Week No. 01: Number Systems, Operations and Codes

Decimal Number:-

- It has ten digits. 0 - 9
- Each digit in decimal number represents a certain value or quantity
- If a value or quantity is greater than nine, two or more digits then,

For example: - 23

$= 2 \times 10^1 + 3 \times 10^0$

$= 2 \times 10 + 3 \times 1 = 20 + 3 = 23$

Weight:-

- Weight for whole numbers are positive powers of ten that increase from right to left and beginning with $10^0=1$ and weight depend on position of the digit.
- So decimal number system has a base of 10.

  ……$10^4\ 10^3\ 10^2\ 10^1\ 10^0$
- For fractional numbers, weights are negative powers of 10 that decrease from left to right and beginning with $10^{-1} = 0.1$

  …….. $10^3\ 10^2\ 10^1\ 10^0 . 10^{-1}\ 10^{-2}\ 10^{-3}$ ……..

  Decimal Point

Examples: (1) Express 47 as sum of the values of each digit.
(2) Express 568.23 as sum of the values of each digit.

Sol 01: $47 = 4 \times 10^1 + 7 \times 10^0 = 4 \times 10 + 7 \times 1 = 40 + 7 = 47$

Sol 02: $568.23 = (5 \times 10^2 + 6 \times 10^1 + 8 \times 10^0) + (2 \times 10^{-1} + 3 \times 10^{-2})$

$= (500 + 60 + 8) + (0.2 + 0.03) = 568 + 0.23 = 568.23$

Binary numbers:-

- Another way to represent quantities
- Less complicated then decimal system because it has two digits, 0 or 1.
- Weights in binary numbers are based on powers of two.
- It has two digits (bits) 0 or 1.

- It has a base of 2 e.g. …….$2^3\ 2^2\ 2^1\ 2^0$
- For fractional numbers: ……. $2^3\ 2^2\ 2^1\ 2^0 . 2^{-1}\ 2^{-2}\ 3^{-3}$……..

  Binary Point
- LSB (Least Significant Bit) is the right most bit e.g. $2^0 = 1$
- MSB (Most Significant Bit) is left most bit, its depend on the size of binary numbers.

For fractional numbers:-

- MSB is the left most bit e.g. $2^{-1} = 0.5$
- LSB is the right most bit e.g. 2^{-n}

Binary to Decimal conversion:-

Examples:- (1) Convert 1 1 0 1 1 0 1 to Decimal Number (2) Convert 0.1 0 1 1 to Decimal Number

Sol 01: $2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

1 1 0 1 1 0 1

$= 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 1 \times 64 + 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 64 + 32 + 0 + 8 + 4 + 0 + 1 = 109$

Sol 02: $2^0\ .\ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}$

0 . 1 0 1 1

$= (0 \times 2^0) + (1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}) = (0 \times 1) + (1 \times 0.5 + 0 \times 0.25 + 1 \times 0.125 + 1 \times 0.0625) = (0) + (0.5 + 0 + 0.125 + 0.0625) = 0 + 0 . 6875 = 0. 6875$

Related problems: - Convert 1 0 . 1 1 1 to decimal number

Binary Arithmetic:- The following are binary arithmetic's.

(1) Binary Addition
(2) Binary Subtraction
(3) Binary Multiplication
(4) Binary Division

❖ ***Binary Addition:-***

- These are the four basic rules for adding binary digits.

(1) 0 + 0 = 0 (2) 0 + 1 = 1 (3) 1 + 0 = 1 (4) 1 + 1 = 10

Examples:-

(1) 1 1 + 1 0

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

(2) 1 0 0 + 1 0

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

- **Now if there are three digits:**

(1) 1 + 0 + 0 = 01 (2) 1 + 1 + 0 = 10 (3) 1 + 0 + 1 = 10 (4) 1 + 1 + 1 = 11

Examples:-

(1) 1 1 + 1 1

```
  1  ← carry bit
  1 1
+ 1 1
-----
1 1 0
```

(2) 1 1 1 + 1 1

```
  1 1  ⟶ carry bits
  1 1 1
+   1 1
-------
1 0 1 0
```

❖ **Binary Subtraction:-**

(1) 0 – 0 = 0 (2) 1 – 1 = 0 (3) 1 – 0 = 1 (4) 0 – 1 = 1 (10 - 1) with a borrow of one.

Examples:-

(1) 1 1 – 0 1

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

(2) 1 1 – 1 0

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

(3) 1 0 1 – 0 1 1

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

(4) 1 1 0 – 1 0 1

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


❖ **Binary Multiplication:-**

(1) 0 × 0 = 0 (2) 0 × 1 = 0 (3) 1 × 0 = 0 (4) 1 × 1 = 1

Examples:-

(1) 1 1 × 1 1

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

(2) 1 0 1 × 1 1 1

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

❖ **Binary Division:-**

- Same procedure as division in decimal

(1) 1 1 0 ÷ 1 1

```
11 √110   |10|
   11
   000
```

(2) 1 1 0 ÷ 1 0

```
10 √110   |11|
   10
   010
    10
    00
```

(3) 1 1 0 0 ÷ 1 0 0

```
100 √1100  |11|
    100
    0100
     100
     0000
```


1's and 2's complement of Binary Numbers:-

- It is important because they permit the representation of negative numbers
- The method of 2's complement arithmetic is commonly used in computers to handle negative numbers.

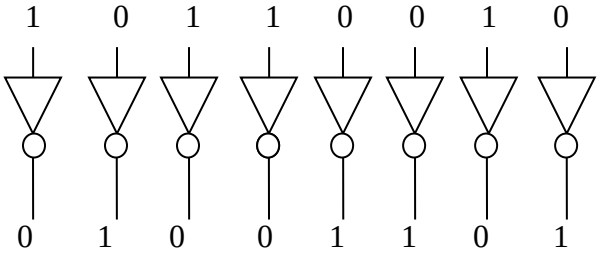
Finding the 1's complement:- To change all 1's to 0's and all 0's to 1's

1 0 1 1 0 0 1 0 ⟶ Binary number

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

0 1 0 0 1 1 0 1

Represent 1's complement of binary number in inverter



Finding the 2's complement:-

- 2's complement = (1's complement) + 1

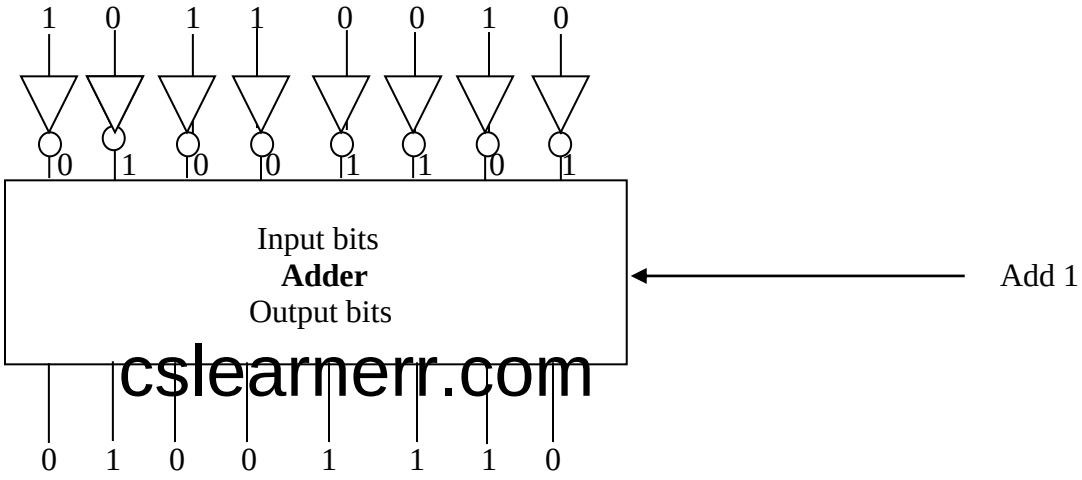
Example:- Find 2's complement of 1 0 1 1 0 0 1 0

1 0 1 1 0 0 1 0 ⟶ *Binary number*

0 1 0 0 1 1 0 1 ⟶ $1's\ complement$

+ 1 ⟶ *add 1*

0 1 0 0 1 1 1 0 ⟶ $2's\ complement$



Week No. 02: Signed Numbers

Signed numbers:-

- Computer (digital systems) must be able to handle both positive and negative numbers
- A signed binary number consists of both sign and magnitude information
- The **sign** indicates weather a number is positive or negative
- And the **magnitude** is the value of the number
- There are three forms in which sign integer (whole) number can be represented in binary:-

  (1) Sign magnitude form (2) 1's complement (3) 2's complement

Sign bit:-

- Left most bit in a signed binary number is the sign bit, which tells you whether the number is positive or negative
- A "0" sign bit indicates a positive number
- A "1" sign bit indicates a negative number

(1) Sign Magnitude Form:-

- When a signed binary number is represented in sign magnitude, the left most bit is the sign bit and the remaining bits are the magnitudes bits

  For example:- A decimal number +25 is expressed as an 8 bit, signed binary number using sign magnitude form as

  0 0 0 1 1 0 0 1

  Sign bit Magnitude bit

  The decimal number -25 is expressed as

  1 0 0 1 1 0 0 1

  Sign bit Magnitude bit

(2) 1's complement form:-

- +25 expressed 0 0 0 1 1 0 0 1 using 8-bits
- -25 expressed 1 1 1 0 0 1 1 0 as 1's complement

(3) 2's complement form:-

- +25 expressed 0 0 0 1 1 0 0 1
- -25 expressed 1 1 1 0 0 1 1 0 + 1 = 1 1 1 0 0 1 1 1 as 2's complement

Arithmetic Operations With Signed Numbers:-

- How signed numbers are added, subtracted, multiplicated and divided.

1. Addition:-

There are four cases when two signed binary number are added.

i. Both numbers positive:-

$$\begin{array}{r} 00000111 \\ +\ 00000100 \\ \hline 00001011 \end{array} \qquad \begin{array}{r} 7 \\ +\ 4 \\ \hline 11 \end{array}$$

ii. Positive number with magnitude larger than negative numbers:-

- For negative number first take 2's complement and then add with positive number.

```
0 0 0 0 0 1 1 0   (- 6)
1 1 1 1 1 0 0 1
          + 1
-------------
1 1 1 1 1 0 1 0   → 2's complement
```

Discard carry bit

```
  0 0 0 0 1 1 1 1        15
+ 1 1 1 1 1 0 1 0      + −6
-----------------      -----
1 0 0 0 0 1 0 0 1         9
```

iii. Negative number with magnitude larger than positive numbers:-

- For negative number first take 2's complement and then add with positive number.

```
0 0 0 1 1 0 0 0   (- 24)
1 1 1 0 0 1 1 1   → 1's complement
          + 1
-------------
1 1 1 0 1 0 0 0   → 2's complement
```

```
  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
```


iv. Both numbers negative:-

- For negative numbers first take 2's complements and then add the resultant numbers.

Discard carry bit

```
  1 1 1 1 1 0 1 1        −5 (11111011  2's complement)
+ 1 1 1 1 0 1 1 1      + −9 (11110111  2's complement)
-----------------      -----
1 1 1 1 0 0 1 0           −14
```

Overflow condition:-

- Indicated by an incorrect sign bit
- There is a carry into the sign bit which produces an overflow condition
- Can occur only when both numbers are positive or both numbers are negative

```
  0 1 1 1 1 1 0 1        125
+ 0 0 1 1 1 0 1 0      +  58
-----------------      -----
1 0 1 1 0 1 1 1           183
```

Sign incorrect — Magnitude incorrect

2. Subtraction:-

- Subtraction is a special case of addition. For example +6 (subtrahend) subtracting from +9 (the minuend) is equivalent to adding -6 to +9, basically, the subtraction operation changes the sign of the subtrahend and adds it to the minuend. The result of a subtraction is called difference.
- The sign of a positive or negative binary number is changed by taking its 2's complement.
- Subtract two signed numbers, take the 2's complement of the subtrahend and add with minuend.

For example:-

(1) 0 0 0 0 1 0 0 0 – 0 0 0 0 0 0 1 1

```
00000011
11111100  → 1's comp.
      +1
---------
11111101  → 2's complement
```

Discard carry bit

```
  0 0 0 0 1 0 0 0
+ 1 1 1 1 1 1 0 1    2's complement
-----------------
1 0 0 0 0 0 1 0 1    Difference positive
```

(2) 0 0 0 0 1 1 0 0 – 1 1 1 1 0 1 1 1

```
  11110111
  00001000   1's complement
+       1
----------
  00001001   2' complement
```

```
  00001100
+ 00001001   2's complement
----------
  00010101   Difference positive
```

(3) 1 1 1 0 0 1 1 1 – 0 0 0 1 0 0 1 1

```
  00010011
  11101100   1's comp.   Discard Cary bit
+       1
----------
  11101101   2's complement
```

```
  11100111
+ 11101101   2's complement
----------
1 11010100   Difference negative
```


3. **Multiplication:-**

- The numbers in a multiplication are the multiplicand, the multiplier and product. e.g.

```
    8   multiplicand
×   3   multiplier
-------
   24   product
```

- Direct addition and partial products are two basic methods for performing multiplication using addition. e.g. 8 × 3 = 8 + 8 + 8 = 24 →**Direct addition product method**. So disadvantage of this approach is that it becomes very lengthy.

Direct addition product method

Example: - Multiply the signed binary numbers 0 1 0 0 1 1 0 1 (multiplicand) and 0 0 0 0 1 0 0 (multiplier) using direct addition method. So the decimal value of the multiplier is 4, so multiplicand is added to itself four times.

```
          0 1 0 0 1 1 0 1  ⟶  1st time
          0 1 0 0 1 1 0 1  ⟶  2nd time
        ------------------
          1 0 0 1 1 0 1 0  ⟶  partial sum
          0 1 0 0 1 1 0 1  ⟶  3rd time
        ------------------
          1 1 1 0 0 1 1 1  ⟶  partial sum
Sign bit  0 1 0 0 1 1 0 1  ⟶  4th time
        ------------------
        0 1 0 0 1 1 0 1 0 0  ⟶  product
```

Partial product method:-

- In this method multiplicand is multiplied by multiplier. E.g. 8 × 3 = 24.
- In sign number if the signs are the same, the product is positive
- If the signs are different, the product is negative.

There are some steps in partial product method:

Step 1: Determine if signs of the multiplicand and multiplier are same, product will be positive and if signs are different product will be negative.

Step 2: If signs are different then takes the 2's complement of the negative number to put into true form.

Step 3: In multiplication only the magnitude bits are used to multiply.

Step 4: If the product sign is 1 (as determined in step1) then take 2's compliment of the product and if positive, leave the product in true form and attach the sign bit.

For example: 0 1 0 1 0 0 1 1 multiply by 1 1 0 0 0 1 0 1

Step1: The sign bit of the multiplicand is 0 and the sign bit of the multiplier is 1. The sign bit of the product will be 1 (negative).

Step2: Take the 2's complement of the multiplier to put it in true form.

11000101 ⟶ 00111011

Step3: The multiplication proceeds as follows.

1010011	Multiplicand
x 0111011	Multiplier
1010011	1st partial product
+ 1010011	2nd partial product
11111001	sum of 1st and 2nd partial product
+ 0000000	3rd partial product
011111001	sum
+1010011	4th partial product
1110010001	sum
+ 1010011	5th partial product
100011000001	sum
+ 1010011	6th partial product
1001100100001	sum
+ 0000000	7th partial product
1001100100001	Final product

Step4: Since the sign of the product is 1 as determined in step1, take the 2's complement of the product.

1001100100001 ⟶ 0110011011111

Attach the sign bit ⟶

1 0110011011111

4. Division:-

- The numbers in a division are the dividend, the divisor and the quotient

$$\frac{Dividend}{Diviser} = \text{Quotient}$$

- If the signs are the same, the quotient is positive
- If the signs are different, the quotient is negative

There are some steps in Division:

Step 1: Determine the signs of dividend and divisor, this determine what will be the sign of quotient. Initialize the quotient to zero.

Step 2: Subtract the divisor from the dividend using 2's complement addition to get the first partial remainder and add 1 to quotient. If this partial remainder is positive, go to step 3. If the partial remainder is zero or negative, the division is completed.

Step 3: Subtract the divisor from the partial remainder using 2's complement addition and add 1 to quotient. If the result is positive, repeat this step for next partial remainder. If the result is zero or negative the division is completed.

Example:- Divide 0 1 1 0 0 1 0 0 by 0 0 0 1 1 0 0 1

Step1: The sign of both numbers are positive, so the quotient will be positive. Initialize the quotient with zero: 00000000

Step2:
01100100 Dividend
\+ 11100111 2's complement of divisor
01001011 Positive 1st partial reminder
Add 1 to quotient: 00000000 + 00000001 = 00000001

Step3:
01001011 1st partial remainder
\+ 11100111 2's complement of divisor
00110010 Positive 2nd partial remainder
Add 1 to quotient: 00000001 + 00000001 = 00000010

Step4:
00110010 2nd partial remainder
\+ 11100111 2's complement of divisor
00011001 positive 3rd partial remainder
Add 1 to quotient: 00000010 + 00000001 = 00000011

Step5:
00011001 3rd partial remainder
\+ 11100111 2's complement of divisor
00000000 Zero remainder
Add 1 to quotient: 00000011 + 00000001 = 00000100 (final quotient)

Hexadecimal Number:-

- Hexadecimal number system has sixteen characters.
- The hexadecimal number system consists of digits 0 - 9 and letters A - F.
- The hexadecimal number has a base of sixteen.
- It is composed of 16, numeric and alphabetic characters.
- Each hexadecimal digit represents a 4-bits binary number.
- Weight: ……16^3 16^2 16^1 16^0
- Ten numeric and six alphabetic characters make up the hexadecimal number system.

Table:-

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

Octal Number:-

- Like the hexadecimal number system, the octal number system provides a convenient way to express binary numbers and codes, however, it is used less frequently than hexadecimal.
- The octal number system is composed of eight digits which are, 0, 1, 2, 3, 4, 5, 6, 7.
- The octal number system has a base of 8.
- Each octal digit represents a 3 – bits binary number.
- Weight:- …….8^3 8^2 8^1 8^0

Binary Coded Decimal (BCD):-

- Binary coded decimal (BCD) is a way to express each of the decimal digits with a binary code of 4-bits.

  The 8421 BCD code:-

  - The 8421 code is a type of BCD code. BCD means that each decimal digit 0 through 9 is represented by a binary code of four bits. The designation 8421 indicates the binary weights of the four bits, 2^3 2^2 2^1 2^0 = 8421.

    8 4 2 1

  Decimal/BCD conversion:-

Decimal Number	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Digital codes:-

The following are some digital codes:-

1. Gray codes
2. ASCII codes
3. Unicode

1. Gray codes:-

The gray code is unweighted and is not an arithmetic code, there is no specific weights assigned to the bit positions. The important feature of the Gray code is that it exhibits only a single bit change from one code word to the next in sequence.

Binary to Gray code conversion:

There are two steps:

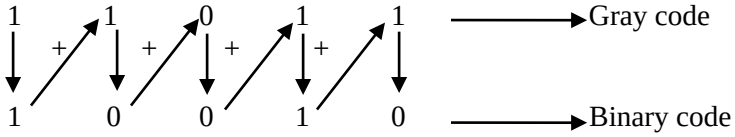
1. MSB in the Gray code is same as the MSB in the binary number.
2. Going from left to right, add each adjacent pair of binary code to get the next Gray code bit and discard carries.

Four-bit Gray Code: For decimal numbers 0 through 15.

Decimal	Binary	Gray code	Decimal	Binary	Gray code
0	0 0 0 0	0 0 0 0	8	1 0 0 0	1 1 0 0
1	0 0 0 1	0 0 0 1	9	1 0 0 1	1 1 0 1
2	0 0 1 0	0 0 1 1	10	1 0 1 0	1 1 1 1
3	0 0 1 1	0 0 1 0	11	1 0 1 1	1 1 1 0
4	0 1 0 0	0 1 1 0	12	1 1 0 0	1 0 1 0
5	0 1 0 1	0 1 1 1	13	1 1 0 1	1 0 1 1
6	0 1 1 0	0 1 0 1	14	1 1 1 0	1 0 0 1
7	0 1 1 1	0 1 0 0	15	1 1 1 1	1 0 0 0

Gray to Binary Code conversion:-

1. MSB in the binary code is the same as MSB in the gray code.
2. Add each binary code bit to the gray code bit in the next adjacent position and discard carries
   For example: - Gray code 1 1 0 1 1 = 1 0 0 1 0 Binary code



2. ASCII codes:-

- ASCII is the abbreviation for "American Standard Code for Information Interchange".
- ASCII is a universally accepted alphanumeric code used in most computers and other electronic equipment. Most computer keyboards are standardized with the ASCII.
- ASCII has 128 characters and symbols represented by a 7-bit binary code. Actually ASCII can be considered an 8-bit code with the MSB always 0. This 8-bit code is 00 through 7F in hexadecimal.

3. Unicode:-

- Unicode provides the ability to encode all of the characters used for the written languages of the world by assigning each character a unique numeric value and name utilizing the universal character set. It is applicable in computer applications.
- It is 32 bit code to represent each letter, number or symbol and it is also compatible with ASCII code and the first 256 symbols are identical with those ones, represented in ASCII.
- Unicode has a wide array of characters and Unicode consists of about 100,000 characters.

Parity:-

- Parity is used to detect a single-bit error. A parity bit tells that the total number of 1's is odd or even in a given binary number.
- 0 or 1 indicates even/odd parity bit
- **Even Parity bit:** An even parity bit makes the total number of 1's even.
- **Odd parity bit:** An odd parity bit makes the total number of 1's odd.
- A given system operates with even or odd parity, but not both. For example, if a system operates with even parity, a check is made on each group of bits received to make sure the total number of 1's in that group is even. If there is an odd number of 1's, an error has occurred.
- Let's assume that we wish to transmit the BCD code 0101. The total code transmitted, including the even parity bit, is

e.g. 0 0 1 0 1

Even parity bit BCD code

0 0 0 0 1

Even parity bit BCD code bit error

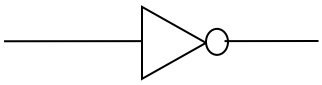
Even Numbers: 2, 4, 6, 8,……..
Odd Numbers: 1, 3, 5,……..

Week No. 03: Logic Gates

Inverter:-

- The inverter (NOT circuit) performs the operation which is called inversion or complementation.
- The inverter changes one logic level to the opposite level. In terms of bits, it changes 1 to 0 and 0 to 1.

Standard logic symbol for the Inverter:-



Inverter Truth Table:-

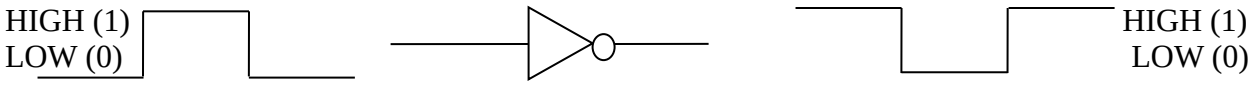
- When a HIGH level is applied to an inverter input, a LOW level will appear on its output.
- When a LOW level is applied to its input, a HIGH level will appear on its output.

INPUT (A)	OUTPUT (X)
LOW (0)	HIGH (1)
HIGH (1)	LOW(0)

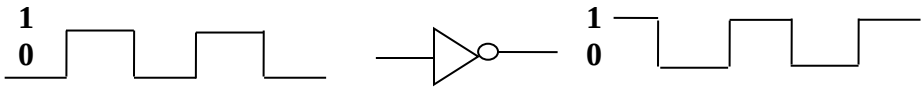
Logic expression for inverter:-

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

Inverter operation: - When the input is LOW, the output is HIGH, when the input is HIGH, the output is LOW, producing an inverted output pulse.



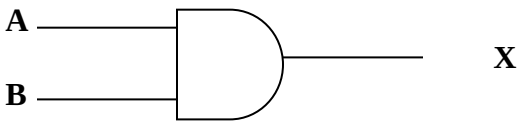
Example:-



AND GATE:-

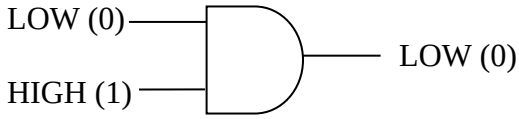
- The AND gate is one of the basic gates that can combined to form any logic function.
- An AND gate can have two or more inputs and performs which is known as logical multiplication.
- The term gate is used to describe a circuit that performs a basic logic operation.
- AND gate is composed of two or more inputs and a single output

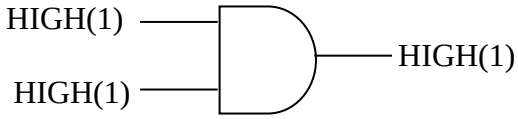
Logic Symbol:-

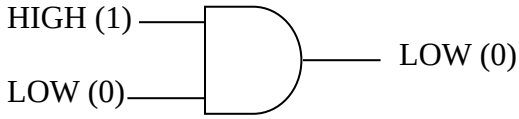


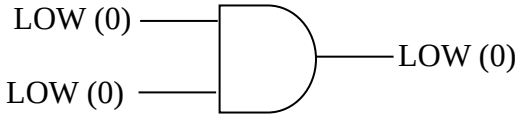
Operation of an AND gate: -

- For two inputs AND gate, output X is high only when input A and B are HIGH.
- Output X is LOW when either A or B is LOW, or when both A and B are LOW.









AND gate Truth Table:-

INPUTS		OUTPUT
A	**B**	**X**
0	0	0
0	1	0
1	0	0
1	1	1

- The total numbers of possible combinations of binary inputs to a gate is determined by the following formula; $N = 2^n$
  Where N is the number of possible input combinations and 'n' is the number of input variables.
- For two inputs variables:- $N = 2^n = 2^2 = 4$ combinations
- For three inputs variables:- $N = 2^n = 2^3 = 8$ combinations
- For four inputs variables:- $N = 2^n = 2^4 = 16$ combinations

For example: - 3-inputs variables truth table:

INPUTS			OUTPUT
A	**B**	**C**	**X**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Logic expression for AND gate:-

X = A.B 0.0 = 0 0.1 = 0 1.0 = 0 1.1 = 1

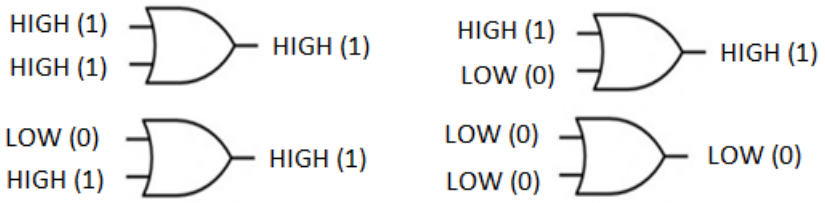
OR GATE: -

- The OR gate is another of the basic gates from which all logic functions are constructed.
- An OR gate can have two or more inputs and performs which is known as logical addition

Standard Symbol:-

Operation of an OR gate:-

- For a 2-inputs OR gates, output X is HIGH when either input A or B is HIGH.
- Or when both A and B are HIGH.
- X output is LOW only when both A and B are LOW.



OR gate Truth Table:-

INPUTS		OUTPUT
A	**B**	**X**
0	0	0
0	1	1
1	0	1
1	1	1

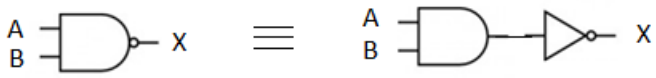
Logic expression for OR gate:-

X = A + B, 0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 1

NAND gate: -

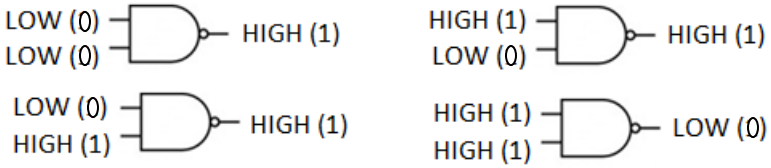
- The NAND gate is a popular logic element because it can be used as a universal gate.
- NAND gate can be used in combination to perform the AND, OR and inverter operations.
- The NAND gate is the same as the AND gate except the output is inverted.
- The term NAND is a construction of NOT-AND and implies an AND function with a complemented (inverted) output.

Standard logic symbol:- For 2-inputs



Operation of NAND gate:-

- For a 2-input NAND gate, output X is LOW only when inputs A and B are HIGH.
- And output X is HIGH when either A or B is LOW, or when both A and B are LOW.



NAND gate Truth Table: -

INPUTS		OUTPUT
A	**B**	**X**
0	0	1
0	1	1
1	0	1
1	1	0

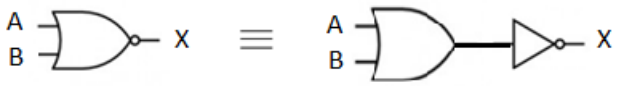
Logic expression for NAND gate: -

$X = \overline{A.B}$ $\overline{0.0} = \overline{0} = 1$ $\overline{0.1} = \overline{0} = 1$ $\overline{1.0} = \overline{0} = 1$ $\overline{1.1} = \overline{1} = 0$

NOR gate: -

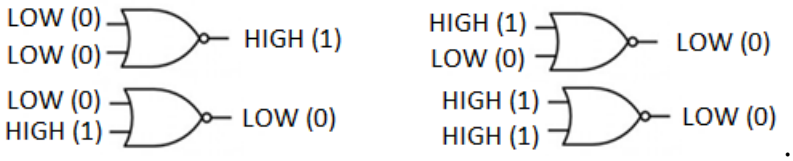
- The NOR gate, like the NAND gate is useful logic element because it can also be used as a universal gate, NOR gates can be used in combination to perform the AND, OR, and inverter operations.
- The NOR gate is the same as the OR gate except the output is inverted
- The term NOR is a construction of NOT-OR and implies OR function with a complemented (inverted) output.

Standard logic symbol: -



Operation of NOR gate:-

- For a 2-input NOR gate, output X is LOW when either input A or B is HIGH, or when both A and B are HIGH
- Output X is HIGH only when both A and B are LOW



NOR gate truth table:-

INPUTS		OUTPUT
A	**B**	**X**
0	0	1
0	1	0
1	0	0
1	1	0

Logic expression for NOR gate:-

$X=\overline{A+B}$ $\quad \overline{0+0} = \overline{0} = 1$ $\quad \overline{0+1} = \overline{1} = 0$ $\quad \overline{1+0} = \overline{1} = 0$ $\quad \overline{1+1} = \overline{1} = 0$

Exclusive-OR and exclusive-NOR gates: -

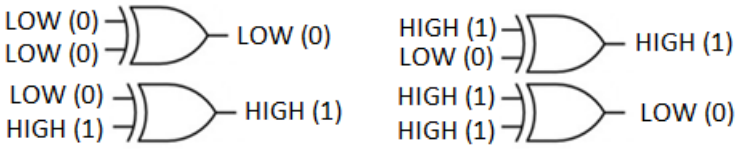
- These gates are formed by a combination of other gates.

Exclusive OR gate:-

Standard logic symbol of XOR gate:-

Exclusive-OR gate operation:-

- For an exclusive-OR gate, output X is HIGH when input A is LOW and input B is HIGH or when input A is HIGH and input B is LOW.
- And output X is LOW when A and B are both HIGH or both LOW.



XOR gate truth table:-

INPUTS		OUTPUT
A	**B**	**X**
0	0	0
0	1	1
1	0	1
1	1	0

Logic expression of XOR:- For 2-inputs

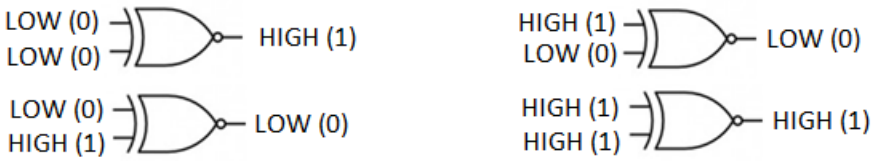
$X=A \oplus B$ **$0 \oplus 0=0$** **$0 \oplus 1=1$** **$1 \oplus 0=1$** **$1 \oplus 1=0$**

Exclusive NOR gate:-

Standard logic symbol:-

Operation of X-NOR gate:-

- For an exclusive-NOR gate, output X is LOW when input A is LOW and input B is HIGH or when A is HIGH and B is LOW.
- And output X is HIGH when A and B are both HIGH and both LOW.



X-NOR gate truth table:-

INPUTS		OUTPUT
A	**B**	**X**
0	0	1
0	1	0
1	0	0
1	1	1

Logic expression for X-NOR gate:-

$X=\overline{A \oplus B}$ **$\overline{0 \oplus 0} = \overline{0} = 1$** **$\overline{0 \oplus 1} = \overline{1} = 0$** **$\overline{1 \oplus 0} = \overline{1} = 0$** **$\overline{1 \oplus 1} = \overline{0} = 1$**

Week No. 04: Boolean Algebra and Logic Simplification

Boolean operations and Expressions:-

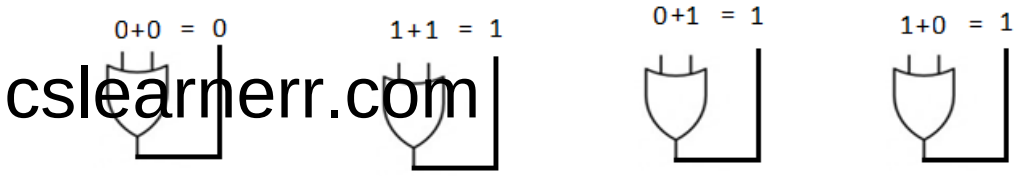
Boolean Algebra is the mathematics of digital systems. A basic knowledge of Boolean algebra is the study and analysis of logic circuits. There are some terms used in Boolean algebra.

i. **Variable:** A variable is a symbol used to represent an action, a condition or data. Any single variable can have only a 1 or 0 value.

ii. **Complement:** The complement is the inverse of a variable and is indicated by a bar over the variable (over bar). For example the complement of the variable A is $\overline{A}$. If A=1, then $\overline{A}$=0. If A=0 then $\overline{A}$=1. The complement of the variable A is read as "not A" or "A bar".

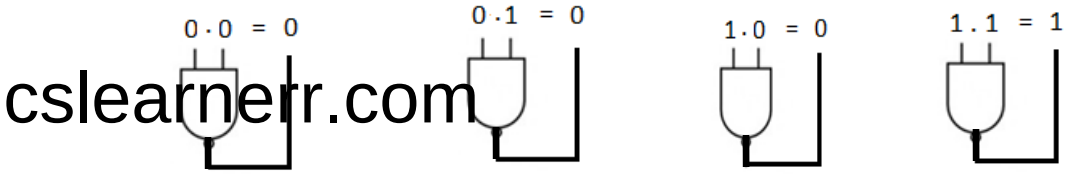
iii. **Literal:** A literal is a variable or the complement of a variable.

Boolean Addition: Boolean addition is equivalent to the OR operation with the basic rules.



- In Boolean algebra, a sum term is a sum of literals
- In logic circuits, a sum term is produced by an OR operation with no AND operations involved.
- Examples: (1) A+B (2) A+$\overline{B}$ (3) A+B+C
- A sum term is equal to 1 when one or more of the literals are 1.
- A sum term is equal to 0 only if each the literal is 0.
- The OR operation is the Boolean form of addition.

Boolean multiplication: - Boolean multiplication is equivalent to the AND operation with the basic rules.



- In Boolean algebra, a product term is the product of literals.
- In logic circuits, a product term is produced by AND operation with no OR operations involved.
- Examples: (1) AB (2) A$\overline{B}$ (3) ABC (4) A$\overline{B}$C$\overline{D}$ etc.
- A product term is equal to 1 only if each of the literals is 1.
- A product term is equal to 0 when one or more of the literals are 0.

Laws and Rules of Boolean Algebra: -

Laws of Boolean Algebra: The basic laws of Boolean algebra are;

Commutative Laws:

1. **Commutative Law for Addition:**
   - The commutative law of addition for two variables is written as A + B = B + A
   - This law states that the order in which the variables are ORed makes no difference.
   - In Boolean algebra as applied to logic circuits, addition and the OR operation are the same.

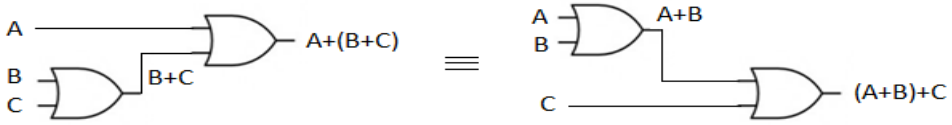
2. **Commutative Law for Multiplication:**
   - The commutative law of multiplication for two variables is AB = BA
   - This law states the order in which the variables are ANDed makes no difference

Associative Laws:

1. **Associative Law for Addition:**
   - The associative law of addition is written as follows for three variables:

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

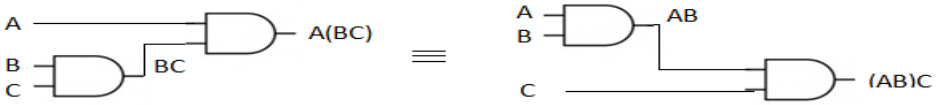
   - This law states that when ORing more than two variables, the result is the same regardless of the grouping of the variables.



2. **Associative Law for Multiplication:**
   - The associative law of multiplication is written as follows for three variables:

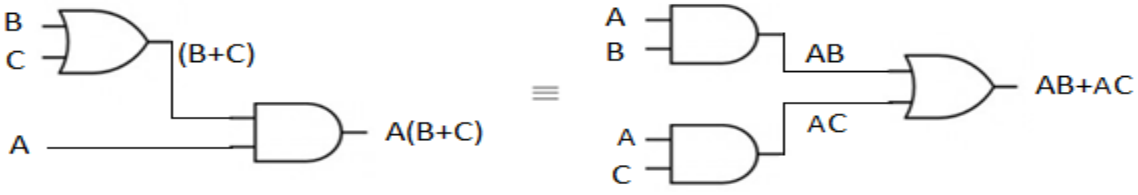
$$A\,(BC) = (AB)\,C$$

   - This states that it makes no difference in which order the variables are grouped when ANDing more than two variables.



Distributive law: -

- The distributive law is written for three variables as A (B + C) = AB + AC
- This law states that ORing two or more variables and then ANDing the result with a single variable is equivalent to ANDing the single variable with each of the two or more variables and then ORing the products.



Rules of Boolean algebra:-

- The following are 12 basic rules that are useful in manipulating and simplifying Boolean expressions.
- Rule 1 through 9 will be viewed in terms of their application to logic gates
- Rule 10 through 12 will be derived in terms of the simpler rules and laws.

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

Rule 1: A+0=A

- A+0=A, a variable ORed with 0 is always equal to the variable.
- If the input variable A is 1, the output variable X is 1, which is equal to A.
- If A is 0, the output is 0 which is equal to A.

So X=A+0=A

Rule 2: A+1=1

- A+1=1, a variable ORed with 1 is always equal to 1
- If the input variable A is 1, the output X is 1
- If the input A is 0, the output X is 1.

So X=A+1=1

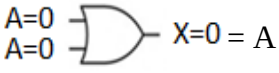
Rule 3: A.0=0

So X=A.0=0

Rule 4: A.1=A

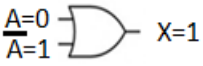
So X=A.1=A

Rule 5: A+A=A



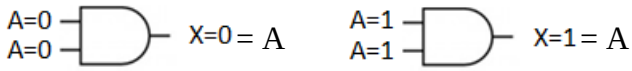
So A+A=A

Rule 6: $A+\bar{A}=1$



So $X=A+\bar{A}=1$

Rule 7: A.A=A

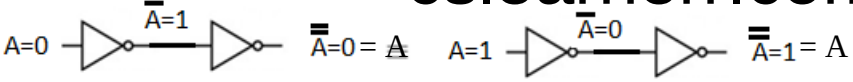


So X=A.A=A

Rule 8: A.$\overline{A}$=0

So X=A.$\overline{A}$=0

Rule 9: $\overline{\overline{A}}$=A



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

Rule 10: A+AB=A

A+AB = A (1+B) (Taking common)

= A.1 (Rule 2: A+1=1)

= A (Rule 4: A.1=A)

Truth Table:

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

Equal

Rule 11: A+$\overline{A}$B=A+B

A+$\overline{A}$B = (A+AB)+$\overline{A}$B (Rule 10: A=A+AB)

= (AA+AB)+$\overline{A}$B (Rule 7: A=A.A)

= AA+AB+A$\overline{A}$+$\overline{A}$B (Rule 8: A$\overline{A}$=0)

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

= (A+B).1 (Rule 6: A+$\overline{A}$=1)

= A+B (Rule 4: A.1=A)

Truth Table:

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

Equal

Rule 12: (A+B) (A+C) = A+BC

(A+B)(A+C) = AA+AC+AB+BC (Distributive law)

= A+AC+AB+BC (Rule 7: A.A=A)

= A(1+C)+AB+BC (Factorizing {Taking common})

= A.1+AB+BC (Rule 2: 1+C=1)

= A(1+B)+BC (Factorizing {Taking common})

= A.1+BC (Rule 2: 1+B=1)

= A+BC (Rule 4: A.1=A)

Truth Table:

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

Equal

De-Morgan's Theorems: -

- De-Morgan's Theorems provide mathematical verification of the equivalency of the NAND and negative-OR gates and the equivalency of the NOR and negative – AND gates.

De Morgan's first theorem:- It is stated as follows:

- The complement of a product of variables is equal to the sum of the complements of the variables.
- Or the complement of two or more ANDed variables is equivalent to the OR of the complements of the individual variables.
- Theorem for two variables: $\overline{XY} = \overline{X} + \overline{Y}$

Truth Table: For two variables:

X	Y	$\overline{X}$	$\overline{Y}$	XY	$\overline{XY}$	$\overline{X} + \overline{Y}$
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

De Morgan's second theorem: - It is stated as follows:

- The complement of a sum of variables is equal to the product of the complements of the variables.
- Or the complement of two or more ORed variables is equivalent to the AND of the complements of individual variables.
- Theorem for two variables: $\overline{X}+\overline{Y} = \overline{X + Y}$

Truth Table: For two variables:

X	Y	$\overline{X}$	$\overline{Y}$	X + Y	$\overline{X+Y}$	$\overline{X}\ \overline{Y}$
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

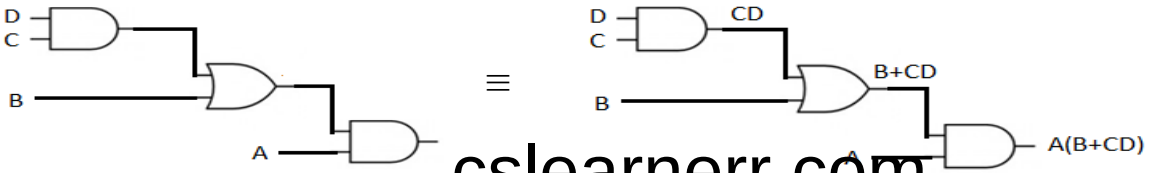
Week No. 05: Boolean Analysis of logic Circuits

Boolean analysis of logic circuit:-

- Boolean algebra provides a concise way to express the operation of a logic circuit formed by a combination of logic gates, so that the output can be determine for various combinations of input values.

Boolean expression for a logic circuit:-

- To derive the Boolean expression for a given combinational logic circuit, begin at the left-most inputs and work toward the final output, writing the expression for each gate. For example:



- A combination logic circuit can be described by a Boolean equation.
- In the figure, a combinational logic circuit showing the development of the Boolean expression for the output.
- In this example, the Boolean expression is determined in the following three steps.
  1. The expression for the left-most AND gate with the inputs C and D is CD.
  2. The output of the left-most AND gate is one of the input to the OR gate and B is the other input. Therefore, the expression for the OR gate is (B + CD)
  3. The output of the OR gate is one of the inputs to the right-most AND gate and A is the other input, therefore, the expression for this AND gate is A (B + CD), which is the final output expression for the entire circuit.

Constructing a Truth Table for a logic circuit:-

- We take the above logic circuit and construct a Truth Table for that logic circuit, from which we derived a Boolean expression A (B + CD).
- Total combination for 4-inputs are $N=2^n=2^4=16$

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

Simplification using Boolean algebra:-

Examples: **(1)** AB+A(B+C)+B(B+C)

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

= AB+AB+AC+BB+BC (Distributive law)

= AB+AB+AC+B+BC (Rule 7: B.B=B)

= AB+AC+B+BC (Rule 5: AB+AB=AB)

= AB+AC+B (Rule 10: B+BC=B)

= AB+B+AC (arranged)

= B+AC (Rule 10: AB+B=B)

= B+AC (answer)

(2) $[A\overline{B}(C+BD)+\overline{A}\overline{B}]C$

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

$=[A\overline{B}C+A\overline{B}BD+\overline{A}\overline{B}]C$ (Distributive law)

$=(A\overline{B}C+A.0.D+\overline{A}\overline{B})C$ (Rule 8: $\overline{B}.B=0$)

$=(A\overline{B}C+0+\overline{A}\overline{B})C$ (Rule 3: A.0.D=0)

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

$=A\overline{B}CC+\overline{A}\overline{B}C$ (Distributive law)

$=A\overline{B}C+\overline{A}\overline{B}C$ (Rule 7: C.C=C)

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

$=\overline{B}C.1$ (Rule 6: $A+\overline{A}=1$)

$=\overline{B}C$ (Rule 4: $\overline{B}C.1=\overline{B}C$)

$=\overline{B}C$ (answer)

(3) $\underline{\overline{A}\,B\,C} + A\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + \underline{A\,B\,C}$

$= B\,C\,(\overline{A} + A) + \underline{A\,\overline{B}\,\overline{C}} + \underline{\overline{A}\,\overline{B}\,\overline{C}} + \underline{A\,\overline{B}\,C}$ (Taking Common)

$= B\,C.1 + A\,\overline{B}\,(\overline{C} + C) + \overline{A}\,\overline{B}\,\overline{C}$ (Rule 6: $\overline{A}+A=1$)

$= B\,C + A\,\overline{B}.1 + \overline{A}\,\overline{B}\,\overline{C}$ (Rule 4: $A\overline{B}.1=A\overline{B}$ & Rule 6: $\overline{C}+C=1$)

$= B\,C + \underline{A\,\overline{B}} + \underline{\overline{A}\,\overline{B}\,\overline{C}}$ (Rule 4: $A\overline{B}.1=A\overline{B}$)

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

$= B\,C + \overline{B}\,(A + \overline{C})$ (Rule 11: $A+\overline{A}\overline{C}=A+\overline{C}$)

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

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

$= (\overline{A\,B})\,(\overline{A\,C}) + \overline{A}\,\overline{B}\,C$ (De-Morgan's Second Theorem)

$= (\overline{A} + \overline{B})\,(\overline{A} + \overline{C}) + \overline{A}\,\overline{B}\,C$ (De-Morgan's First Theorem)

$= \overline{A}\,\overline{A} + \overline{A}\,\overline{C} + \underline{\overline{A}\,\overline{B}} + \overline{B}\,\overline{C} + \underline{\overline{A}\,\overline{B}\,C}$ (Distributive law)

$= \overline{A} + \overline{A}\,\overline{C} + \overline{A}\,\overline{B}\,(1 + C) + \overline{B}\,\overline{C}$ (Rule 7: $\overline{A}.\overline{A} = \overline{A}$ & Taking Common)

$= \underline{\overline{A} + \overline{A}\,\overline{B}} + \overline{B}\,\overline{C}$ (Rule 10: $\overline{A} + \overline{A}\,\overline{C} = \overline{A}\,(1 + \overline{C}) = \overline{A}$ & Rule 2: 1 + C = 1)

$= \overline{A}\,(1 + \overline{B}) + \overline{B}\,\overline{C}$ (Taking $\overline{A}$ as Common)

$= \overline{A}.1 + \overline{B}\ \overline{C}$ (Rule 2: 1+A=1 So $1+\overline{B}=1$)

$= \overline{A} + \overline{B}\,\overline{C}$ (Rule 4: $A.1=A$ So $\overline{A}.1=\overline{A}$)

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

$= A\overline{B} + A\,(\overline{B}\,\overline{C}) + B\,(\overline{B}\,\overline{C})$ (De-Morgan's Second Theorem)

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

$= A\overline{B}\,(1 + \overline{C}) + 0.\overline{C}$ ` (Rule08: $B\overline{B} = 0$)

$= A\overline{B}.1 + 0$ (Rule02: $1 + \overline{C} = 1$ & Rule03: A.0 = 0)

$= A\overline{B} + 0 = A\overline{B}$ (Rule04: A.1 = A & Rule01: A + 0 = A)

(6) $[AB\,(C + \overline{B}D) + \overline{AB}]\,CD$

$= [AB\,(C + \overline{B} + \overline{D}) + \overline{A} + \overline{B}]\,CD$

$= [ABC + AB\overline{B} + AB\overline{D} + \overline{A} + \overline{B}]\,CD$

$= ABC + AB\overline{D} + \overline{A} + \overline{B}]\,CD$

$= ABCCD + AB\overline{D}DC + \overline{A}CD + \overline{B}CD$

$= ABCD + \overline{A}CD + \overline{B}CD = CD\,(AB + \overline{B}) + \overline{A}CD$

$= CD\,(A + \overline{B}) + \overline{A}CD = ACD + \overline{B}CD + \overline{A}CD = CD\,(A + \overline{A}) + \overline{B}CD$

$= CD + \overline{B}CD = CD\,(1 + \overline{B}) = CD$

(7) $AB\overline{C} + \overline{A}\,\overline{B}C + \overline{A}BC + \overline{A}\overline{B}\overline{C}$

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

$= AB\overline{C} + \overline{A}\,\overline{B} + \overline{A}BC$

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

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

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

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

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

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

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

Week No. 06: Standard Forms of Boolean Expressions

Standard Form of Boolean expressions:-

- All Boolean expression can be converted in either of two standard forms:
  1. The Sum-of-Product (SOP) Form.
  2. The Product-of-Sum (POS) Form.

The Sum-of-Product (SOP) Form:-

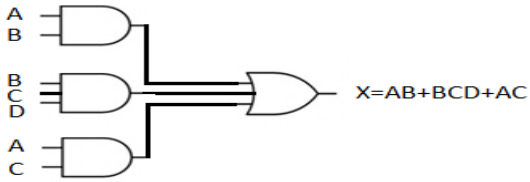
- When two or more products are summed by Boolean addition, the resulting expression is a sum of products (SOP). **Examples**:
  (1) $\bar{A}B+\bar{A}B\bar{C}+AC$ (2) $AB+ABC$ (3) $ABC+CD\bar{E}+\bar{B}C\bar{D}$
- Also an SOP expression can contain a single variable as $A + \bar{A}\,\bar{B}\,C + \bar{B}\,C\,\bar{D}$
- In an SOP expression, a single over bar can't extend over more than one variable
- However, more than one variable can have an over bar, for example an SOP expression can have $\bar{A}\,\bar{B}\,\bar{C}$ but not $\overline{A\,B\,C}$.

Domain of Boolean expression:-

- The domain of a general Boolean expression is the set of variables contained in the expression in either complemented or uncomplemented form.
- **For example:-** The domain of the expression $\bar{A}B+A\bar{B}C$ is the set of variables A,B,C and the domain of the expression $A\bar{B}C + CD\bar{E} + \bar{B}\,C\,\bar{D}$ is the set of variables A,B,C,D and E.

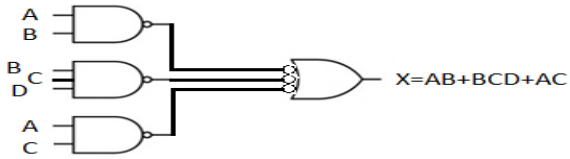
AND/OR implementation of an SOP expression:-

- Implementing an SOP expression simply requires ORing the outputs of two or more AND gates.
- So an SOP expression can be implemented with one OR gate and two or more AND gates.
- **For example**, the expression: AB+BCD+AC, the output X of the OR gate equals to the SOP expression.



NAND/NOR implementation of an SOP expression:-

- NAND gates can be used to implement an SOP expression
- **Example:-** AB+BCD+AC



Conversion of a general expression to SOP Form:-

- Any logic expression can be changed into SOP form by applying Boolean algebra techniques. For example, the expression A (B+CD) can be converted to SOP form by applying the distributive law.
  (1) A(B+CD)=AB+ACD
  (2) AB+B(CD+EF)=AB+BCD+BEF
  (3) $(\overline{\overline{A+B}}) + C = (\overline{\overline{A+B}})\,\bar{C} = (A+B)\,\bar{C} = A\,\bar{C} + B\,\bar{C}$
  (4) (A + B) (B + C + D) = AB + AC + AD + BB + BC + BD

The Standard SOP Form:-

- A standard SOP expression is one in which all the variables in the domain appear in each product in the expression. For example, $A\ \bar{B}\ C\ D + \bar{A}\ \bar{B}\ C\ \bar{D} + A\ B\ \bar{C}\ \bar{D}$ are a standard SOP expression. Standard SOP expressions are important in construction of truth tables.
- Any nonstandard SOP expression can be converted to the standard form using Boolean algebra.

Converting product to Standard SOP:-

- **Example:** $A\ \bar{B}\ C + \bar{A}\ \bar{B} + A\ B\ \bar{C}\ D$, the domain of this SOP is A, B, C, D.
- The first term missing variable D or $\bar{D}$, so multiply first term by $D + \bar{D}$
- Now second term, $\bar{A}\ \bar{B}$ missing C or $\bar{C}$ and D or $\bar{D}$ so first second term are multiply by C or $\bar{C}$ and then the result multiply by D or $\bar{D}$.

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

So

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

Binary representation of a standard product:-

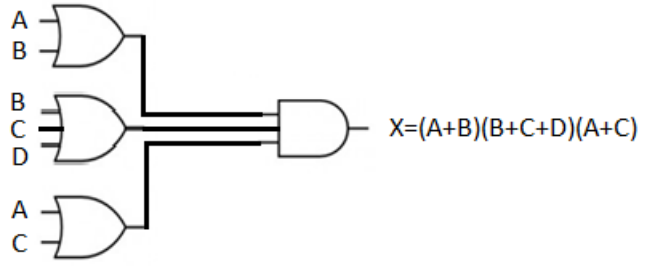
- An SOP expression is equal to 1 only if one or more of the product in the expression is equal to 1.
- Example: $A\ \bar{B}\ C\ \bar{D}$ is equal to 1 when A=1, B=0, C=1, D=0.
- $A\ \bar{B}\ C\ \bar{D} = 1\ .\bar{0}.\ 1\ .\ \bar{0} = 1\ .\ 1\ .\ 1\ .\ 1 = 1$

The Product-of-Sum (POS) Form:-

- When two or more sum are multiplied, the resulting expression is a product-of-sum (POS)
- For example: 1) $(\bar{A} + B)\ (A + \bar{B} + C)$, 2) $(\bar{A} + \bar{B} + \bar{C})\ (C + \bar{D} + E)\ (\bar{B} + C + D)$
- A POS expression can contain a single variable as $\bar{A}\ (A + \bar{B} + C)\ (\bar{B} + \bar{C} + D)$
- In a POS expression, a single over bar cannot extend over more than one variable.
- However, more than one variable can have an over bar.
- For example; a POS expression can have $\bar{A} + \bar{B} + \bar{C}$ but not $\overline{A+B+C}$

Implementation of POS expression:-

- Implementing a POS expression simply requires ANDing the outputs of two or more OR gates.
- For example: the expression (A+B)(B+C+D)(A+C). The output X of the AND gate equals to the POS expression.



The Standard POS Form:-

- A standard POS expression is one in which all variables in the domain appear in each sum in expression. For example: $(\bar{A} + \bar{B} + \bar{C} + \bar{D})\ (\bar{A} + B + C + D)\ (A + \bar{B} + C + D)$.
- Any nonstandard POS expression can be converted to the standard form using Boolean algebra.

Converting a sum to standard POS:-

- Example: - $(A + \bar{B} + C)\ (\bar{B} + C + \bar{D})\ (A + \bar{B} + \bar{C} + D)$, Domain of this POS expression is A,B,C,D. So the first term missing D or $\bar{D}$, so add $D\bar{D}$ and apply rule 12. Now in 2nd term $\bar{A}$ or A missing so add $A\bar{A}$ and apply rule 12.
  $(A + \bar{B} + C)\ (\bar{B} + C + \bar{D})\ (A + \bar{B} + \bar{C} + D)$
  $= (A + \bar{B} + C + D\,\bar{D})\ (\bar{B} + C + \bar{D})\ (A + \bar{B} + \bar{C} + D)$
  $= (A + \bar{B} + C + D)\ (A + \bar{B} + C + \bar{D})\ (\bar{B} + C + \bar{D})\ (A + \bar{B} + \bar{C} + D)$
  $= (A + \bar{B} + C + D)\ (A + \bar{B} + C + \bar{D})\ (\bar{B} + C + \bar{D} + A\,\bar{A})\ (A + \bar{B} + \bar{C} + D)$
  $= (A + \bar{B} + C + D)\ (A + \bar{B} + C + \bar{D})\ (A + \bar{B} + C + \bar{D})\ (\bar{A} + \bar{B} + C + \bar{D})\ (A + \bar{B} + \bar{C} + D)$
  = Thus, $(A + \bar{B} + C)\ (\bar{B} + C + \bar{D})\ (A + \bar{B} + \bar{C} + D)$
  $= (A + \bar{B} + C + D)\ (A + \bar{B} + C + \bar{D})\ (A + \bar{B} + C + \bar{D})\ (\bar{A} + \bar{B} + C + \bar{D})\ (A + \bar{B} + \bar{C} + D)$

Binary representation of a standard sum:-

- A POS expression is equal to 0 only if one or more of the sum in the expression is equal to 0.
- For example: The sum $A+\bar{B}+C+\bar{D}$ is 0, when A=0, B=1, C=0, and D=1.
- $A+\bar{B}+C+\bar{D} = 0 + \bar{1} + 0 + \bar{1} = 0 + 0 + 0 + 0 = 0$.

Boolean expressions and Truth Table:-

- All standard Boolean expressions can be easily converted into truth table format using binary values for each term in the expression.

Converting SOP expressions to Truth Table:-

- For example: $\bar{A}\ \bar{B}\ C + A\ \bar{B}\ \bar{C} + A\ B\ C$ (Three variables are used so $N=2^n=2^3=8$)

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

Converting POS expressions to Truth Table Format:-

- For example: $(A+B+C)(A+\bar{B}+C)(A+\bar{B}+\bar{C})$

A	B	C	$\bar{B}$	$\bar{C}$	A + B + C	A + $\bar{B}$ + C	A + $\bar{B}$ + $\bar{C}$	(A+B+C) (A+$\bar{B}$+C) (A+$\bar{B}$+$\bar{C}$)
0	0	0	1	1	0	1	1	0
0	0	1	1	0	1	1	1	1
0	1	0	0	1	1	0	1	0
0	1	1	0	0	1	1	0	0
1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	1	1
1	1	0	0	1	1	1	1	1
1	1	1	0	0	1	1	1	1