Boolean Logic

chapter 3

In This Chapter

3.1 Development of Boolean Logic

Long ago *Aristotle* constructed a complete system of formal logic and wrote six famous works on the subject. Centuries afterward, *George Boole* could manipulate these symbols successfully to arrive at a solution with his own mathematical system of logic, which led to the development of new system, the *algebra of logic*, 'BOOLEAN ALGEBRA' or 'BOOLEAN LOGIC'.

In 1938, *Claude E. Shannon* applied Boolean Logic to solve relay logic problems and published it in paper titled, '*A symbolic Analysis of Relay Switching Circuits*'. As logic problems are binary decisions and Boolean logic effectively deals with these binary values. Thus it is also called '*Switching Algebra*'.

3.2 Binary Valued Quantities

The logic decision which results into either YES (TRUE) or NO (FALSE) is called a Binary Decision, *e.g.*, '*Should I write the answer with blue pen or black pen ?*' Binary decision-making also applies to formal logic. For example, let us consider the following :

1. Indira Gandhi was the only woman Prime Minister of India.
2. 13 – 2 = 11.
3. Delhi is the biggest state in India.
4. What do you say ?
5. What did I tell you yesterday ?

BINARY DECISION

The logic decision which results into either YES (TRUE) or NO (FALSE) is called a Binary Decision.

1st and 2nd sentences are TRUE but 3rd is FALSE ; 4th and 5th are questions which cannot be answered in TRUE and FALSE.

Thus, sentences which can be determined to be *true* or *false* are called *logical statements or truth functions* and the results TRUE or FALSE are called *truth values*. The truth values are depicted by *logical constants* TRUE and FALSE or 1 and 0, *i.e.*, 1 means TRUE and 0 means FALSE. And the variables which can store these truth values are called *logical variables* or ***binary valued variables*** as these can store one of the two values TRUE or FALSE.

Values *true* and *false* are called Truth values.

NOTE

Boolean variables can have value either as 1 (True) or as 0 (False)

3.3 Logical Operations

There are some specific operations that can be applied on logical statements or truth functions. Before learning about these operations, you must know about compound logical functions and logical operators.

3.3.1 Logical Function or Compound Statement

Algebraic variables like *a, b, c* or *x, y, z* etc. are combined with the help of *mathematical operators* like $+, -, \times, /$ to form algebraic expressions *e.g.*,

$$2 \times A + 3 \times B - 6 \times C = (10 \times Z)/2 \times Y \quad \textit{i.e.,} \quad 2A + 3B - 6C = 10Z/2Y$$

Similarly, logic statements or truth functions are combined with the help of *Logical Operators* like AND, OR and NOT to form a *Compound statement or Logical function. e.g.,*

He prefers tea *not* coffee.

He plays guitar *and* she plays sitar.

I watch TV on Sundays *or* I go for swimming.

These logical operators are also used to combine logical variables and logical constants to form *logical expressions e.g.,* assuming *x, y* are logical variables

X NOT Y OR Z

Y AND X OR Z

3.3.2 Logical Operators

Before we start discussion about logical operators, let us first understand *what a Truth Table is.*

For example, following logical statements can have only one of the *two* values (TRUE (YES) or FALSE (NO))

1. I want to have tea. **2.** Tea is readily available.

TRUTH TABLE

A *Truth Table* is a table which represents all the possible values of logical variables / statements along with all the possible results of the given combinations of values.

Let us represent all the possible combinations of values these statements can have in the tabular form :

I want to have tea	T	T	F	F
Tea is readily available	T	F	T	F
(Result) I'll have tea	T	F	F	F

T represents True
F represents False

Or If we represent first statement as X and second statement as Y and result as R then the above table can also be written as follows :

Table 3.1

X	Y	R
1	1	1
1	0	0
0	1	0
0	0	0

1 represents TRUE value and
0 represents FALSE value

TAUTOLOGY

If the result of any logical statement or expression is always TRUE or 1 for all input combinations, it is called Tautology.

FALLACY

If the result of any logical statement or expression is always FALSE or 0 for all input combinations, it is called Fallacy.

This is a truth table *i.e., table of truth values of truth functions.*

Now let us proceed with our discussion about logical operators *i.e.,*

- NOT Operator
- OR Operator
- AND Operator

NOT Operator

This operator **operates on single variable** and operation performed by NOT operator is called *complementation* and the symbol we use for it is $^{-}$ (bar). Thus $\overline{X}$ means complement of X and $\overline{YZ}$ means *complement of* YZ. The complement operation can be defined quite simply.

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

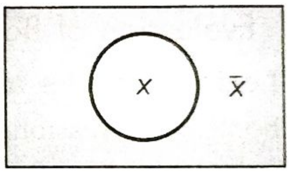


Figure 3.1 Venn diagram for $\overline{X}$

Table 3.2 *Truth Table for NOT Operators*

X	$\overline{X}$ (i.e., NOT X)
0	1
1	0

Several other symbols *e.g.* ~, ' are also used for the complementation symbol. If ~ is used then ~ X is read is 'negation of X' and if ' is used then X' is read as complement of X.

NOTE

NOT operation is singular or unary operation as it operates on single variable.

OR Operator

Another important operator is OR operator which denotes operation called *logical addition* and the symbol we use for it is +. The + symbol, therefore, does not have the 'normal' meaning, but is a *logical addition* or logical OR symbol. Thus X + Y can be read as **X OR Y**. For OR operation the possible input and output combinations are as follows :

$$0 + 0 = 0, \qquad 0 + 1 = 1,$$
$$1 + 0 = 1, \qquad 1 + 1 = 1$$

And the truth table of OR operator is given below :

Shaded Portion shows X + Y

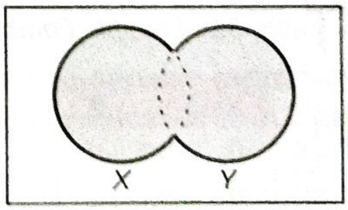


Figure 3.2 Venn diagram for X + Y.

Table 3.3 *Truth Table for OR Operator*

X	Y	X + Y (i.e., X OR Y)
0	0	0
0	1	1
1	0	1
1	1	1

Note that when any one of X and Y is 1, X + Y is 1.

To avoid ambiguity, there are other symbols *e.g.,* ∪ and ∨ have been recommended as replacements for the + sign.

AND Operator

AND operator performs another important operation of boolean algebra called *logical multiplication* and the symbol for AND operation is (.) dot. Thus $X \cdot Y$ will be read as X AND Y. The rules for AND operation are :

$0 . 0 = 0, \quad 0 . 1 = 0,$

$1 . 0 = 0, \quad 1 . 1 = 1$

and the truth table for AND is as follows :

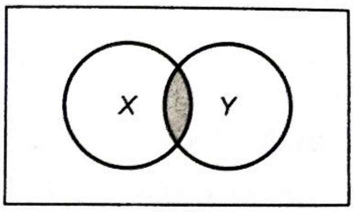


Figure 3.3 Venn diagram for $(X . Y)$.

Table 3.4 *Truth Table for AND Operator*

X	Y	X . Y (i.e., X AND Y)
0	0	0
0	1	0
1	0	0
1	1	1

Note that only when both X and Y are 1's, then XY has the result 1.

Venn diagram for $X.Y$ is given in Fig. 3.3, where shaded area depicts $(X . Y)$.

3.3.3 Evaluation of Boolean Expressions using Truth Table

Logical variables are combined by means of logical operators (AND, OR, NOT) to form a boolean expression. For example, $X + \overline{Y.Z} + \overline{Z}$ is a boolean expression.

It is often convenient to shorten $X . Y . Z$ to XYZ, and using this convention, above expression can be written as $X + \overline{YZ} + \overline{Z}$

> **NOTE**
> A truth table of n input variables with have 2^n input combinations *i.e.*, 2^n rows *e..g.*, a 4-variable truth table will have 2^4 *i.e.*, 16 rows in it.

To study a boolean expression, it is very useful to construct a table of values for the variables and then to evaluate the expression for each of the possible combinations of variables in turn.

Consider the expression $X + \overline{YZ}$. Here three variables X, Y, Z are forming the expression, each of the variables can assume the value 0 or 1. The possible combinations of values may be arranged in ascending order as in Table 3.5. Next a column is added to list Y.Z (Table 3.6)

To see Truth Table Formation in action

Scan QR Code

Table 3.5 *Possible Combinations of X, Y and Z*

X	Y	Z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Since X, Y, Z are three (3) variables in total a truth table involving 3 input variables will have 2^3 i.e., 8 rows in total. The left most column will have half of total entries (i.e., 4 entries) as zeros and half as 1's (in total 8).

The next column will have no of zero's and 1's halved than first column completing 8 rows and so on. That is why, first column has 4 0's and 4 1's, next column has two 0's followed by two 1's completing 8 rows in total and the last column has one 0's followed by one 1's completing 8 rows in total.

Table 3.6 *Truth Table for (Y . Z)*

X	Y	Z	Y . Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Here AND operation is applied only on columns Y and Z

One more column is now added to list the values of $\overline{YZ}$ (Table 3.7).

Table 3.7 *Truth Table for Y . Z and $\overline{YZ}$.*

X	Y	Z	Y . Z	$\overline{YZ}$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

Note that $\overline{YZ}$ contains complemented values of YZ

Now values of X are ORed (*logical addition*) to the values of $\overline{YZ}$ and the resultant values are contained in the last column (Table 3.8).

Table 3.8 *Truth Table for $X + \overline{YZ}$.*

X	Y	Z	Y . Z	$\overline{YZ}$	$X + \overline{YZ}$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	0	1

Now observe the expression $X + \overline{YZ}$, after ANDing Y and Z, the result has been complemented and then ORed with X.

Here the result is 0 only when both the columns X and $\overline{YZ}$ have 0, otherwise if there is 1 in any of the two columns X and $\overline{YZ}$, the result is 1.

Please note here, while evaluating boolean expression there is a precedence order which is to be taken care of always. Always the order of evaluation of logical operators is firstly NOT then AND and then OR. If there are parenthesis, then the expression in parenthesis is evaluated first.

Check Point 3.1

1. **Name the person who developed boolean logic.**
2. **What is the other name of boolean logic ? In which year was the boolean logic/algebra developed ?**
3. **What is a binary decision ? What do you mean by a binary valued variable ?**
4. **What do you mean by tautology and fallacy ?**
5. **What is a logic gate ? Name the three basic logic gates.**

EXAMPLE 1 *Using Boolean logic, verify using truth table that $X + XY = X$ for each X, Y in {0, 1}.*

SOLUTION

As the expression $X + XY = X$ is a two-variable expression, so we require possible combination of values of X, Y. Truth Table will be as follows :

X	Y	XY	X + XY
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Comparing the columns $X + XY$ and X, we find, contents of both the columns are identical, hence verified.

EXAMPLE 2 *Using Boolean logic, verify using truth table that $(X + Y)' = X' Y'$ for each X, Y in {0, 1}.*

SOLUTION As it is a 2-variable expression, truth table will be as follows :

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

Comparing the columns $(X + Y)'$ and $X' Y'$, both the columns are identical, hence verified.

EXAMPLE 3 *Prepare a table of combinations for the following boolean logic expressions :*

(a) $\overline{X}\,\overline{Y} + \overline{X}Y$ (b) $XY\overline{Z} + \overline{X}\,\overline{Y}Z$ (c) $\overline{X}Y\overline{Z} + X\overline{Y}$

SOLUTION (a) As $\overline{X}\overline{Y} + \overline{X}Y$ is a 2-variable expression, its truth table is as follows :

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X}\,\overline{Y}$	$\overline{X}\,Y$	$\overline{X}\,\overline{Y} + \overline{X}\,Y$
0	0	1	1	1	0	1
0	1	1	0	0	1	1
1	0	0	1	0	0	0
1	1	0	0	0	0	0

(b) Truth table for this 3 variable expression is as follows :

X	Y	Z	$\overline{X}$	$\overline{Y}$	$\overline{Z}$	$XY\overline{Z}$	$\overline{X}\,\overline{Y}Z$	$XY\overline{Z} + \overline{X}\,\overline{Y}Z$
0	0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	1	1
0	1	0	1	0	1	0	0	0
0	1	1	1	0	0	0	0	0
1	0	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0	0
1	1	0	0	0	1	1	0	1
1	1	1	0	0	0	0	0	0

Check Point 3.2

1. Which gates implement logical addition, logical multiplication and complementation ?
2. What is the other name of NOT gate ?
3. What is a truth table ? What is the other name of truth table ?
   (i) $A + 0 = ?$ (ii) $A + 1 = ?$
   (iii) $A . 0 = ?$ (iv) $A . 1 = ?$
4. How many input combinations can be there in the truth table of a logic system having (N) input binary variables ?

(c) Truth table for $\overline{X}Y\overline{Z} + X\overline{Y}$ is as follows :

X	Y	Z	$\overline{X}$	$\overline{Y}$	$\overline{Z}$	$\overline{X}Y\overline{Z}$	$X\overline{Y}$	$\overline{X}Y\overline{Z} + X\overline{Y}$
0	0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0	0
0	1	0	1	0	1	1	0	1
0	1	1	1	0	0	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	0	1	0	0	1	1
1	1	0	0	0	1	0	0	0
1	1	1	0	0	0	0	0	0

EXAMPLE 4 *Prepare truth table for the following boolean expressions :*

(*a*) $X(\overline{Y}+\overline{Z})+X\overline{Y}$ (*b*) $X\overline{Y}(Z+Y\overline{Z})+\overline{Z}$ (*c*) $A[(\overline{B}+C)+\overline{C}]$

SOLUTION (*a*) Truth table for $X(\overline{Y}+\overline{Z})+X\overline{Y}$ is as follows :

X	Y	Z	$\overline{Y}$	$\overline{Z}$	$(\overline{Y}+\overline{Z})$	$X(\overline{Y}+\overline{Z})$	$X\overline{Y}$	$X(\overline{Y}+\overline{Z})+X\overline{Y}$
0	0	0	1	1	1	0	0	0
0	0	1	1	0	1	0	0	0
0	1	0	0	1	1	0	0	0
0	1	1	0	0	0	0	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	0	1
1	1	1	0	0	0	0	0	0

(*b*) Truth Table for $X\overline{Y}(Z+Y\overline{Z})+\overline{Z}$ is as follows :

X	Y	Z	$\overline{Y}$	$\overline{Z}$	$Y\overline{Z}$	$Z+Y\overline{Z}$	$X\overline{Y}$	$X\overline{Y}(Z+Y\overline{Z})$	$X\overline{Y}(Z+Y\overline{Z})+\overline{Z}$
0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	0	1	0	0	0
0	1	0	0	1	1	1	0	0	1
0	1	1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	1	0	1
1	0	1	1	0	0	1	1	1	1
1	1	0	0	1	1	1	0	0	1
1	1	1	0	0	0	1	0	0	0

(*c*) Truth Table for $A[(\overline{B}+C)+\overline{C}]$ is as follows :

A	B	C	$\overline{B}$	$\overline{C}$	$(\overline{B}+C)$	$(\overline{B}+C)+\overline{C}$	$A[(\overline{B}+C)+\overline{C}]$
0	0	0	1	1	1	1	0
0	0	1	1	0	1	1	0
0	1	0	0	1	0	1	0
0	1	1	0	0	1	1	0
1	0	0	1	1	1	1	1
1	0	1	1	0	1	1	1
1	1	0	0	1	0	1	1
1	1	1	0	0	1	1	1

3.4 Basic Logic Gates

After *Shannon* applied boolean logic in telephone switching circuits, engineers realized that boolean algebra could be applied to computer electronics as well.

In the computers, these boolean operations are performed by logic gates.

What is a Logic Gate ?

A Gate is a basic electronic circuit which operates on one or more signals to produce an output signal.

Gates are digital (two-state) circuits because the input and output signals are either low voltage (denotes 0) or high voltage (denotes 1). Gates are often called *logic circuits* because they can be analyzed with boolean logic. There are *three* types of logic gates :

- Inverter (NOT gate)
- OR gate
- AND gate

GATE

A Gate is a basic electronic circuit which operates on one or more signals to produce an output signal.

3.4.1 Inverter (NOT Gate)

An **Inverter (Not Gate)** is a gate with only one input signal and one output signal; the output state is always the opposite of the input state. The output is sometimes called the *complement* (opposite) of the input.

INVERTER (NOT GATE)

An Inverter (Not Gate) is a gate with only one input signal and one output signal; the output state is always the opposite of the input state.

Following tables summarise the operation :

Table 3.9 *Truth Table for NOT gate*

X	$\overline{X}$
Low	High
High	Low

Table 3.10 *Alternative Truth Table for NOT Gate*

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

A low input *i.e.*, 0 produces high output *i.e.*, 1, and vice versa. The symbol for inverter is given in adjacent Fig. 3.4.

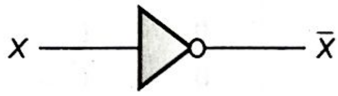


Figure 3.4 NOT gate symbol

3.4.2 OR Gate

The **OR Gate** has two or more input signals but only one output signal. If any of the input signals is 1 (high), the output signal is 1 (high). If all inputs are 0 then output is also 0. If one or more inputs are 1, the output is 1.

OR GATE

The OR Gate has two or more input signals but only one output signal. If any of the input signals is 1 (high), the output signal is 1 (high).

Following tables show OR action.

Table 3.11 *Two Input OR Gate*

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

$F = X + Y$

Table 3.12 *Three Input OR Gate*

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$F = X + Y + Z$

The symbol for OR gate is given below :

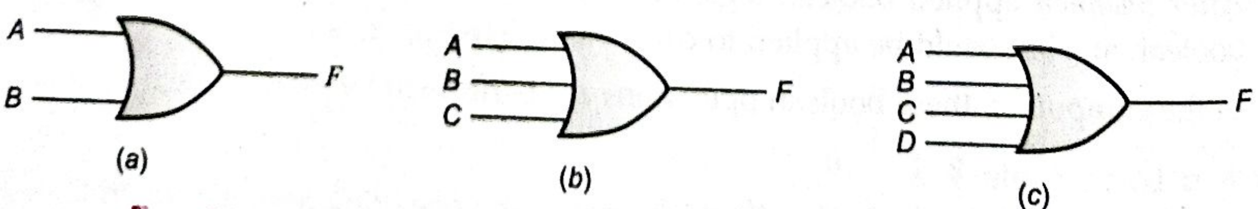


Figure 3.5 (*a*) Two input OR gate (*b*) Three input OR gate (*c*) Four input OR gate.

3.4.3 AND Gate

The **AND Gate** can have two or more than two input signals and produce an output signal. When all the inputs are 1 *i.e.*, high then the output is 1 otherwise output is 0 only. If any of the inputs is 0, the output is 0. To obtain output as 1, all inputs must be 1.

AND GATE

The AND Gate can have two or more than two input signals and produce one output signal. When all the inputs are 1 *i.e.*, high then the output is 1 otherwise output is 0 only.

Table 3.13 *Two Input AND Gate*

X	Y	F
0	0	0
0	1	0
1	0	0
1	1	1

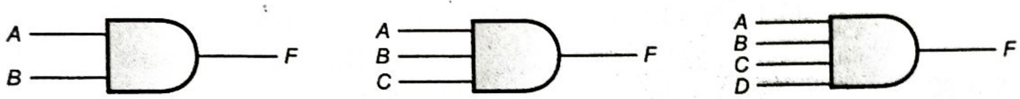
Here, F = X . Y

Table 3.14 *Three Input AND Gate*

X	Y	Z	F
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

Here, F = X . Y . Z

The symbol for AND is



(*a*) (*b*) (*c*)

Figure 3.6 (*a*) 2-input AND gate (*b*) 3-input AND gate (*c*) 4-input AND gate

EXAMPLE 5 *From the expressions given below, identify the logic gates that can implement such expressions.*

(*i*) XYZ (*ii*) $\overline{Y}Z$ (*iii*) $\overline{X}+Y$ (*iv*) $P+Q$ (*v*) $P+\overline{Q}+\overline{R}$ (*vi*) $\overline{A}B+\overline{C}D$ (*vii*) $A\overline{B}\,\overline{C}$

SOLUTION (*i*) $X.Y.Z \rightarrow$. signifies AND gate, hence only AND gates required.

(*ii*) $\overline{Y}.Z \rightarrow$ ¯ signifies NOT and . signifies AND, hence NOT and AND gates.

(*iii*) $\overline{X}+Y \rightarrow$ ¯ signifies NOT and + signifies OR, hence NOT and OR gates.

(*iv*) $P+Q \rightarrow$ + signifies OR, hence OR gate.

(*v*) $P+\overline{Q}+\overline{R} \rightarrow$ ¯ signifies NOT and + signifies OR, hence NOT and OR gates.

(*vi*) $\overline{A}.B+\overline{C}.D \rightarrow$ ¯ signifies NOT, . signifies AND and + signifies OR, hence NOT, AND and OR gates.

(*vii*) $A.\overline{B}.\overline{C} \rightarrow$ ¯ signifies NOT, . signifies AND, hence NOT and AND gates.

3.5 Basic Postulates of Boolean Logic

Boolean logic algebra, being a system of mathematics, consists of *fundamental laws* that are used to build a workable, cohesive framework upon which are based the theorems of boolean algebra. These fundamental laws are known as *Basic postulates of boolean logic algebra*. These postulates state basic relations in boolean algebra, that follow :

I. If $X \neq 0$ then $X = 1$; and If $X \neq 1$ then $X = 0$

II. OR Relations (*Logical Addition*)

$0+0=0$; $0+1=1$; $1+0=1$; $1+1=1$

III. AND Relations (*Logical Multiplication*)

$0.0=0$; $0.1=0$; $1.0=0$; $1.1=1$

IV. Complement Rules : $\overline{0}=1$; $\overline{1}=0$

3.6 Principle of Duality

This is a very important principle used in boolean logic. This states that every Boolean expression has a **dual expression**, which can be derived by :

1. changing each OR sign (+) to an AND sign (.)
2. changing each AND sign (.) to an OR sign (+)
3. replacing each 0 by 1 and each 1 by 0.

For example, $1 + X = 1$ has dual as $0 . X = 0$ (as per above rules). In Boolean Algebra, if an expression is *true*, its dual will also be *true* and vice-versa.

3.7 Basic Theorems of Boolean Algebra/Logic

Basic postulates of Boolean algebra are used to define *basic theorems of boolean algebra* that provide all the tools necessary for manipulating boolean expressions.

3.7.1 Properties of 0 and 1

(*a*) $0 + X = X$ (*b*) $1 + X = 1$ (*c*) $0 . X = 0$ (*d*) $1 . X = X$

Proof. (*a*) $0 + X = X$

Truth table for above expression is :

Table 3.15 *Truth Table for* $0 + X = X$.

0	X	R (0 + X)
0	0	0
0	1	1

as X can have values either 0 or 1 (*postulate I*) both the values ORed with 0 produce the *same output as that of X*.

Hence proved.

(*b*) $1 + X = 1$

Truth table for this expression is :

Table 3.16 *Truth Table for* $1 + X = 1$

1	X	R (1 + X)
1	0	1
1	1	1

Again X can have values 0 or 1. Both the values (0 and 1) ORed with 1 produce the output as 1. Hence proved.

Therefore $1 + X = 1$ *is a* ***tautology***.

(*c*) $0 . X = 0$

The truth table for this expression is :

Table 3.17 *Truth Table for* $0 . X = 0$.

0	X	R (0 . X)
0	0	0
0	1	0

Both the values of X(0 and 1) when ANDed with produce the *output as* 0.

Hence proved.

Therefore, $0 . X = 0$ is a ***fallacy***.

(*d*) $1 . X = X$

The truth table for this expression is :

Table 3.18 *Truth Table for* $1 . X = X$

1	X	R (1 . X)
1	0	0
1	1	1

Now observe both the values (0 and 1) when ANDed with 1 produce the *same output as that of* X. Hence proved. Here properties *b* and *c* are duals of each other and properties *a* and *d* are duals of each other.

3.7.2 Indempotence Law

This law states that : (*a*) $X + X = X$ (*b*) $X . X = X$

Proof. (*a*) $\mathbf{X + X = X}$

Using its truth table :

Table 3.19 *Truth Table for* $X + X = X$

X	X	R
0	0	0
1	1	1

$0 + 0 = 0$ (*ref. postulate II*)

and $1 + 1 = 1$ (*ref. postulate II*)

$\Rightarrow$ $X + X = X$, as it holds true for both values of X. Hence proved.

(*b*) $\mathbf{X . X = X}$

Using its truth table :

Table 3.20 *Truth Table for* $X . X = X$

X	X	R
0	0	0
1	1	1

$0 . 0 = 0$ (*ref. postulate III*)

and $1 . 1 = 1$ (*ref. postulate III*)

$\Rightarrow$ $X . X = X$, as it holds true for both values of X. Hence proved.

(*a*) and (*b*) are duals of each other.

3.7.3 Involution

This law states that : $(\overline{\overline{X}}) = X$

To prove this, again we'll prepare truth table which is given below.

Table 3.21 *Truth Table for* $\overline{\overline{X}} = X$

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

In Boolean Algebra, if an expression holds *true* then its dual is also *true* and vice-versa.

We can see that complement of $\overline{X}$ *i.e.,* $\overline{\overline{X}}$ column has the same values as the column X. Hence proved. This law is also called *double-inversion rule.*

3.7.4 Complementarity Law

These laws state that : (*a*) $X + \overline{X} = 1$ (*b*) $X . \overline{X} = 0$

Proof. (*a*) $\mathbf{X + \overline{X} = 1}$

Using its truth table :

Table 3.22 *Truth Table for* $X + \overline{X} = 1$

X	$\overline{X}$	$X+\overline{X}$
0	1	1
1	0	1

X and $\overline{X}$ values are ORed and the output is shown in third column as

$0 + 1 = 1$, (*ref. postulate II*)

$1 + 0 = 1$ (*ref. postulate II*)

$\Rightarrow X + \overline{X} = 1$, as it holds true for both possible values of X.

Hence proved. It is a *tautology*.

(*b*) $\mathbf{X . \overline{X} = 0}$

Using its truth table :

Table 3.23 *Truth Table for* $X . \overline{X} = 0$

X	$\overline{X}$	$X.\overline{X}$
0	1	0
1	0	0

as $0 . 1 = 0$ (*ref. postulate III*)

and $1 . 0 = 0$ (*ref. postulate III*)

$\Rightarrow$ $X . \overline{X} = 0$, as it holds true for both the values of X. Hence proved. It is a *fallacy*.

Observe here $X . \overline{X} = 0$ *is dual* of $X + \overline{X} = 1$.

Changing (+) to (.) and 1 to 0, and we get $X . \overline{X} = 0$.

3.7.5 Commutative Law

These laws state that : (*a*) $X + Y = Y + X$ (*b*) $X . Y = Y . X$

Proof. (*a*) $X + Y = Y + X$

Truth Table for $X + Y = Y + X$ is given below :

Table 3.24 *Truth Table for X + Y = Y + X*

X	Y	X+Y	Y+X
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Compare the columns $X + Y$ and $Y + X$, both of these are identical. Hence proved.

(*b*) $X . Y = Y . X$

Truth table for $X . Y = Y . X$ is given below :

Table 3.25 *Truth Table for X .Y = Y .X*

X	Y	X . Y	Y . X
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Both of the columns $X . Y$ and $Y . X$ are identical, hence proved.

3.7.6 Associative Law

These laws state that : (*a*) $X + (Y + Z) = (X + Y) + Z$ (*b*) $X (YZ) = (XY) Z$

Proof. (*a*) Truth table for $X + (Y + Z) = (X + Y) + Z$ is given below.

Table 3.26 *Truth Table for X + (Y + Z) = (X + Y) + Z*

X	Y	Z	Y + Z	X + Y	X + (Y + Z)	(X + Y) + Z
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Compare the columns $X + (Y + Z)$ and $(X + Y) + Z$, both of these are identical. Hence proved. Since rule (*b*) is dual of rule (*a*), hence it is also proved.

3.7.7 Distributive Law

This law states that : (*a*) $X (Y + Z) = XY + XZ$ (*b*) $X + YZ = (X + Y)(X + Z)$

Proof. (*a*) Truth table for $X(Y + Z) = XY + XZ$ is given below :

Table 3.27 *Truth Table for X (Y + Z) = XY + XZ*

X	Y	Z	Y + Z	XY	XZ	X(Y + Z)	XY + XZ
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	0	1	1	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1

Both the columns $X(Y + Z)$ and $XY + YZ$ are identical, **hence proved.**

(*b*) Since rule (*b*) is dual of rule (*a*), hence it is also proved.

However, we are giving the algebraic proof of law $X + YZ = (X + Y)(X + Z)$

R.H.S. $= (X + Y)(X + Z) = XX + XZ + XY + YZ$

$= X + XZ + XY + YZ$ ($XX = X$, *Indempotence law*)

$= X + XY + XZ + YZ = X(1 + Y) + XZ + YZ$

$= X.1 + XZ + YZ$ ($1 + Y = 1$, *property of 0 and 1*)

$= X + XZ + YZ$ ($X.1 = X$, *property of 0 and 1*)

$= X(1 + Z) + YZ = X.1 + YZ$ ($1 + Z = 1$, *property of 0 and 1*)

$= X + YZ$ ($X.1 = X$, *property of 0 and 1*)

= L.H.S.

Hence proved.

NOTE

$X + YZ$ expression is **sum of two product**-terms ($X.1, YZ$) and $(X + Y)(X + Z)$ is product of sum-terms ($X + Y, X + Z$). So, this law is a useful one to convert a sum-of-product type expression to product-of-sum type expression and vice-versa.

3.7.8 Absorption Law

According to this law : (*a*) $X + XY = X$ (*b*) $X(X + Y) = X$

Proof.

(*a*) $X + XY = X$

Truth table for $X + XY = X$ is given below :

Table 3.28 *Truth Table for* $X + XY = X$

X	Y	XY	X + XY
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Column X and $X + XY$ are identical. Hence proved. Also it can be proved algebraically as

L.H.S. $= X + XY$

$= X(1 + Y)$

Putting $1 + Y = 1$

(*ref. properties of 0, 1 Theorem 1*)

$X.1 = X$ = R.H.S. (*ref. properties of 0, 1*)

Hence proved.

(*b*) $X(X + Y) = X$

Since rule (*b*) is dual of rule (*a*), it is also proved. However, we are giving the algebraic proof of this law.

L.H.S. $= X(X + Y) = X.X + XY$

$= X.X + XY$

$= X + XY$

($X.X = X$: *Indempotence Law*)

$= X(1 + Y)$

$= X.1$

(*using* $1 + Y = 1$, *properties of 0, 1*)

$= X$ ($X.1 = X$, *using property of 0, 1*)

= R.H.S

Hence proved.

3.7.9 Some Other Rules of Boolean Logic Algebra

There are some more rules of Boolean algebra which are given below :

$X + \overline{X}Y = X + Y$ (Sometimes also referred to as the *third distributive law*)

This rule can easily be proved by truth tables.

All the theorems of Boolean algebra, which we have covered so far, are summarised in the following table :

1.	*Properties of 0*	$0 + X = X$; $0 . X = 0$
2.	*Properties of 1*	$1 + X = 1$; $1 . X = X$
3.	*Indempotence law*	$X + X = X$; $X . X = X$
4.	*Involution*	$\overline{\overline{X}} = X$
5.	*Complementarity law*	$X + \overline{X} = 1$; $X . \overline{X} = 0$
6.	*Commutative law*	$X + Y = Y + X$; $X . Y = Y . X$
7.	*Associative law*	$X + (Y + Z) = (X + Y) + Z$; $X (YZ) = (XY) Z$
8.	*Distributive law*	$X (Y + Z) = XY + XZ$; $X + YZ = (X + Y)(X + Z)$
9.	*Absorption law*	$X + XY = X$; $X . (X + Y) = X$
10.	*Other (3rd distributive law)*	$X + \overline{X}Y = X + Y$

3.8 DeMorgan's Theorems

One of the most powerful identities used in Boolean logic is DeMorgan's theorem. *Augustus DeMorgan* had paved the way to Boolean logic by discovering these two important theorems. This section introduces these two theorems of DeMorgan.

3.8.1 DeMorgan's First Theorem

It states that $\overline{X + Y} = \overline{X}\ \overline{Y}$

Proof. To prove this theorem, we need to recall complementarity laws, which state that

$$X + \overline{X} = 1 \text{ and } X . \overline{X} = 0$$

i.e., a logical variable/expression when added with its complement produces the output as 1 and when multiplied with its complement produces the output as 0.

Now to prove DeMorgan's first theorem, we will use complementarity laws.

Let us assume that $P = X + Y$ where, P, X, Y are Logical/Boolean variables. Then, according to complementation law : $P + \overline{P} = 1$ and $P . \overline{P} = 0$.

That means, if P, X, Y are Boolean variables then this complementarity law must hold for variable P too. In other words, if $\overline{P}$ *i.e.*, if $\overline{X + Y} = \overline{X}\ \overline{Y}$ then

$(X + Y) + \overline{X}\ \overline{Y}$ must be equal to 1. *(as* $X + \overline{X} = 1$*)*

and $(X + Y) . \overline{X}\ \overline{Y}$ must be equal to 0. *(as* $X . \overline{X} = 0$*)*

Let us first prove the first part, *i.e.*,

$$(X + Y) + (\overline{X}\ \overline{Y}) = 1$$

$$\begin{aligned}(X + Y) + \overline{X}\ \overline{Y} &= ((X + Y) + \overline{X}).((X + Y) + \overline{Y}) && (ref.\ X + YZ = (X + Y)(X + Z))\\ &= (X + \overline{X} + Y).(X + Y + \overline{Y}) \\ &= (1 + Y).(X + 1) && (ref.\ X + \overline{X} = 1)\\ &= 1.1 && (ref.\ 1 + X = 1)\\ &= 1\end{aligned}$$

So first part is proved.

Now let us prove the second part *i.e.,*

$$(X+Y).\overline{X}\,\overline{Y}=0$$

$$\begin{aligned}(X+Y).\overline{X}\,\overline{Y} &= \overline{X}\,\overline{Y}.(X+Y) && (ref.\ X(YZ)=(XY)Z)\\ &= \overline{X}\,\overline{Y}\,X+\overline{X}\,\overline{Y}\,Y && (ref.\ X(Y+Z)=XY+XZ)\\ &= X\,\overline{X}\,\overline{Y}+\overline{X}\,Y\,\overline{Y}\\ &= 0.\overline{Y}+\overline{X}.0 && (ref.\ X.\overline{X}=0)\\ &= 0+0=\mathbf{0}\end{aligned}$$

So, second part is also proved, thus : $\overline{X+Y}=\overline{X}\,\overline{Y}$

3.8.2 DeMorgan's Second Theorem

This theorem states that : $\overline{X.Y}=\overline{X}+\overline{Y}$

Proof. Again to prove this theorem, we will make use of complementarity law *i.e.,*

$$X+\overline{X}=1 \quad \text{and} \quad X.\overline{X}=0.$$

If XY's complement is $\overline{X}+\overline{Y}$ then it must be true that

(*a*) $XY+(\overline{X}+\overline{Y})=1$ and (*b*) $XY(\overline{X}+\overline{Y})=0$

To prove the *first part*

$$\begin{aligned}\text{L.H.S} &= XY+(\overline{X}+\overline{Y})\\ &= (\overline{X}+\overline{Y})+XY && (ref.\ X+Y=Y+X)\\ &= (\overline{X}+\overline{Y}+X).(\overline{X}+\overline{Y}+Y)\\ & && (ref.\ X+YZ=(X+Y)(X+Z))\\ &= (X+\overline{X}+\overline{Y}).(\overline{X}+Y+\overline{Y})\\ &= (1+\overline{Y}).(\overline{X}+1) && (ref.\ X+\overline{X}=1)\\ &= 1.1 && (ref.\ 1+X=1)\\ &= 1=\text{R.H.S}\end{aligned}$$

Now the *second part i.e.,*

$$XY.(\overline{X}+\overline{Y})=0$$

$$\begin{aligned}\text{L.H.S} &= XY.(\overline{X}+\overline{Y})\\ &= XY\overline{X}+XY\overline{Y}\\ & && (ref.\ X(Y+Z)=XY+XZ)\\ &= X\overline{X}Y+XY\overline{Y}\\ &= 0.Y+X.0 && (ref.\ X.\overline{X}=0)\\ &= 0+0=0=\text{R.H.S.}\end{aligned}$$

$$XY.(\overline{X}+\overline{Y})=0$$

and $XY+(\overline{X}+\overline{Y})=1$

$\Rightarrow$ $\overline{XY}=\overline{X}+\overline{Y}$. Hence the theorem.

Although the identities above represent DeMorgan's theorem, the transformation is more easily performed by following these steps :

(*i*) Complement the entire function

(*ii*) Change all the ANDs (.) to ORs (+) and all the ORs (+) to ANDs (.)

(*iii*) Complement each of the individual variables.

NOTE

'Break the line, change the sign" to demorganize a boolean expression.

This process is called *demorganization.* For example,

$$\begin{aligned}\overline{\overline{AB}+\overline{A}+AB} &= \overline{\overline{AB}}.\overline{\overline{A}}.\overline{AB} && \text{[Changed + to . and complemented individual expressions]}\\ &= AB.A.\overline{AB} && [\because \overline{\overline{AB}}=AB \text{ and } \overline{\overline{A}}=A]\\ &= AB.\overline{AB}.A\\ &= 0.A && [AB.\overline{AB}=0]\\ &= 0 && [\because 0.A=0]\end{aligned}$$

Do you know that you can find the complement of a Boolean expression using its Dual also ? It is very simple :

1. Write the dual form of given Boolean expression.
2. Swap variables and complemented variables.

For example, to get the complement of expression $(A + BC' + DA')$

1. Write its Dual : $A.(B+C').(D+A')$
2. Replace variables with their complemented form : $A'.(B'+C).(D'+A)$ (the result)

Let us find the complement using the DeMoragan Laws and compare the result.

$$F = (A + BC' + DA')$$
$$F = (A + BC' + DA')' = (A'.(BC')'.(DA')')$$
$$= A'.(B'+C'').(D'+A'') = A'.(B'+C).(D'+A)$$

It is the same as the complement determined using Duality principle.

3.9 More about Logic Gates

We have covered three basic logic gates NOT, OR, AND so far. But there are some more logic gates also which are derived from three basic gates (*i.e.*, AND, OR and NOT). These gates are more popular than NOT, OR and AND and are widely used in industry. This section introduces NOR, NAND, XOR, XNOR gates.

NOR GATE

The Nor Gate has two or more input signals but only one output signal. If all the inputs are 0 (*i.e.*, low), then the output signal is 1 (high).

3.9.1 NOR Gate

The **Nor Gate** has two or more input signals but only one output signal. If all the inputs are 0 (*i.e.*, low), then the output signal is 1 (*high*). If either of the two inputs is 1 (*high*), the output will be 0 (*low*). NOR *gate is nothing but inverted* OR *gate.*

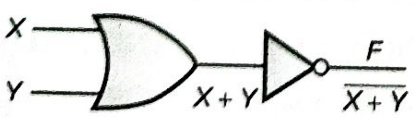
The NOR gate can have as many inputs (2 or more inputs) as desired. No matter how many inputs are there, the action of NOR gate is the same *i.e.*, All 0 (*low*) inputs produce output as 1.

Following truth Tables (3.29 and 3.30) illustrate NOR action.

Table 3.29 *2-input NOR gate*

X	Y	F
0	0	1
0	1	0
1	0	0
1	1	0

$F = \overline{X + Y}$



(a)

Table 3.30 *3-input NOR gate*

X	Y	Z	F
0	0	0	1
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	0

$F = \overline{X + Y + Z}$

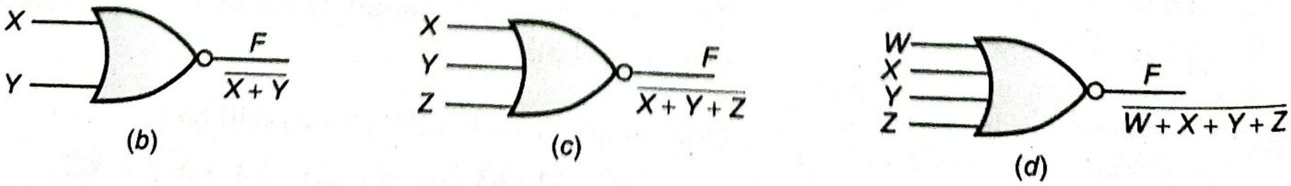


Figure 3.7 (*a*) Logical meaning of NOR gate (*b*) 2 input NOT gate (*c*) 3 input NOR gate (*d*) 4 input NOR gate

3.9.2 NAND Gate

The **NAND Gate** has two or more input signals but only one output signal. If all of the inputs are 1 (high), then the output produced is 0 (low).

NAND *gate is inverted* AND *gate.* Thus, for all 1 (*high*) inputs, it produces 0 (*low*) output, otherwise for any other input combination, it produces a 1 (*high*) output. NAND gate can also have as many inputs as desired.

NAND GATE

The NAND Gate has two or more input signals but only one output signal. If all of the inputs are 1 (high), then the output produced is 0 (low).

NAND action is illustrated in following Truth Tables (3.31 and 3.32).

Table 3.31 *2-input NAND gate*

X	Y	F
0	0	1
0	1	1
1	0	1
1	1	0

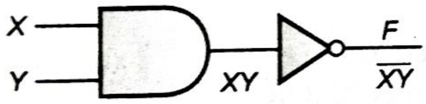
$F = \overline{XY}$

Table 3.32 *3-input NAND gate*

X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$F = \overline{XYZ}$

The logical meaning of NAND gate can be shown as follows :



The symbols of 2, 3, 4 input NAND gates are given below :

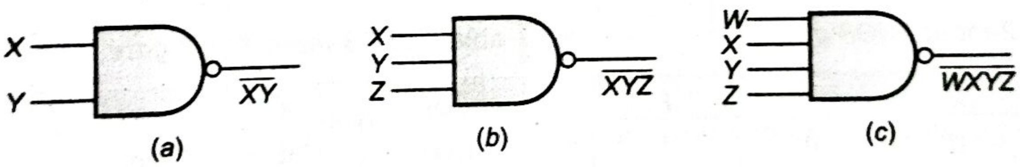


Figure 3.8 (*a*) 2-input NAND gate (*b*) 3-input NAND gate (*c*) 4-input NAND gate.

3.9.3 XOR Gate (Exclusive OR Gate)

The **XOR Gate** can also have two or more inputs but produces one output signal. Exclusive-OR gate is different from OR gate. OR gate produces output 1 for any input combination having one or more 1's, but XOR gate produces output 1 for only those input combinations that have odd number of 1's.

In boolean algebra ⊕ sign stands for XOR operation. Thus *A* XOR *B* can be written as $A \oplus B$.

Following Truth Tables (3.33 and 3.34) illustrate XOR operation.

XOR GATE

XOR gate produces output 1 for only those input combinations that have odd number of 1's.

Table 3.33 *2-input XOR gate*

No. of 1's even/odd	X	Y	F
Even	0	0	0
Odd	0	1	1
Odd	1	0	1
Even	1	1	0

Table 3.34 *3-input XOR gate*

No. of 1's	X	Y	Z	F
Even	0	0	0	0
Odd	0	0	1	1
Odd	0	1	0	1
Even	0	1	1	0
Odd	1	0	0	1
Even	1	0	1	0
Even	1	1	0	0
Odd	1	1	1	1

The symbols of XOR gates are given below :

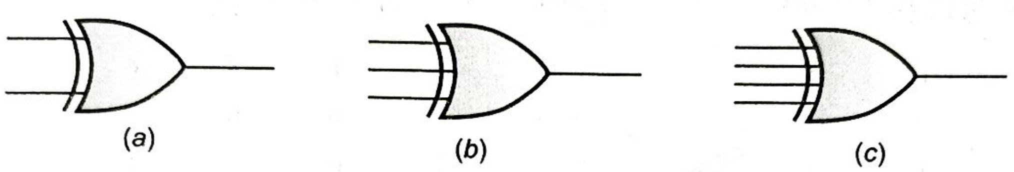


Figure 3.9 (*a*) 2-input XOR gate (*b*) 3-input XOR gate (*c*) 4-input XOR gate.

XOR addition can be summarised as follows :

$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 0 = 1, \quad 1 \oplus 1 = 0$

NOTE

Remember odd number of 1's produce output 1.

3.9.4 XNOR Gate (Exclusive NOR gate)

The **XNOR Gate** is logically equivalent to an inverted XOR *i.e.*, XOR gate followed by a NOT gate (inventor). Thus XNOR produces 1 (high) output when the input combination has even number of 1's. Following tables (3.35 and 3.36) illustrate XNOR action.

XNOR GATE

XNOR gate produces output 1 for only those input combinations that have even number of 1's.

Table 3.35 *2-input XNOR gate*

No. of 1's	X	Y	F
Even	0	0	1
Odd	0	1	0
Odd	1	0	0
Even	1	1	1

Table 3.36 *3-input XNOR gate*

No. of 1's	X	Y	Z	F
Even	0	0	0	1
Odd	0	0	1	0
Odd	0	1	0	0
Even	0	1	1	1
Odd	1	0	0	0
Even	1	0	1	1
Even	1	1	0	1
Odd	1	1	1	0

NOTE

The XNOR Gate is logically equivalent to an inverted XOR *i.e.*, XOR gate followed by a NOT gate (inventor).

Following are the XNOR gate symbols :

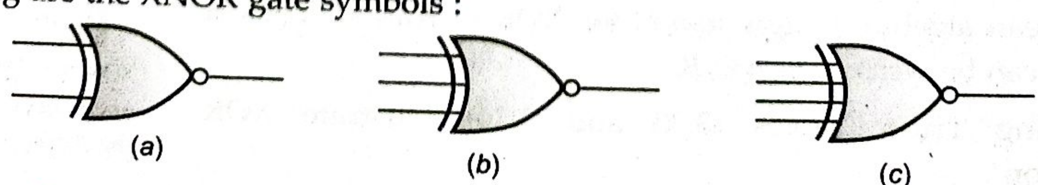


Figure 3.10 (*a*) 2-input XNOR gate (*b*) 3-input XNOR gate (*c*) 4-input XNOR gate.

The bubble (small circle), on the outputs of NAND, NOR, XNOR gates represents complementation.

Now that we are familiar with logic gates, we can use them in designing logic circuits.

3.10 Logic Circuits

LOGIC CIRCUIT

A logic circuit is a circuit that carries out a set of logic actions based on an expression.

A logic circuit is a circuit that carries out a set of logic actions based on an expression. To execute a Boolean expression, you require a logic circuit and the input values for the variables of Boolean expression.

You can represent a Boolean expression in the form of a logic ***circuit using a combination of logic gates*** so that the output of the Boolean expression can be determined for various combinations of input values. Logic circuit diagrams are used to represent Boolean expressions through the combination of logic gates.

The rules to create a logic circuit are :

1. Break the Boolean expression in smaller sub expressions,

   e.g.,

   for a Boolean expression : $AB + \overline{B}CD$, there are two sub-expressions : AB and $\overline{B}CD$.

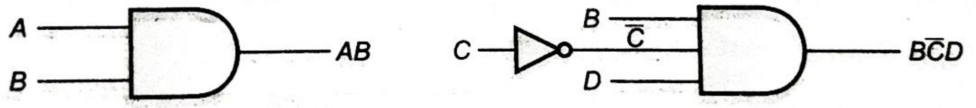
2. For each sub-expression, determine the logic gates that can implement them. (*Refer to Example 3.5 to recall this*), *e.g.,*

   for sub-expressions determined in previous step,

   AB : requires ***AND*** gate

   $B\overline{C}D$: requires ***NOT*** and ***AND*** gates

3. Implement the sub-expressions using the gates determined in previous steps :

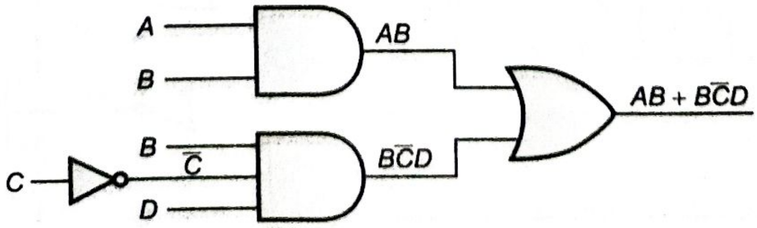


4. Determine the Logic gate for the symbol joining the sub-expressions, *e.g.,*

   for the given expression,

   symbol + joins the sub-expressions (AB and $B\overline{C}D$) ; the symbol + signifies **OR gate.**

5. Using the logic gate for the joining symbol (from step 4), connect the sub-expressions' gate implementation (from step 3)



6. Yay! Our logic circuit diagram is ready.

Let us consider some examples now.

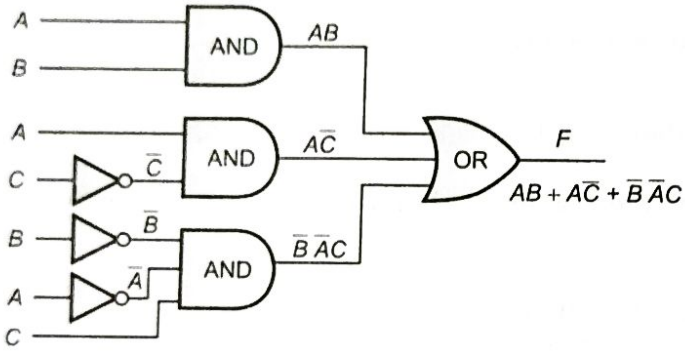
EXAMPLE 14 *Design a circuit to realise the following :* $F(a, b, c) = AB + A\overline{C} + \overline{B}\,\overline{A}C$.

SOLUTION The given boolean expression can also be written as follows :

$$F(a, b, c) = A.B + A.\overline{C} + \overline{B}.\overline{A}.C$$

or $F(a, b, c)$ = (A AND B) OR
(A AND (NOT C)) OR
((NOT B) AND (NOT B) AND C)

Now these logical operators can easily be implemented in form of logic gates. Thus, the circuit diagram for the above expression will be as follows :



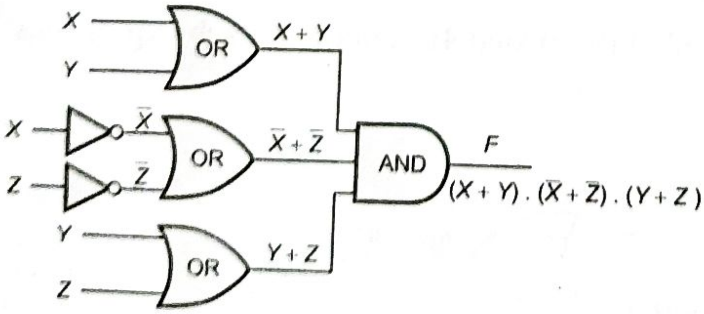
EXAMPLE 15 *Draw the diagram of digital circuit for the function :*

$$F(X, Y, Z) = (X + Y).(\overline{X} + \overline{Z}).(Y + Z).$$

SOLUTION Above expression can also be written as :

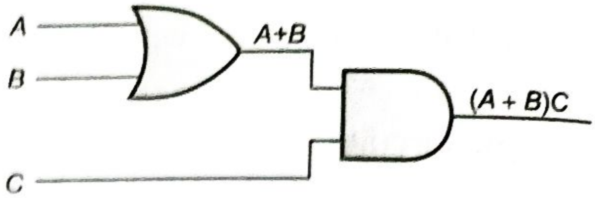
$F(X, Y, Z)$ = (X OR Y) AND ((NOT X) OR (NOT Z)) AND (Y OR Z)

Thus circuit diagram will be



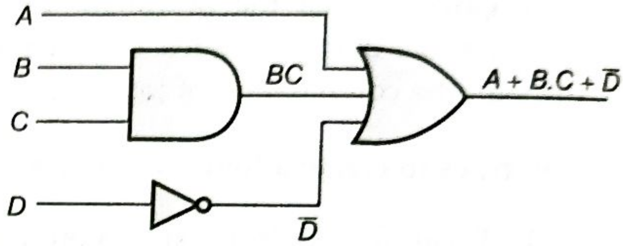
EXAMPLE 16 *Design a logic circuit and draw its diagram for the Boolean expression :* $(A + B)C$.

SOLUTION



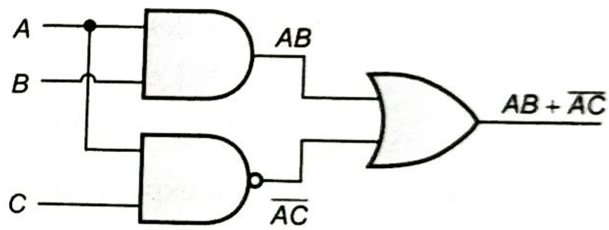
EXAMPLE 17 *Design a logic circuit and draw its diagram for the Boolean expression :* $A + BC + \overline{D}$.

SOLUTION



EXAMPLE 18 *Design a logic circuit and draw its diagram for the Boolean expression :* $AB + \overline{AC}$.

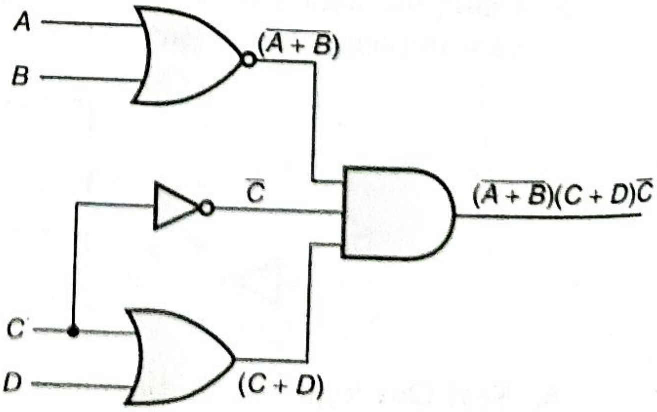
SOLUTION



EXAMPLE 19 *Design a logic circuit and draw its diagram for the Boolean expression :*

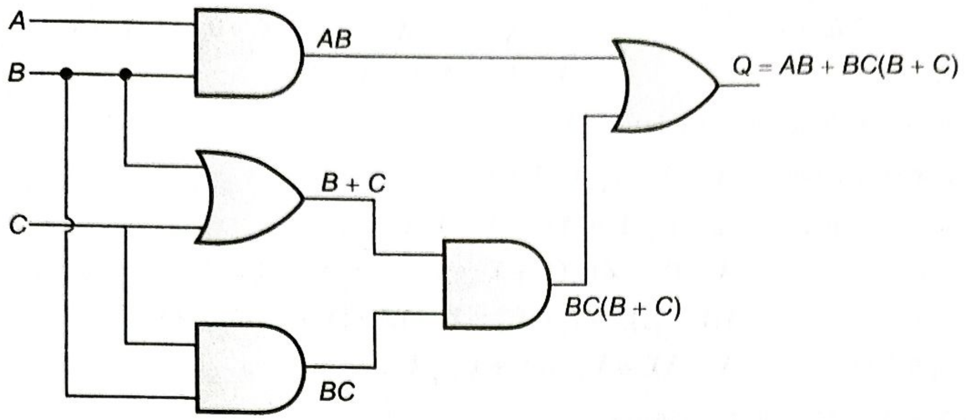
$$(\overline{A + B})(C + D)\overline{C}.$$

SOLUTION



EXAMPLE 20 *Design a logic circuit and draw its diagram for the Boolean expression : $Q = AB + BC(B + C)$.*

SOLUTION



LET US REVISE

- *The decision which results into either YES (TRUE) or NO (FALSE) is called a binary decision.*
- *The statements which can be determined to be True or False are called* ***logical statements*** *or* ***truth functions.***
- *Truth values are TRUE and FALSE or 1 and 0.*
- *Truth table is a table which represents all the possible values of logical variables/ statements along with all the possible results of the given combinations of values.*
- *If result of any logical statement or expression is always TRUE or 1, it is called* ***tautology*** *and if the result is always FALSE or 0 it is called* ***fallacy.***
- *The operation performed by NOT operator is called* ***complementation*** *and the rules of complementation are :*

$$\overline{0} = 1 \text{ and } \overline{1} = 0.$$

- *The operation performed by OR operator is called* ***logical addition*** *and the rules are :*

$$0 + 0 = 0; \quad 0 + 1 = 1; \quad 1 + 0 = 1; \quad 1 + 1 = 1$$

- *The operation performed by AND operator is called logical multiplication and the rules are :*

$$0.0 = 0; \quad 0.1 = 0; \quad 1.0 = 0; \quad 1.1 = 1.$$

- *The NOT gate or inverter is a gate with only one input signal and one output signal; the output state is always the opposite of the input state.*
- *The OR gate has two or more input signals but only one output signal. If any of the input signals is 1 (high), the output signal is 1 (high).*
- *The AND gate can have two or more input signals and produce an output signal. When all the inputs are 1 i.e., high then the output is 1 otherwise output is 0.*
- *Basic Postulates of Boolean Algebra are :*

I *If* $X \neq 0$ *then* $X \neq 1$ *and if* $X \neq 1$ *then* $X = 0$

II OR Relations *(Logical Addition) :*

$$0 + 0 = 0 \; ; \quad 0 + 1 = 1 \; ; \quad 1 + 0 = 1 \; ; \quad 1 + 1 = 1$$

III AND Relations *(Logical Multiplication) :*

$$0.0 = 0 \; ; \quad 0.1 = 0 \; ; \quad 1.0 = 0 \; ; \quad 1.1 = 1$$

IV Complementation Rules : $\overline{0} = 1 \; ; \quad \overline{1} = 0$

Check Point 3.4

1. Why are NAND and NOR gates called Universal gates ?
2. Which gates are called Universal gates and why ?
3. State the purpose of reducing the switching functions to the minimal form.
4. Draw a logic circuit diagram using NAND or NOR only to implement the Boolean function $F(a, b) = a'b' + ab$.
5. What is inverted AND gate called ? What is inverted OR gate called ?
6. When does an XOR gate produce a high output ? When does an XNOR gate produce a high output ?