation of boolean expression.

→ for a boolean expression consisting of n-variables number of cell required in K-map $= 2^n$ cell.

Rules for K-Map

~~(i) No zeros allowed~~.

(I) groups can be vertical or Horizontal but can't be diagonals.

(II) overlapping allowed.

(III) Group should be as large as possible.