Digital Logical Disign

Desimal number :-

Decimal numbes system is a numbes system that we use in our daily life. It includes the numbes from 0 to 9 and the base of decimal numbes is 10.

Example :

$(128)_{10}$ or $(256.68)_{10}$

Decimal value to sum of each value :

$(128)_{10}$

$= 1 \times 10^2 + 2 \times 10^1 + 8 \times 10^0$

$= 100 \quad 20 \quad 8$

$(256.68)_{10}$

$= 2 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 + 6 \times 10^{-1} + 8 \times 10^{-2}$

$= 200 + 50 + 6 + 0.6 + 0.08$

Binary Numbes :-

Binary numbes system is a numbes system to represents values. It uses only two ~~numbess~~ bits 0 and 1 to represent values. Binary numbes has base 2.

Example : The numbes 4 represent in 4 bit binary code

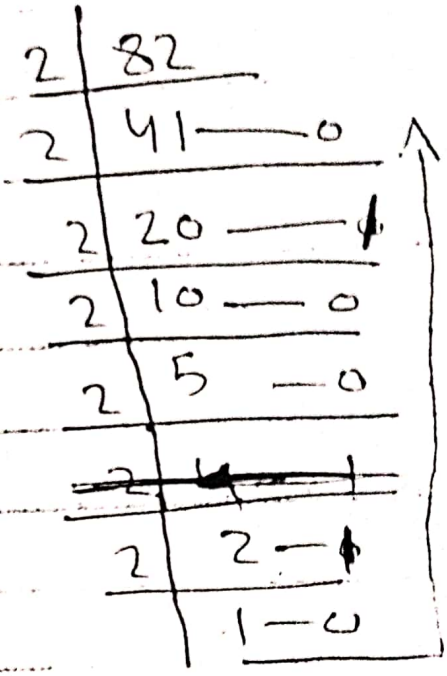
$(4)_{10} = (0101)_2 \quad (0100)_2$

2 | 4
2 | 2 - 0
1 - 0

Decimal to binary Conversion :-

$(82)_{10} = (?)_2$

$(82)_{10} = (1010010)_2$



by LCM Method

$(25)_{10} = (?)_2$

$25 = 16 + 8 + 1$

$= 2^4 + 2^3 + 2^0$

$= (11001)_2$

Sum of weights method

$(0.625)_{10} = (.?)_2$

$= 0.5 + 0.125$

$= 2^{-1} + 2^{-3}$

$= 0.101$ - by sum of weight

$(45.5)_{10} = (?)_2$

$45 = (101101)_2$

$0.5 = (1.00)_2$

$45.5 = 101101.1$

2	45
2	22 – 1
2	11 – 0
2	5 – 1
2	2 – 1
	1 – 0

$0.5 \times 2 = 1.00 \quad 1$

$(0.3125)_{10} = (?)_2$

$(.0101)_2$

0.3125×2	$= 0.625$	0
0.625×2	$= 1.25$	1
0.25×2	$= 0.50$	0
0.5×2	$= 1.00$	1

Binary Arthmatics :-

1. Binary addition :-

Some basic rules to add binary digits

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 10 \longrightarrow$ with Carry 1

Example:

111 + 11

```
  ①①
  111
+  11
 1010
```

110 + 100

```
  110
+ 100
 1010
```

Binary Subtraction:-

Same basic rules to subtract two binary digits

0 - 0 = 0

1 - 0 = 1

borrow 1 ⟵ 10 - 1 = 1

1 - 1 = 0

Example:

(1) 11 - 10

```
  11
- 10
  01
```

(2) 101 - 011

```
  ⓪
  101
- 011
  010
```

Binary Multiplication :-

Some basic rules to multiply binary digits.

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

(i) 11×11

```
  11
  11
 ---
  11
 11X
1001
```

(ii) 101×111

```
  101
  111
 ----
  101
 101X
1010XX
------
100011
```

Binary Division :-

(1) $110 \div 10$

```
      11
   ------
10 | 110
     10
   ----
     10
     10
   ----
      0
```

(2) $1100 \div 100$

```
      11
   ------
100 | 1100
      100
   -----
      100
      100
   -----
        0
```

(3) $110 \div 11$

```
      10
   ------
11 | 110
     110
   -----
       0
```

Complements

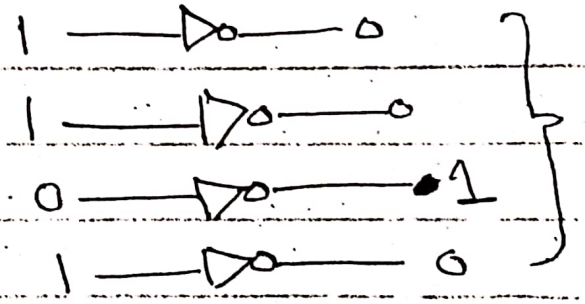
(1) 1st Compliment :-

The fist Complement of binary numbe is founded by Converting all 1 to 0 and all 0 to 1.

The simplest way to get 1st Complement of binary numbes with digital ciscuit is to use parallel inverstors (NoT ciscuit)

Example:

binary number = 1101



1st Complement = 0010

(2) 2nd Complement :-

The 2nd Complement is founded by Just adding the 1 to Leaßt significant bit (LSB) of 1st Complement.

Example:

binary numbes = 10110010

1st Complement = 01001101

01001101

\+ 1

01001110

~~01001101~~

\+ 1111

2nd Complement = 01001110

Signed Numbers :-

A numbers that Consist both sign and magnitude information is called Sign numbers. The sign indicate whether numbers is positive or nagitive and the magnitude indicate the value of its numbers.

Digital system such as Computes must be able to handle both positive and nagitive number. There are three forms in which sign integers numbers can be represent in binary are given bellow

(1) Sign-magnitude

(2) 1st - Complement

(3) 2nd Complement

Sign Bit :-

The left most bit in sign numbers is called sign bit.
A 0 sign bit indicates a positive number and 1 sign bit indicates a negitive number.

1) Sign magnitude Form :-

when a binary numbers is represented in sign magnitude form the left most bit is sign bit and othe bits are represented as magnitude bits.
In sign magnitude form a negitive numbers has same magnitude bit as the ~~positive~~ corosponding positive numbers but the sign bit is a 1 scithes 0.

Example :-

25 in 8 bit binary = 00011001

magnitude bit

sign bit

2	25
2	12 - 1
2	6 - 0
2	3 - 0
	1 - 1

-25 in 8 bit binary = 10011001

-sign bit

2) 1st Complement form :-

Positive numbers in 1st Complement form are represented same way as positive sign-magnitude numbers. and a negative number is the 1st Complement of the corresponding positive numbers

+25 in 8 bit 1st Complement form

00011001

And

-25 in 8 bit 1 Complement form

11100110

3) 2nd Complement form :-

Positive numbe in 2nd Complement form are represent as same way in sign-magnitude or 1st Complement form and a negitive number is the 2nd Complement of the corresponding positive number.

+25 in 3bit 2nd Complement form

00011001

-25 in 8 bit 2nd Complement form

11100111

11100110	
+	1
11100111	

Decimal values of Signed Numbers

Sign magnitude :

$10010101 = ?$

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

1 0 0 1 0 1 0 1

magnitude bit

(Show −) Sign bit

$= \pm 16 + 4 + 1$

$= 21$

Decimal value $= -21$

1st Complement :

00010111

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

0 0 0 1 0 1 1 1

$16 + 4 + 2 + 1 = +23$ (Decimal)

11101000

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

1 1 1 0 1 0 0 0

$-128 + 64 + 32 + 8 = -24$

Decimal $= -24 + 1$ (adding 1)

Decimal $= -23$

2nd Complement :

01010110

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

0 1 0 1 0 1 1 0

$64 + 16 + 4 + 2 = +86$

10101010

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

1 0 1 0 1 0 1 0

$-128 + 32 + 8 + 2 = -86$

Asthmatic Operation with Sign Numbers

Addition :

(1) Both positive

00000111	7
+ 00000100	+ 4
00001011	11

(2) Positive numbers larger than Negitive

00000111	15
+ 11111010	+ -6
00001001	9

Discard $\rightarrow$ 1 0 0 0 0 1 0 0 1

③ Negitive number Integer positive

```
  0 0 0 1 0 0 0 0      16
+ 1 1 1 0 1 0 0 0    + 24
-----------------    ----
  1 1 1 1 1 0 0 0      -8
```

④ Both negitive

```
                1 1 1 1 1 1 1
  1 1 1 1 1 0 1 1      -5
+ 1 1 1 1 1 0 1 1    + -9
-----------------    ----
1 1 1 1 1 0 0 1 0      -14
```

Discard → 1

Subtraction :

① 8 − 3 = 8 + (−3) = 5

```
  0 0 0 0 1 0 0 0  → 8
+ 1 1 1 1 1 1 0 1  → 2nd Complement (-3)
-----------------
1 0 0 0 0 1 0 1
```

Discard Carry ←

② 12 − (−9) = 12 + 9 = 21

```
  0 0 0 0 1 1 0 0  → 12
+ 0 0 0 0 1 0 0 1  → (2nd Comp - +9
-----------------
  0 0 0 1 0 1 0 1
```

Hexadecimal Number :-

Hexadecimal number system is a 16 numbers system from 0 to 9 and A to F. The base of Hexadecimal number is 16. It has 4 bit binary table.

TABLE

Decimal	Binary	Hexadecimal
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
10	1 0 1 0	A
11	1 0 1 1	B
12	1 1 0 0	C
13	1 1 0 1	D
14	1 1 1 0	E
15	1 1 1 1	F

Binary to Hexadecimal

(a) $(1100101001011)_2$

Make Combinations of 4-bits from Right

1100 1010 0101 0111

C A 5 7 using table

Hexadecimal to Binary

(a) $10A4_{16}$

1 → 0001
0 → 0000
A → 1010
4 → 0100

Using table

Hexadecimal to Decimal

(a) $(A85)_{16}$

$= A \times 16^2 + 8 \times 16^1 + 5 \times 16^0$

$= 2560 + 128 + 5$

$= 2693$

Decimal to Hexadecimal

(a) $(650)_{10}$

$28A_{16}$

16	650
16	40 — 10-A
16	2 — 8
	0 — 2

Octal Numbers :-

Octal Numbers System is 8 digit numbers System from 0 to 7. The base of Octal numbers system is 8, Octal numbers system ~~bad~~ serves 3 bit binary code.

Octal numbers 0, 1, 2, 3, 4, 5, 6, 7.

TABLE

Octal	Binary
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

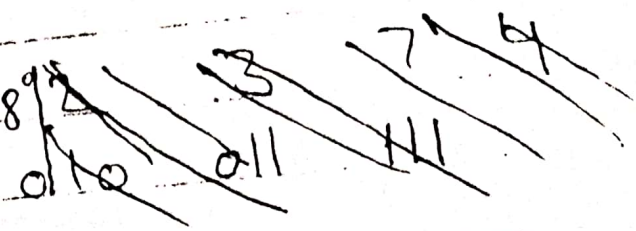
Octal to decimal

$(2374)_8$

$= 2\times8^3 + 3\times8^2 + 7\times8^1 + 4\times8^0$

$= 1024 + 192 + 56 + 4$

$= (1276)_{10}$



Decimal to Octal

$(359)_{10}$

8	359
8	44 — 7
8	5 — 4
	0 — 5

$(547)_8$

Binary to Octal

(a) 110101

110 101

6 5 $= (65)_8$

Using table

Octal to binary

$(140)_8$

1 4 0

↓ ↓ ↓

001 100 000

$(001100000)_2$

Binary Coded Decimal :-

Binary Coded Decimal is a way to express each of the decimal digit with a binary Code. These are only ten code group in the BCD system from 0 to 9. It uses 4-bits to express each decimal to binary.

(Decimal)	(BCD)
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Invalid Codes :-

4-bits binary representation has sixteen numbers but in BCD we uses only fist ten numbers and the codes of other six numbers are called Invalid Codes.

Example :- Decimal (15) → 1111 (invalid Code)

BCD Addition :-

(a) 0011 + 0100

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

3 + 4 = 7

(b) 1001 + 0100

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

9 + 4 = 13

invalid BCD Add 6 BCD

1 3

13

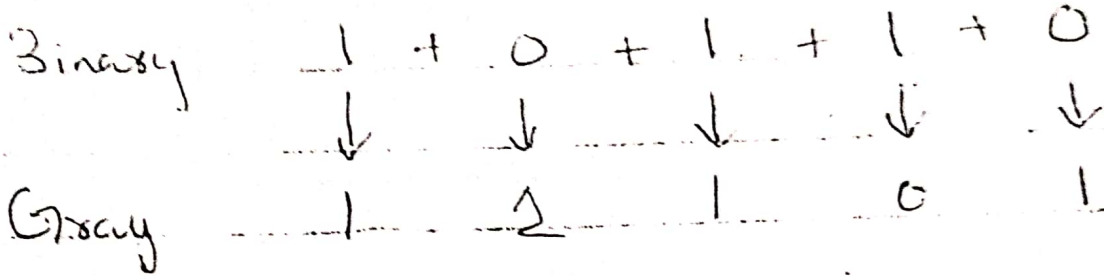
Digital Codes :-

There are many codes used in digital system are given bellow

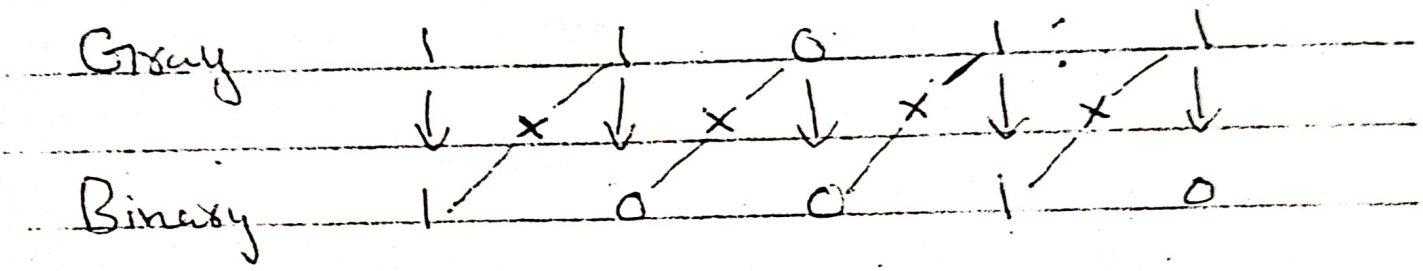
1) Gray Code :-

Gray Code is unweighted and not an arithmatic code. It is also called RBC (Reflected binary code). It has only single bit change from one code word to next in sequence.

Binary to Gray Code



Gray Code to binary



(2) Alphanumaxic Code :-

As the name of code represent it includes alphabets and numaxic values. It uses a 6-bit code and represent 64 characters.

(3) ASCII :-

ASCII stand for "American Standard Code for Information Interchan
-ASCI has 7-bit code and represe 128 characters includes alphabets, numaxic values and some special characters.

(4) Unicode :-

Unicode stand for "Universal code". It Consist of characters of almost of all languages in the word. Unicode is mostly uses now days in the world.

Parity Method of Error Detection

Any group of bits Contain even or odd numbers of 1s. A Parity bit is attached to the group of bits to tell the system the number of 1 in group is even or odd

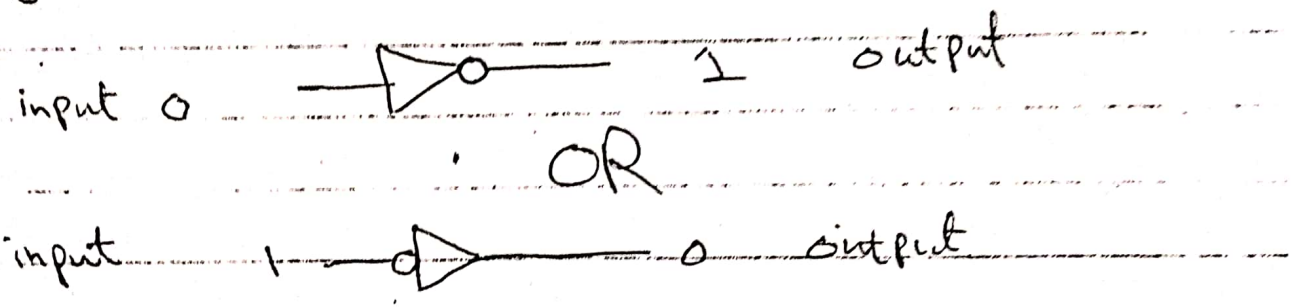
Parity bit	BCD
0	0000
1	0001
1	0010
0	0011
1	0100
0	0101
0	0110
1	0111

DLD Outline No 2

The Inverter:-

The inverter or not circuit perform the operation called inversion or Complement. The inverter Convert all logical levels to oposit levels. In term of bits it changes 0 to 1 and 1 to 0.

Symbol of inverter



Truth table of inverter

input	output
0	1
1	0

input	output
Low	High
High	Low

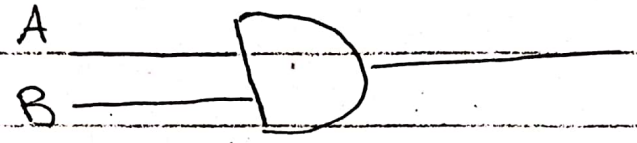
Logical Expression for inverter

$$X = \bar{A}$$

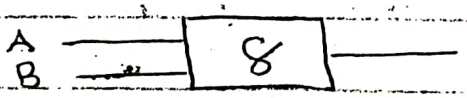
AND Gate :-

AND is a Logical gate that preform the operation called multiplication between given inputs. AND Gate generate output 1 if and only if all the inputs are 1 otherwise it generate 0 output.

Symbol of And



OR



Truth table of AND

input		output
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

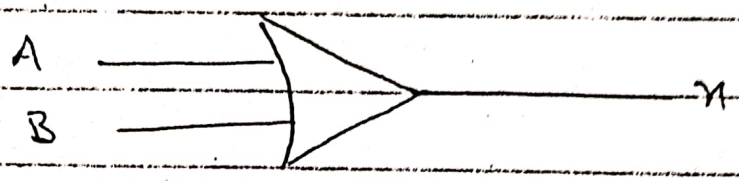
Logical Expression

$X = AB$

OR Gate :-

OR is a logical gate that perform the operation called logical addition between two or more inputs. It generates output 0 if and only if all inputs are 0 otherwise it generates 1 - output

Symbol :



Truth table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1
input		output

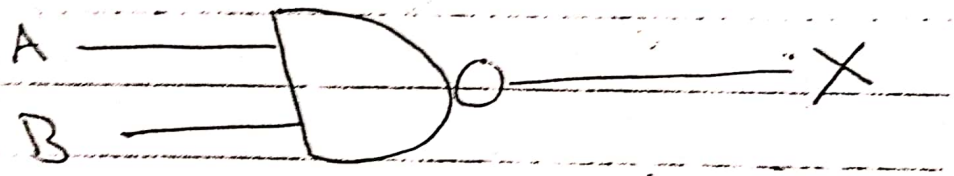
Logical Expression :-

$$X = A + B$$

NAND Gate :-

NAND Gate is the Combination of two Gates AND and NOT. The negitive AND gate is also called NAND gate. A from this gate we get inveated outputs of AND gate. It genrates 0 when both ~~output~~ inputs are 1 otherwise it genrates 1 output.

Symbol



Truth Table

input		output
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

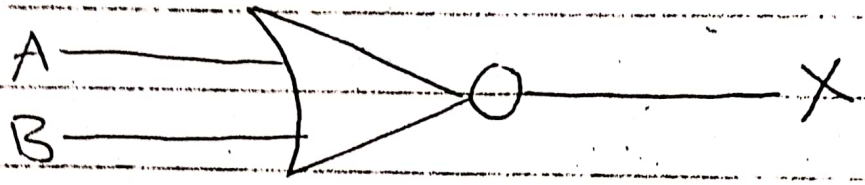
Logical Expression

$$X = \overline{AB}$$

NOR Gate :

NOR gate is a Combination of OR and NOT gate. It generates the inverted output of OR gate. It generates 1 when all inputs are 0 otherwise it generate 0.

Symbol



Truth table

input		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

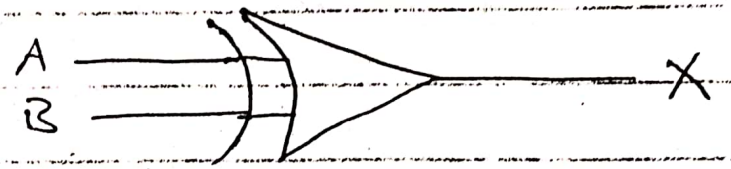
Logical Expression

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

Enclusive-OR Gate :-

Enclusiv-OR gate also written as XOR gate. It generates 0 output when both inputs are same and generate 1 when both are diffsents. It only operate on two inputs.

Symbol



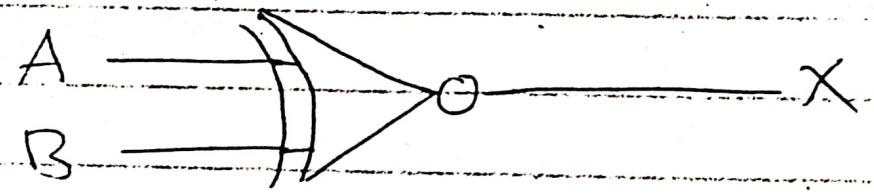
Truth Table

input		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Enclusive-NOR Gate :-

Enclusive-NOR gate also written as XNOR gate. It genrates the complemented output of enclusive-OR gate. It genrates 1 on both same inputs and 0 on both diffrent inputs

Symbol



Truth Table

input		output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

DLD Outlin- no 3

Boolean operation and Expression

Boolean algebra is the methmatics of digital logic.

Boolean Addition :-

Boolean addition is a OR operation between given inputs.

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10 ⟶ with 1 carry

Boolean Multiplication :-

Boolean multiplicatio is equivalent to AND operation.

0 · 0 = 0

0 · 1 = 0

1 · 0 = 0

1 · 1 = 1

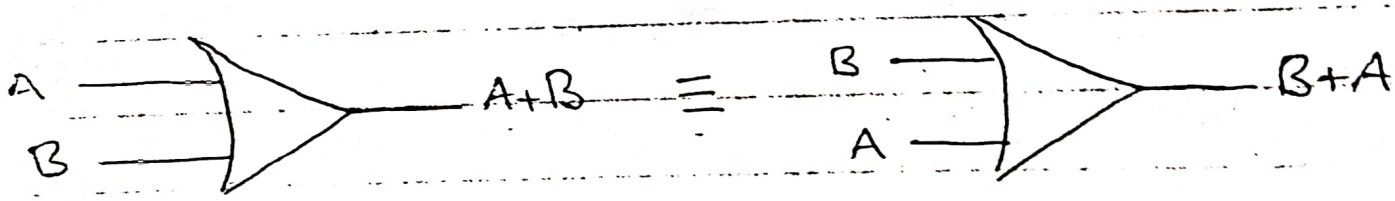
Laws of Boolean Algebra

Different laws of Boolean algebra are given bellow

Commutative Laws :-

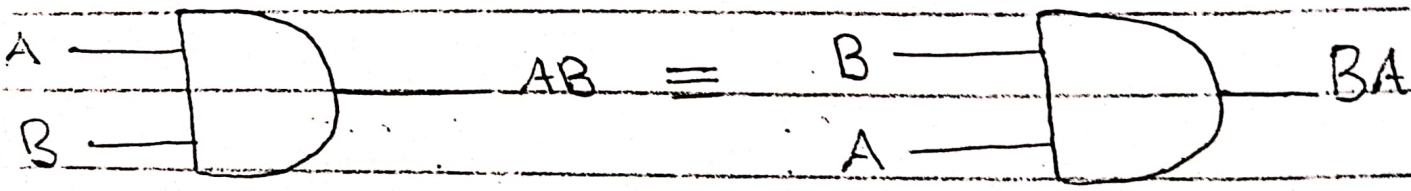
(1) The Commutative law of addition is written as

$$A + B = B + A$$



(2) The Commutative law of multiplication is written as

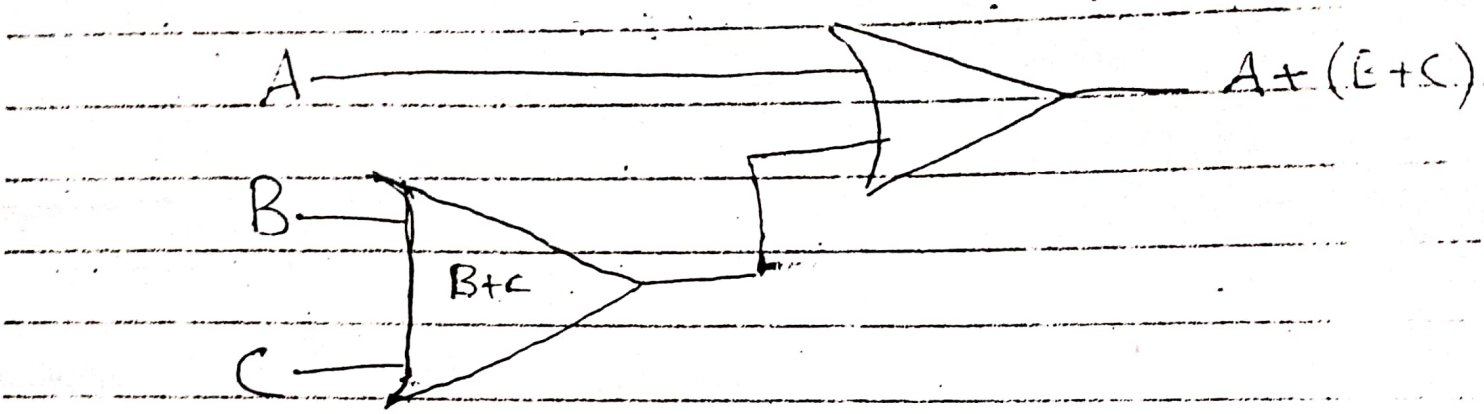
$$AB = BA$$



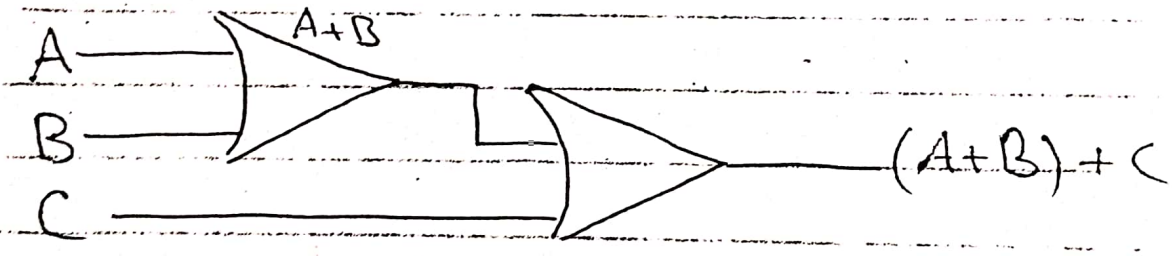
Associative Laws :-

(1) Associative Law of addition

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

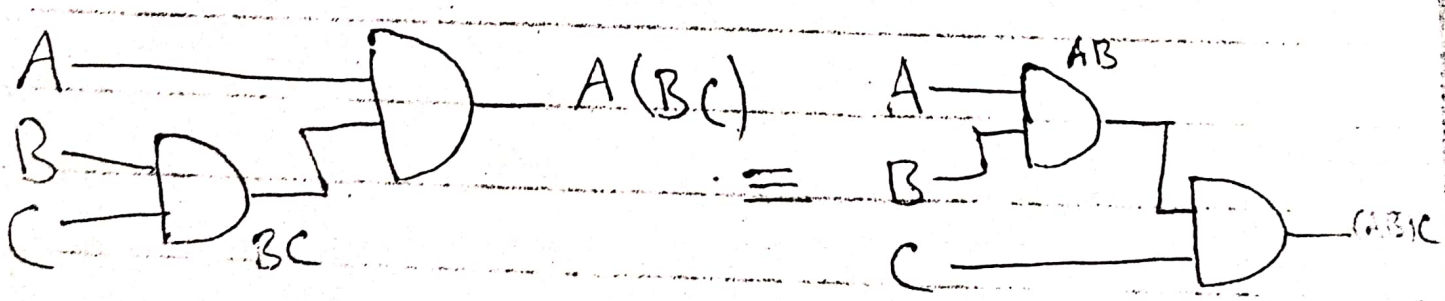


≡



(2) Associative law of multiplication

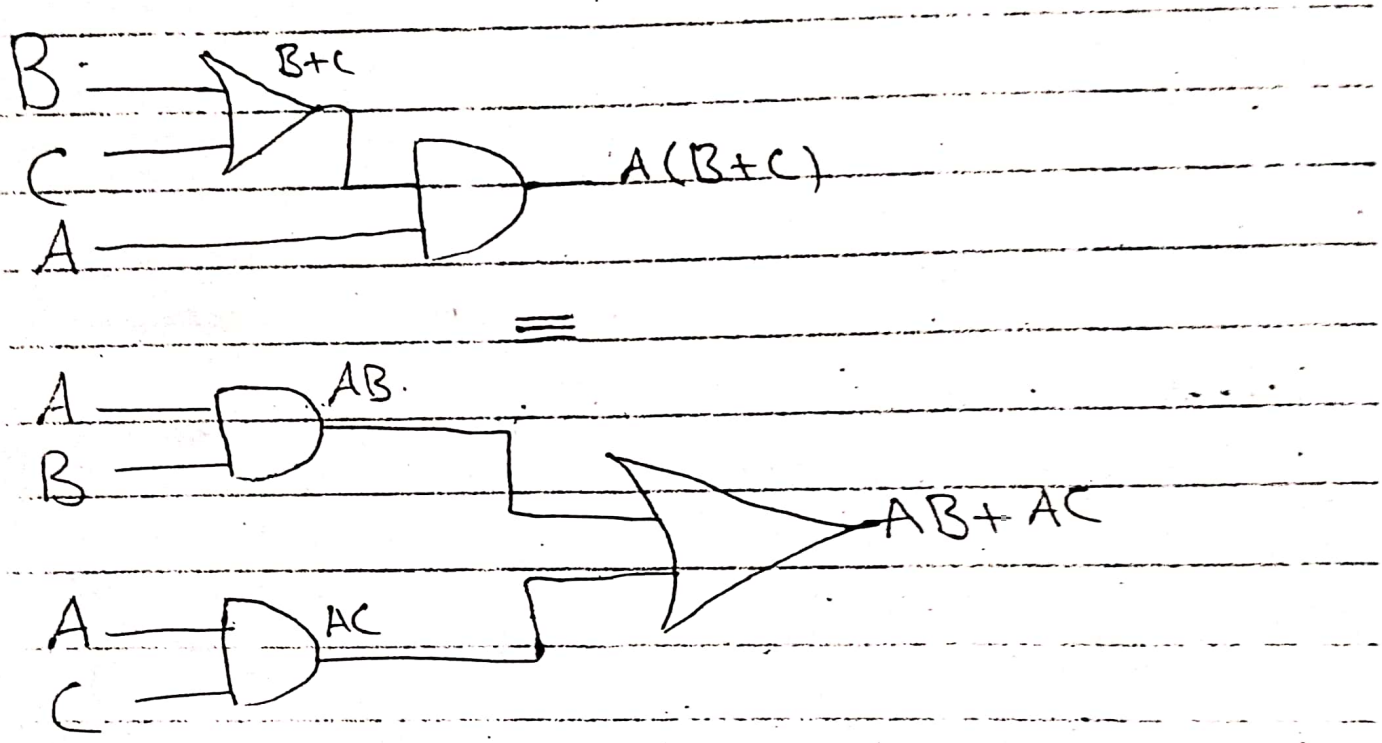
$$A(BC) = (AB)C$$



Distributive law :-

This law implement on three variables

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



Rules of Boolean Algebra

(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 Rule 10 $A+AB = A$

L.H.S

$= A + AB$

$= A \cdot 1 + AB$

use distributive law $AB+AC = A(B+C)$

$= A(1+B)$

use the Rule $A+1 = 1$

$= A(1)$

use Rule $A \cdot 1 = A$

$= A$

Proof Rule 11 $A+\bar{A}B = A+B$

L.H.S

$= A + \bar{A}B$

$= (A+AB) + \bar{A}B$ $\quad \therefore A+AB = A$

$= AA + AB + \bar{A}B$ $\quad \therefore A \cdot A = A$

$= AA + AB + A\bar{A} + \bar{A}B$ $\quad \therefore A \cdot \bar{A} = 0$

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

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

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

$= A+B$

Proof Rule 12 $(A+B)(A+C) = A+BC$

L.H.S $(A+B)(A+C) = AA+AC+AB+BC$

$A+AC+AB+BC$ $\quad \therefore A \cdot A = A$

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

$A \cdot 1 + AB + BC$ $\quad \therefore A+1 = 1$

$A(1+B)+BC$ $\quad \therefore A+1 = 1$

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

DeMorgan's Theorem :-

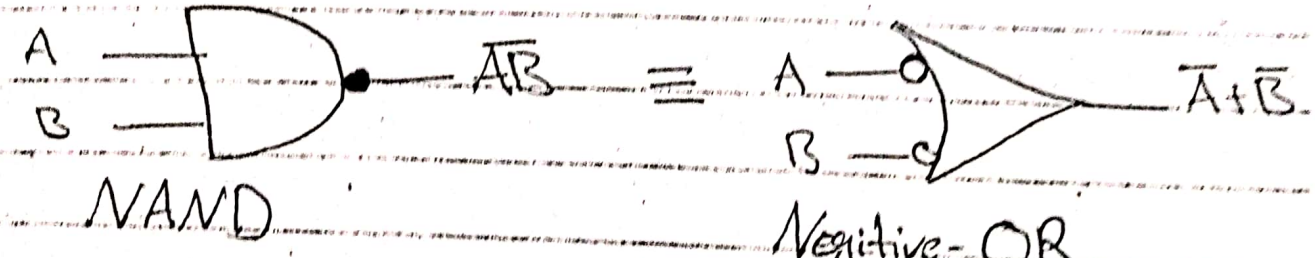
The demorgan's theorem in term of boolean algebra is

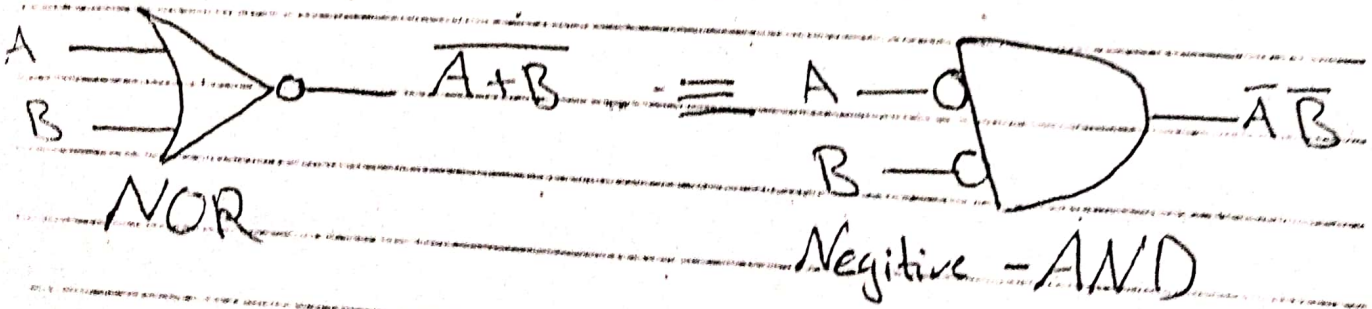
(a) The Complement of two or more ANDed variables is equal to the OR of the Complement of individual variables.

(b) The Complement of tow or more ORed variables is equal to the AND of the Complements of individuals variables.

$$\overline{AB} = \bar{A} + \bar{B} \quad \text{---------- (a)}$$

$$\overline{A+B} = \bar{A}\bar{B} \quad \text{---------- (b)}$$





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

$\overline{AB}$ is same output as $\bar{A}+\bar{B}$

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

$\bar{A}+\bar{B}$ is same output as $\bar{A}\bar{B}$

Demorgan theorem can be implimented on more than 2 variables for example

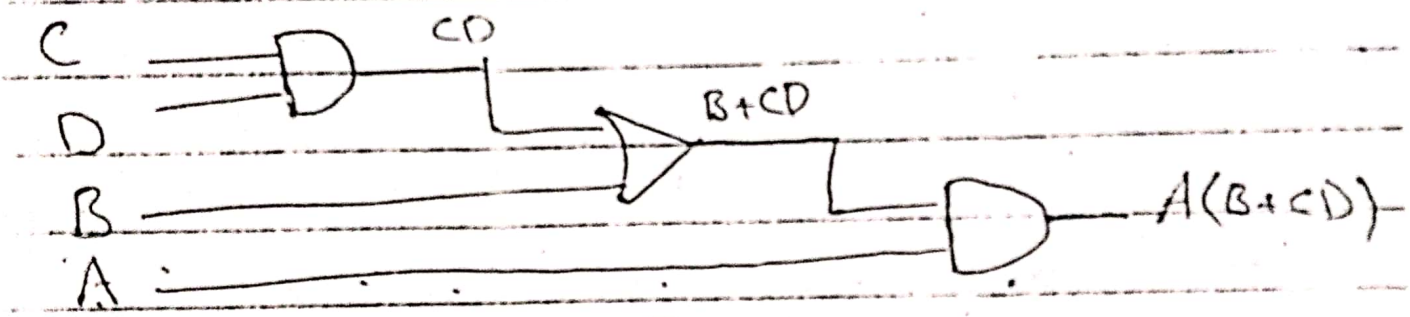
$$\overline{A+B+C} = \bar{A}\bar{B}\bar{C}$$

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

Boolean Expression for a Logic Circuit

In this part of outline we write the Boolean Expression fo logic Circuit and vise versa.

Logic circuit



Logic Expression

$A(B+CD)$

Logic Simplification Using Boolean Algebra

(a) $AB + A(B+C) + B(B+C)$

$= AB + AB + AC + BB + BC$

$=$ using Rule $A \cdot A = A$

$= AB + AB + AC + B + BC$

Using rule $A + A = A$

$= AB + AC + B + BC$

$= AB + AC + B$ $\quad \therefore A + AB = A$

$= B + AC$ $\quad \therefore A + AB = A$

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

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

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

$= A\bar{B}C + A\bar{B}.BCD + \bar{A}\bar{B}C$ $\quad \because A.A = A$

$= A\bar{B}C + ACD(0) + \bar{A}\bar{B}C$ $\quad \because A.\bar{A} = 0$

$= A\bar{B}C + 0 + \bar{A}\bar{B}C$

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

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

$= \bar{B}C$ Ans

Standard forms of boolean Expression

There are two forms of standard boolean Expression are given bellow.

The Sop form:

Sop stand for "sum of products". In this standard form boolean expression are written As

$AB + ABC$ OR

$ABC + C\bar{E}D + \bar{B}C\bar{D}$ OR

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

(a) convert to Sop form

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

$\overline{A+B} + \bar{C} \Rightarrow A+B$

Convert to Sop form

(a) $\overline{\overline{A+B}+C}$

$= (\overline{\overline{A+B}})\,\overline{C}$ ∴ Demorgan Law

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

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

Standard Sop form:

The standard Sop form means each Product form must Contain all variables use in boolean Expression.

$\bar{A}B\bar{C} + A\bar{B}C + \bar{A}BCD \rightarrow$ not standard sop form

Standard sop form.

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

(a) $A\bar{B}C + \bar{A}\bar{B} + AB\bar{C}\bar{D}$ Convert to standard Sop form

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

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

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

PTO

Pos form :-

POS stand for (Product of Sum) we written boolean expression in POS form As

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

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

Standard pos form :-

Standard POS form means each sum form must Contain all vaxiables use in boolean Expression.

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

Converting to Standard POS form

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

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

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

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

$\therefore$ Ans

Boolean Expressions and truth tables

1) Sop to truth table :-

In SOP if these is Complement of vaxiable that indicate that these is 0 in tauth table and if non-Complement vaxiable that indicate that these is '1' in truth table

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

inputs			Output	Product term
a	b	c		
0	0	0	0	
0	0	1	1 m_1	$\bar{A}\bar{B}C$
0	0	0	0	
0	1	1	0	
1	0	0	1 m_4	$A\bar{B}\bar{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1 m_7	ABC

2) POS to truth table :-

In POS if there is Complement of variable that indicates 1 in truth table and non-complement variable indicate 0.

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

Table on Next page

input			output	Sum term
a	b	c	X	
0	0	0	0	$A+B+C$
0	0	1	1	
0	1	0	0	$A+\bar{B}+C$
0	1	1	0	$A+B+\bar{C}$
1	0	0	1	
1	0	1	0	$\bar{A}+B+\bar{C}$
1	1	0	0	$\bar{A}+\bar{B}+C$
1	1	1	1	

Karnaugh Map :-

The karnaugh map is a systematic method for simplifying Boolean expression. karnaugh map is similar to truth table because in karnaugh map we use table to Reesent all possible values of input variables and the resulting output for each value.

formula to Calculate cell

$= 2^n$ → here n is a numbers of variables.