



DIGITAL ELECTRONICS

BOOLEAN ALGEBRA

Boolean algebra is mathematics of logic. It is one of the most basic tools which is used in the analysis and synthesis of logic circuit. In Boolean algebra, often the variables are represented by capital letters such as A, B, C, X, Y, Z. The Boolean value of a variable is either logic 0 or logic 1. These, 0 and 1, are known as Boolean constants.

Boolean Algebra is used to analyze and simplify the digital (logic) circuits. It uses only the binary numbers i.e. 0 and 1. It is also called as **Binary Algebra** or **logical Algebra**. Boolean algebra was invented by **George Boole** in 1854.

Rule in Boolean Algebra

Following are the important rules used in Boolean algebra.

- Variable used can have only two values. Binary 1 for HIGH and Binary 0 for LOW.
- Complement of a variable is represented by an overbar (-). Thus, complement of variable B is represented as \bar{B} . Thus if B = 0 then $\bar{B} = 1$ and B = 1 then $\bar{B} = 0$.
- ORing of the variables is represented by a plus (+) sign between them. For example ORing of A, B, C is represented as A + B + C.
- Logical ANDing of the two or more variable is represented by writing a dot between them such as A.B.C. Sometime the dot may be omitted like ABC.

Important Boolean Theorems

Following are few important boolean Theorems.

Boolean function/theorems	Description
<u>Boolean Functions</u>	Boolean Functions and Expressions, K-Map and NAND Gates realization
<u>De Morgan's Theorems</u>	De Morgan's Theorem 1 and Theorem 2


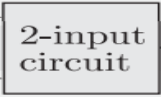
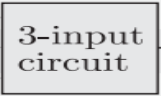
LOGIC LEVELS

Boolean logic variable '0' or '1' is not used to represent actual numbers but it is used to represent the state of voltage variable called logical level. Commonly used representation of logic levels are shown in Table below.

Logic 0	Logic 1
False	True
Open switch	Close switch
Low	High
No	Yes
OFF	ON

TRUTH TABLE

A truth table represents the relation between all inputs and possible outputs of any logic device or logic circuit in a tabular form. The number of inputs may vary from one to many depending upon the device or complexity of the circuit. Number of output also varies in this way and may be one or more. For different digital circuits, some of the examples of truth table are given below.

Input	Output	Input	Output	Input	Output				
A	Y	A	B	Y	A	B	C	Y	
0	1	0	0	0	0	0	0	1	
1	0	0	1	1	0	0	1	0	
		1	0	1	0	1	0	0	
		1	1	0	0	1	1	1	
			1	0	0	1	0	0	1
			1	0	1	0	1	1	0
		1	1	0	1	1	0	1	
		1	1	1	0	1	1	1	0
									

BASIC BOOLEAN OPERATIONS

Boolean algebra uses only three basic operations, namely 1. OR operation 2. AND operation 3. NOT operation

BOOLEAN ADDITION (LOGICAL OR)

The OR operation in Boolean algebra is similar to addition in ordinary algebra i.e., OR means logical addition operation. The logical OR operation on A and B is denoted by

$Y = A + B$, where '+' is the OR operator

The output Y corresponding to various combinations of inputs, A and B,

Input		Output
A	B	$Y = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

BOOLEAN MULTIPLICATION (Logical AND) :

The AND operation in Boolean algebra is similar to multiplication in ordinary algebra i.e, AND performs logical multiplication operation. Let A and B be two Boolean variables. Then, the logical AND operation on A and B is denoted by $Y = A \cdot B$, where \cdot is the AND operator. The output Y corresponding to various combinations of inputs, A and B,

Truth table for AND operation

Input		Output
A	B	$Y = AB$
0	0	0
0	1	0
1	0	0
1	1	1

The minimum number of inputs for AND operation is two. The number of output is always one, irrespective of the number of inputs.

Logical NOT :

NOT is the simplest of the three basic operations of Boolean algebra. It is also known as inversion and complement. The NOT operation is indicated by a bar '-' over the variable. If A is a variable, then NOT of A is expressed as \bar{A} .

Input	Output
A	$Y = \bar{A}$
0	1
1	0

Logical NOT is the only Boolean operation which must be performed with only one operand or one input. Note that in some texts, the NOT operation is also presented as A'.

THEOREMS OF BOOLEAN ALGEBRA

The theorems of Boolean algebra can be used to simplify many complex Boolean expression and also to transform the given expression into a more useful and meaningful equivalent expression. These theorems are discussed as below.

1. Complementation Laws :

The term complement implies to invert, i.e. to change 1's to 0's and 0's to 1's. The five laws of complementation are as follows:

1. The complement of 0 is 1, i.e. $\bar{0} = 1$
2. The complement of 1 is 0, i.e. $\bar{1} = 0$
3. If $A = 0$, then $\bar{A} = 1$
4. If $A = 1$, then $\bar{A} = 0$
5. The double complementation does not change the function, i.e. $\bar{\bar{A}} = A$

2. AND Laws The four AND laws are as follows:

1. Null Law: $A \cdot 0 = 0$
2. Identity Law: $A \cdot 1 = A$
3. $A \cdot A = A$
4. $A \cdot \bar{A} = 0$

3. OR Laws

The four OR laws are as follows:

1. Null Law: $A + 0 = A$
2. Identity Law: $A + 1 = 1$
3. $A + A = A$
4. $A + \bar{A} = 1$

4. Commutative Laws

Commutative law states that the order of the variable in OR and AND operations is not important. The two commutative laws are

$$A + B = B + A$$

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

5 Associative Laws

Associative law states that the grouping of variables in AND or OR expression does not affect the result. There are two associative laws.

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

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

6. Distributive Law

The distributive laws allow factoring or multiplying out of expressions.

There are two distributive laws

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

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

7. Redundant Literal Rule

This law states that ORing of a variable with the AND of the complement of that variable with another variable, is equal to ORing of the two variables, i.e.

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

Another theorem based on this law is

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

8. Idempotent Law

Idempotence means the same value. There are two idempotent laws

$$A \cdot A \cdot A \dots A = A$$

$$A + A + A + \dots + A = A$$

9. Absorption Law

There are two absorption laws

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

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

10. Consensus Theorem

There are two consensus theorems,

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

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

11. Transposition Theorem

There are two transposition theorems, the first is given as

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

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

12. De Morgan's Theorem

De Morgan's theorem gives two of the most powerful laws in Boolean algebra. These theorems are very useful in simplification of Boolean expressions,

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

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

13. Shannon's Expansion Theorem

According to this theorem, any switching expression can be decomposed with respect to a variable A into two parts, one containing A and the other containing \bar{A} . This concept is useful in decomposing complex system into an interconnection of smaller components.

$$f(A, B, C, \dots) = A \cdot f(1, B, C, \dots) + \bar{A} \cdot f(0, B, C, \dots)$$

$$f(A, B, C, \dots) = [A + f(0, B, C, \dots)] \cdot [\bar{A} + f(1, B, C, \dots)]$$

Laws of Boolean Algebra

As well as the logic symbols —0‖ and —1‖ being used to represent a digital input or output, we can also use them as constants for a permanently —Open‖ or —Closed‖ circuit or contact respectively

A set of rules or Laws of Boolean Algebra expressions have been invented to help reduce the number of logic gates needed to perform a particular logic operation resulting in a list of functions or theorems known commonly as the **Laws of Boolean Algebra**.

Boolean Algebra is the mathematics we use to analyse digital gates and circuits. We can use these —Laws of Boolean‖ to both reduce and simplify a complex Boolean expression in an attempt to reduce the number of logic gates required. *Boolean Algebra* is therefore a system of mathematics based on logic that has its own set of rules or laws which are used to define and reduce Boolean expressions.

The variables used in **Boolean Algebra** only have one of two possible values, a logic —0‖ and a logic —1‖ but an expression can have an infinite number of variables all labelled individually to represent inputs to the expression, For example, variables A, B, C etc, giving us a logical expression of $A + B = C$, but each variable can ONLY be a 0 or a 1

