UNIT :3

Boolean Algebra and Logic Gates

3.1 Introduction to Boolean algebra,Variable, Boolean Function:

❖ Boolean algebra:

Boolean algebra is the algebra of logic that deals with the study of binary variables and logical operations. It makes possible to transform logical statements into mathematical symbols and to calculate the truth or falsity of related statements by using rules. It is named after George Boole, in 1854 AD, Mathematician and Philosopher, who was the first to try and to formalize what we call logic or reasoning.

❖ Boolean variable:

A Boolean variable is the variables which have only two states i.e. true/ false or right/ wrong or on/off or 0/1. As a computer is a binary system, it operates on an electronic signal which has only 2 possible states.

The signal that does not change its state with time is called constant signal and its value always remains the same i.e. either 1 or 0 whereas a variable signal continuously changes its state according to the time. At some point, the value of the variable signal may be 1 and at some another point, it might be 0. Therefore, these variables which consist of only two values i.e. 1 and 0 are Boolean variables or logic variables. These variables are denoted by English capital letters like A, B, X, Y, etc.

❖ Boolean algebra as switching algebra:

Boolean algebra which is also known as the Switching algebra consists of two elements (0 and 1) has two binary operators called OR and AND and another unary operator called NOT. The unary operator performs on a single operand whereas a binary operator requires more than one operand. Followings are their symbols:

Operator	**Symbols**
OR	~, +, V, U (Union)
AND	(.),^, Intersection
NOT	Ë…, (‘), (¯), Negation

Differences between Boolean algebra and Ordinary algebra:

Boolean algebra	Ordinary algebra
Its basic operations are AND, OR and NOT operations.	Its basic operations include addition, subtraction, multiplication, and division.
There is no exponents or coefficients involved in Boolean algebra i.e. A+A=A and A.A=A.	It consists of coefficients and exponents such as A+A=2A and $A.A=A^2$.
It has only a finite set of elements. That is, it deals with only two elements; 0 and 1.	It deals with real numbers that contain an infinite number of elements (1, 2, 3…).
It holds both distributive laws: A.(B+C)=(A.B)+(A.C)and A+(B.C)=(A+B).(A+C)	It holds only one distributive law: A.(B+C)=(A.B)+(A.C)
It is used in the field of digital electronics.	It is used in the field of mathematics.

- **Logic function (Boolean function):**

Boolean function, commonly known as a logic function is an expression expressed algebraically with binary variables, logical operation symbols, parenthesis and equal sign. For a given value of the binary variables, the logic function can be either 0 or 1.

Example: Consider the logic function in algebraic expression:

F = X.Y.Z'+X. Y Here, X, Y, Z are Boolean variables. The equation on the right-hand side above is known as an expression. Each occurrence of a variable or its complement in an expression is called literal. So, there are altogether three variables i.e. X, Y and Z and five literals; X, Y, Z', X, and Y.

Logical circuit by using transistors or Integrated Circuits (IC) or (LSI), the high and low-level voltage electrical signals called logical elements are produced. These logical elements are essential for the composition of a circuit for a specific operation called logical circuit.

Basic Logical/ Boolean Operation:

Introduction: An operator is a special symbol that indicates the operation to be carried out between two operands. An operation is an action to be carried out upon operands. There are 3 basic Boolean Operations: AND, OR and NOT operations.

- **AND operation:** Known as logical multiplication, it is carried out by dot (.) operator or simply by AND. If the inputs are true, it generates true output. Otherwise, it generates false output. Its logical equation is written as C=A.B or C=A AND B. The truth table of AND operation is:

Inputs		**Output**
A	**B**	**C=A.B**
False	False	False
False	True	False
False	False	False
True	True	False

- **OR operation** Known as logical addition, it is carried out by plus (+) operator or simply by OR. If at least one input is true, it generates true output or else, it gives false output. The logical equation of OR operation is written as C=A+B or C=A OR B. The truth table of OR operation is given below:

Inputs		**Output**
A	**B**	**C=A+B**
False	False	False
False	True	True
True	False	True
True	True	True

- **NOT Operation** Also known as the logical compliment, it is carried out by prime (') operator or bar (¯). It generates the output opposite the input i.e. if the input is true, it generates false output and vice versa. It's logical equation can be written as C=A'. The truth table of OR operation is:

Inputs	**Output**
A	**C=A'**
True	False
False	True

Truth Table:

The truth table is a table of all possible combinations of the variables showing the relation between the values that variables may take and the result of the operation. The table used to represent the Boolean expression of a logic gate function called a truth table. A truth table shows each possible input combination to the gate or circuit with the resultant output depending upon the combination of input.

3.2 Logic Gates: AND, OR, NOT, NAND, NOR, XOR, and XNOR- its definition, use, truth table, logic symbol:

A logic gate is an electronic circuit that operates on one or more input signals to produce an output signal. A logic gate is also known building block of a digital circuit. Mostly, the logic gate consists of two inputs and one output. Gates produce the signals 1 or 0 if input requirements are satisfied. Digital computer uses different types of logical gates. Each gate has a specific function and graphical symbol. The function of the gate is expressed by means of an algebraic expression. The basic gates are described below:

- **AND Gate:**

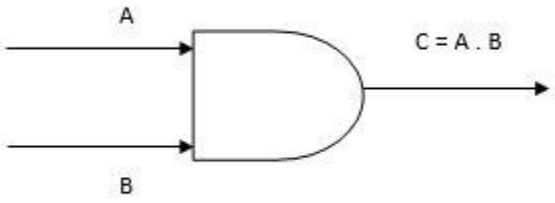
The AND Gate contain two or more than to input values which produce only one output value. AND gate produces 1 output when all inputs are 1, otherwise the output will be 0. It can be explained with the help of two switches connected in series. In AND gate, current is flowing in the circuit only when both switches, A and B, are closed.

The graphical symbol, logical circuit, algebraic expression and truth table of AND gate is shown below:

Truth table:

Input		Output
A	B	C = A . B
0	0	0
0	1	0
1	0	0
1	1	1

Graphical Symbol:



Algebraic Expression is, C = A . B

- **OR Gate:**

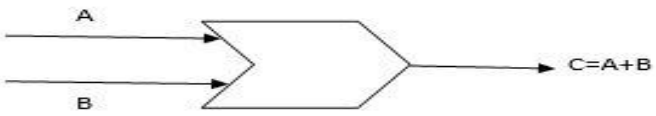
The OR Gate contains two or more than two input values which produce only one output value. OR gate produces 1 output, when one of the inputs is 1. If inputs are 0, then the output will be also 0. It can be explained by taking an example of two switches connected in parallel.

The graphical symbol, algebraic expression and truth table of OR gate is as shown below:

Truth table:

Input		Output
A	B	C = A +B
0	0	0
0	1	1
1	0	1
1	1	1

Graphical Symbol



Algebraic Expression is, C = A+B

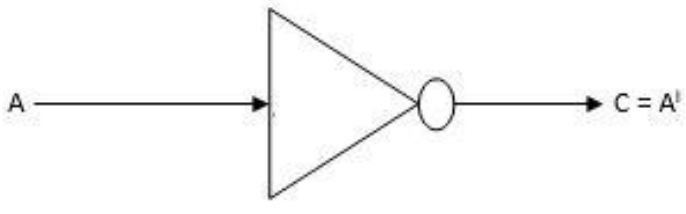
- **NOT Gate:**

The NOT Gate contains only one input value which produces only one output value. This gate is also known as an inverter. So, this circuit inverts the logical sense of a binary signal. It produces the complemented function. If the input is 1, then this gate will produce 0 as output and vice-versa. The graphical symbol, algebraic expression and truth table of a NOT gate is given below.

Truth table:

A	A^{I}
0	1
1	0

Graphical Symbol:



Algebraic Expression is, $C = A^{I}$

- **NAND Gate:**

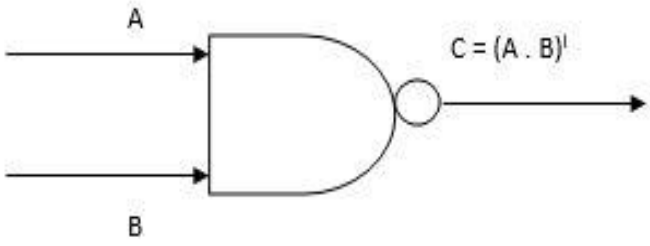
The NAND Gate contains two or more than two input values which produce only one output value. This gate is the combination of AND and NOT gates. This gate is a complement of AND function. This gate produces output 0, when all inputs are 1, otherwise, output will be 1.

The graphical symbol, algebraic expression and truth table of NAND gate is shown below:

Truth table:

Input		Output
A	B	$C=(A \cdot B)^{'}$
0	0	1
0	1	1
1	0	1
1	1	0

Graphical Symbol:



Algebraic Expression is, $C = (A \cdot B)^{'}$

- **NOR Gate:**

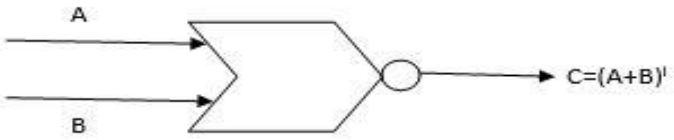
The NOR Gate contains two or more than two input values which produce only one output value. This gate is a combination of OR and NOT gate. This gate is the complement of the OR function. This gate produces 1 output, when all inputs are 0 otherwise output will 0.

The graphical symbol, algebraic expression and truth table of NOR gate is given below:

Truth table:

Input		Output
A	B	$C = (A+B)^{'}$
0	0	1
0	1	0
1	0	0
1	1	0

Graphical Symbol:



Algebraic Expression, $C = (A + B)^{'}$

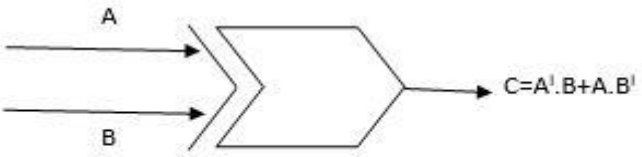
- **Exclusive OR (X-OR) Gate:**

This gate contains two or more than two input values which produce only one output value. The graphical symbol of X-OR gate is similar to OR gate except for the additional curve line on the input side. This gate produces 1 as output, if any input is 1 and 0 if both inputs are either 1 or 0, otherwise its output is 0. The graphical symbol, algebraic expression and truth table of X-OR gate isgivenbelow:

Truth table:

Input		Output
A	B	$C = A^{I} . B + A . B^{I}$
0	0	0
0	1	1
1	0	1
1	1	0

Graphical Symbol:



Algebraic Expression is, $C = A^{I} . B + A . B^{I}$

- **Exclusive NOR (X-NOR) Gate:**

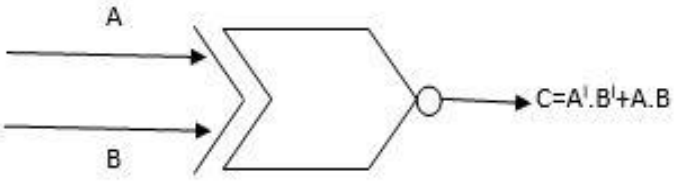
This gate contains two or more than two input values which produce only one output value. The X-NOR is the complement of the X-OR, as indicated by the small circle in the graphical symbol. This gate produces 1 output, when all inputs are either 0 or 1, otherwise its output value is 0.

The graphical symbol, algebraic expression and truth table of X-NOR gate is shown below:

Truth table:

Input		Output
A	B	$C = A^{I} . B^{I} + A . B$
0	0	0
0	1	1
1	0	1
1	1	0

Graphical Symbol:



Algebraic Expression is, $C = A^{I} . B^{I} + A . B$

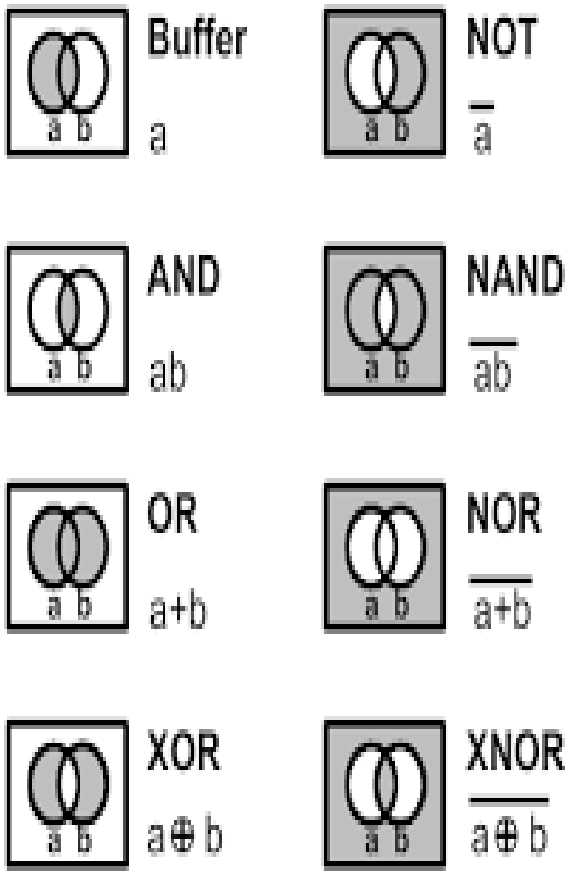


Fig: Venn diagram of All Gates

3.3 Laws of Boolean Algebra:

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

3.4 De-Morgan's Theorem – Statement and Logical Expression:

- **De Morgan's Theorem:**

De Morgan's theorem is associated with Boolean algebra, which was given by great logical and mathematician, De Morgan. So, it is called "De Morgan's theorem".

It consists of first and second theorem which are described below:

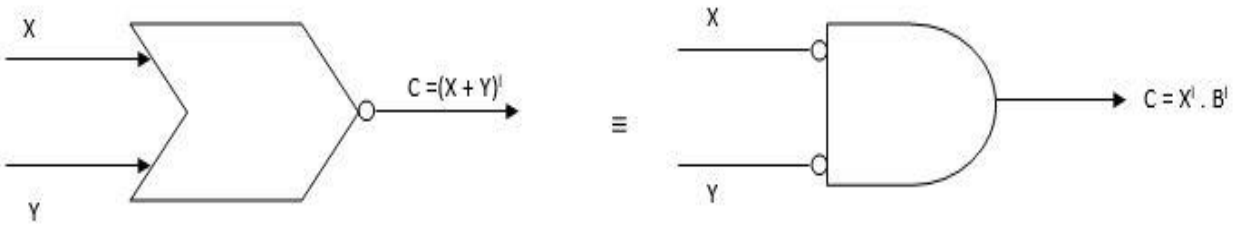
- **First Theorem:**

The De Morgan's first theorem states, "The complement of the sum is equal to the product of complement of individual variable". Let X and Y be two Boolean variables then De Morgan's theorem mathematically expressed as $(X + Y)^{l} = X^{l} . Y^{l}$ for two variable

(A+B+C)'=A'.B'.C' for three variable.

Proof:

Graphical Symbol:



Truth Table:

Inputs			Output 1			Output 2
X	Y	X + Y	$(X + Y)'$	X'	Y'	$X' . Y'$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Conclusion: comparing the value of $(X + Y)'$ and $X' . Y'$ from the truth table, both are equal, hence proved.

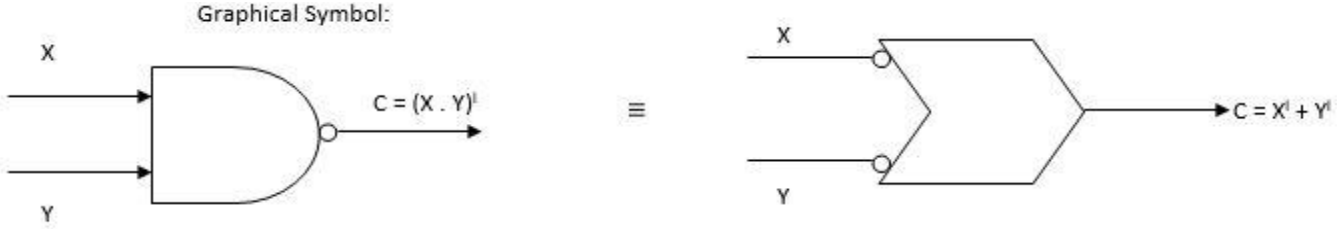
- **Second Theorem:**

De Morgan's second theorem states," The complement of a product is equal to the sum of the complements of individual variable". Let X and Y be two Boolean variables then De Morgan's theorem mathematically expressed as $(X . Y)' = X' + Y'$ for 2 variables.

(A.B.C)'=A'+B'+C' for three variables.

Proof:

Graphical Symbol:



Truth Table:

Inputs			Output 1			Output 2
X	Y	X.Y	$(X.Y)'$	X'	Y'	$X' + Y'$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Conclusion: Comparing the values of $(X.Y)'$ and $X' + Y'$ from the truth table both are equal, hence proved.

3.5 Minterm, Maxterm, SoP and PoS, Canonical and Standard Form:

A minterm is a product (AND) of all variables in the function, in direct or complemented form.
Example:
The min terms are x'y', x'y, xy' and xy.
A maxterm is a sum (OR) of all the variables in the function, in direct or complemented form.
Example:
The Max terms are x + y, x + y', x' + y and x' + y'.

The following table shows the representation of min terms and MAX terms for 2 variables.

x	y	Min terms	Max terms
0	0	m_0=x'y'	M_0=x + y
0	1	m_1=x'y	M_1=x + y'
1	0	m_2=xy'	M_2=x' + y
1	1	m_3=xy	M_3=x' + y'

❖ SoP (Sum of Product):

SOP stands for Sum of Products. Writing a Boolean expression using product terms is called Sum of Products form. The product terms are also known as min-terms. An example is as follows.

For instance, assume that P and Q are input variables and F is the output variable. We take the complement of the variable for 0 and take the variable for 1. Then we can write the Minterm by writing the product terms.

P	Q	F	Minterms
0	0	0	P'Q'
0	1	1	P'Q
1	0	1	PQ'
1	1	1	PQ

Finally, we can take the sum of all the minterms that has 1 for F. Therefore; the final expression is as follows.

F = P'Q + PQ' + PQ

❖ PoS (Product of Sum):

POS stands for Product of Sums. Writing a Boolean expression using sum terms is called Product of Sum form. We also call the sum terms as max-terms.

For example, assume that P and Q are input variables and F is the output variable. Here, we take the variable for 0 and take the complement of the variable for 1. Then we can write the max terms by writing the sum terms.

P	Q	F	Maxterms
0	0	0	P + Q
0	1	1	P+ Q'
1	0	1	P'+Q
1	1	0	P'+Q'

Finally, we can take the product of all max terms that has 0 for F. Thus; the final expression is as follows.

F = (P+Q). (P' + Q')

❖ Canonical SoP and PoS forms:

There are two ways of expressing function they are :

Canonical SoP form
Canonical PoS form

Canonical SoP form:

Canonical SoP form means Canonical Sum of Products form. In this form, each product term contains all literals. So, these product terms are nothing but the min terms. Hence, canonical SoP form is also called as sum of min terms form.
Example
The corresponding min terms are p'qr, pq'r, pqr', pqr. By doing logical OR of these four min terms, we will get the Boolean function of output .
Therefore, the Boolean function of output is, f = p'qr + pq'r + pqr' + pqr. This is the canonical SoP form of output, f. We can also represent this function in following two notations.

f=m3+m5+m6+m7f=m3+m5+m6+m7
f=∑m(3,5,6,7)f=∑m(3,5,6,7)
In one equation, we represented the function as sum of respective min terms. In other equation, we used the symbol for summation of those min terms.

Canonical PoS form:

Canonical PoS form means Canonical Product of Sums form. In this form, each sum term contains all literals. So, these sum terms are nothing but the Max terms. Hence, canonical PoS form is also called as product of Max terms form.
This Boolean function will be in the form of product of Max terms.
Follow the same procedure for other output variables also, if there is more than one output variable.
Example
The corresponding Max terms are p + q + r, p + q + r', p + q' + r, p' + q + r. By doing logical AND of these four Max terms, we will get the Boolean function of output ff.
Therefore, the Boolean function of output is, f = p+q+rp+q+r.p+q+r′p+q+r′.p+q′+rp+q′+r.p′+q+rp′+q+r.
This is the canonical PoS form of output. We can also represent this function in following two notations.
f=M0.M1.M2.M4f=M0.M1.M2.M4
f=∏M(0,1,2,4)f=∏M(0,1,2,4)
In one equation, we represented the function as product of respective Max terms. In other equation, we used the symbol for multiplication of those Max terms.
The Boolean function, f = p+q+rp+q+r.p+q+r′p+q+r′.p+q′+rp+q′+r.p′+q+rp′+q+r is the dual of the Boolean function, f = p'qr + pq'r + pqr' + pqr.
Therefore, both canonical SoP and canonical PoS forms are Dual to each other. Functionally, these two forms are same. Based on the requirement, we can use one of these two forms.

Standard SoP and PoS forms:

We discussed two canonical forms of representing the Boolean outputs. Similarly, there are two standard forms of representing the Boolean outputs. These are the simplified version of canonical forms.
Standard SoP form
Standard PoS form
The main advantage of standard forms is that the number of inputs applied to logic gates can be minimized. Sometimes, there will be reduction in the total number of logic gates required.

Standard SoP form:
Standard SoP form means Standard Sum of Products form. In this form, each product term need not contain all literals. So, the product terms may or may not be the min terms. Therefore, the Standard SoP form is the simplified form of canonical SoP form.
We will get Standard SoP form of output variable in two steps.
Get the canonical SoP form of output variable
Simplify the above Boolean function, which is in canonical SoP form.
Follow the same procedure for other output variables also, if there is more than one output variable.
Sometimes, it may not possible to simplify the canonical SoP form. In that case, both canonical and standard SoP forms are same.

Example
Convert the following Boolean function into Standard SoP form.
f = p'qr + pq'r + pqr' + pqr
The given Boolean function is in canonical SoP form. Now, we have to simplify this Boolean function in order to get standard SoP form.
Step 1 − Use the Boolean postulate, x + x = x. That means, the Logical OR operation with any Boolean variable 'n' times will be equal to the same variable. So, we can write the last term pqr two more times.
⇒ f = p'qr + pq'r + pqr' + pqr + pqr + pqr
Step 2 − Use Distributive law for 1st and 4th terms, 2nd and 5th terms, 3rd and 6th terms.
⇒ f = qrp′+pp′+p + prq′+qq′+q + pqr′+rr′+r
Step 3 − Use Boolean postulate, x + x' = 1 for simplifying the terms present in each parenthesis.
⇒ f = qr11 + pr11 + pq11
Step 4 − Use Boolean postulate, x.1 = x for simplifying above three terms.
⇒ f = qr + pr + pq
⇒ f = pq + qr + pr
This is the simplified Boolean function. Therefore, the standard SoP form corresponding to given canonical SoP form is f = pq + qr + pr

Standard PoS form:

Standard PoS form means Standard Product of Sums form. In this form, each sum term need not contain all literals. So, the sum terms may or may not be the Max terms. Therefore, the Standard PoS form is the simplified form of canonical PoS form.
We will get Standard PoS form of output variable in two steps.
Get the canonical PoS form of output variable
Simplify the above Boolean function, which is in canonical PoS form.
Follow the same procedure for other output variables also, if there is more than one output variable.
Sometimes, it may not possible to simplify the canonical PoS form. In that case, both canonical and standard PoS forms are same.
Example
Convert the following Boolean function into Standard PoS form.
f = p+q+rp+q+r.p+q+r′p+q+r′.p+q′+rp+q′+r.p′+q+rp′+q+r
The given Boolean function is in canonical PoS form. Now, we have to simplify this Boolean function in order to get standard PoS form.
Therefore, both Standard SoP and Standard PoS forms are Dual to each other.

NOTE: Above Concept will be fully used in Upcoming Units. So, students are requested to take a look, a very serious look at above topics too.