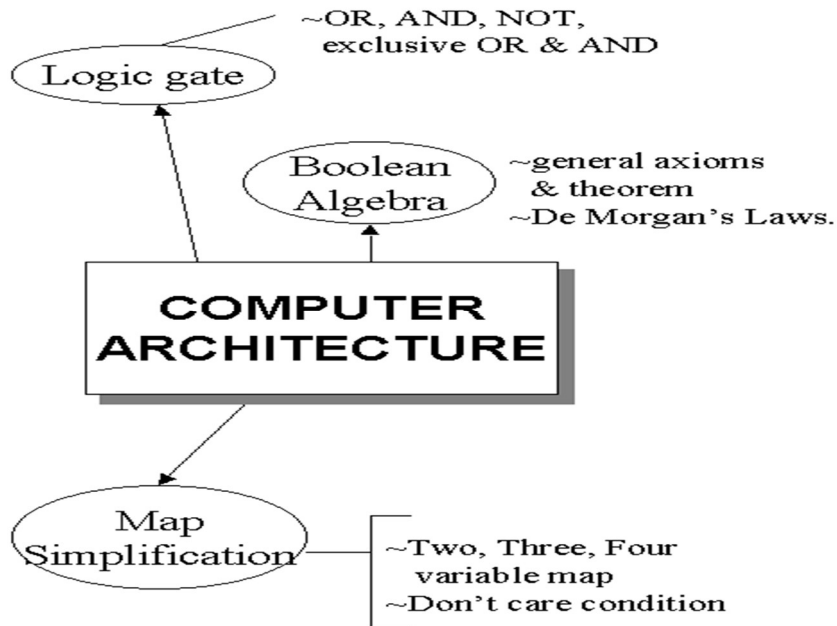


Boolean Algebra



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 $\overline{B}$. Thus if $B = 0$ then $\overline{B} = 1$ and $B = 1$ then $\overline{B} = 0$.
- 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.

Boolean Laws

There are six types of Boolean Laws.

Commutative law

Any binary operation which satisfies the following expression is referred to as commutative operation.

$$(i) A.B = B.A$$

$$(ii) A + B = B + A$$

Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.

Associative law

This law states that the order in which the logic operations are performed is irrelevant as their effect is the same.

$$(i) (A.B).C = A.(B.C) \quad (ii) (A + B) + C = A + (B + C)$$

Distributive law

Distributive law states the following condition.

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

AND law

These laws use the AND operation. Therefore they are called as **AND** laws.

$$(i) A.0 = 0 \quad (ii) A.1 = A \\ (iii) A.A = A \quad (iv) A.\overline{A} = 0$$

OR law

These laws use the OR operation. Therefore they are called as **OR** laws.

$$(i) A + 0 = A \quad (ii) A + 1 = 1 \\ (iii) A + A = A \quad (iv) A + \overline{A} = 1$$

INVERSION law

This law uses the NOT operation. The inversion law states that double inversion of a variable results in the original variable itself.

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

Logic Gate Truth Tables

As well as a standard Boolean Expression, the input and output information of any Logic Gate or circuit can be plotted into a standard table to give a visual representation of the switching function of the system. The table used to represent the boolean expression of a logic gate function is commonly called a Truth Table. A logic gate truth table shows each possible input combination to the gate or circuit with the resultant output depending upon the combination of these input(s).

For example, consider a single **2-input** logic circuit with input variables labelled as A and B. There are “four” possible input combinations or 2^2 of “OFF” and “ON” for the two inputs. However, when dealing with Boolean expressions and especially logic gate truth tables, we do not general use “ON” or “OFF” but instead give them bit values which represent a logic level “1” or a logic level “0” respectively.

Then the four possible combinations of A and B for a 2-input logic gate is given as:

- Input Combination 1. – “OFF” – “OFF” or (0, 0)

- Input Combination 2. – “OFF” – “ON” or (0, 1)
- Input Combination 3. – “ON” – “OFF” or (1, 0)
- Input Combination 4. – “ON” – “ON” or (1, 1)

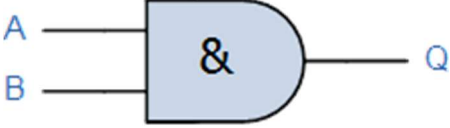
Therefore, a 3-input logic circuit would have 8 possible input combinations or 2^3 and a 4-input logic circuit would have 16 or 2^4 , and so on as the number of inputs increases. Then a logic circuit with “n” number of inputs would have 2^n possible input combinations of both “OFF” and “ON”.

So in order to keep things simple to understand, in this tutorial we will only deal with standard **2-input** type logic gates, but the principals are still the same for gates with more than two inputs.

Then the Truth tables for a 2-input AND Gate, a 2-input OR Gate and a single input NOT Gate are given as:

2-input AND Gate

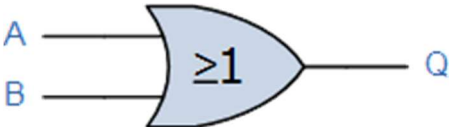
For a 2-input AND gate, the output Q is true if BOTH input A “AND” input B are both true, giving the Boolean Expression of: ($Q = A \text{ and } B$).

Symbol	Truth Table		
 2-input AND Gate	A	B	Q
	0	0	0
	0	1	0
	1	0	0
	1	1	1
Boolean Expression $Q = A.B$	Read as A AND B gives Q		

Note that the Boolean Expression for a two input AND gate can be written as: $A.B$ or just simply AB without the decimal point.

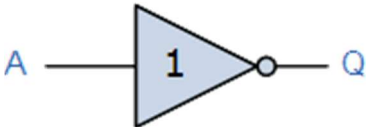
2-input OR (Inclusive OR) Gate

For a 2-input OR gate, the output Q is true if EITHER input A “OR” input B is true, giving the Boolean Expression of: ($Q = A \text{ or } B$).

Symbol	Truth Table		
 2-input OR Gate	A	B	Q
	0	0	0
	0	1	1
	1	0	1
	1	1	1
Boolean Expression $Q = A+B$	Read as A OR B gives Q		

NOT Gate

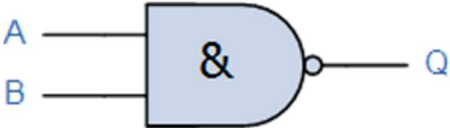
For a single input NOT gate, the output Q is ONLY true when the input is “NOT” true, the output is the inverse or complement of the input giving the Boolean Expression of: ($Q = \text{NOT } A$).

Symbol	Truth Table	
 Inverter or NOT Gate	A	Q
	0	1
	1	0
Boolean Expression $Q = \text{NOT } A$ or A		Read as inversion of A gives Q

The NAND and the NOR Gates are a combination of the AND and OR Gates with that of a NOT Gate or inverter.

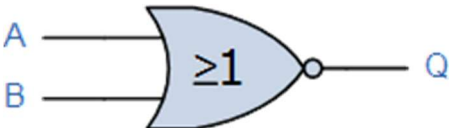
2-input NAND (Not AND) Gate

For a 2-input NAND gate, the output Q is true if BOTH input A and input B are NOT true, giving the Boolean Expression of: ($Q = \text{not}(A \text{ and } B)$).

Symbol	Truth Table		
 2-input NAND Gate	A	B	Q
	0	0	1
	0	1	1
	1	0	1
	1	1	0
Boolean Expression $Q = A . B$		Read as A AND B gives NOT-Q	

2-input NOR (Not OR) Gate

For a 2-input NOR gate, the output Q is true if BOTH input A and input B are NOT true, giving the Boolean Expression of: ($Q = \text{not}(A \text{ or } B)$).

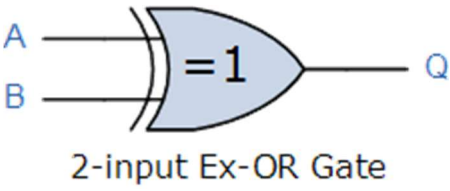
Symbol	Truth Table		
 2-input NOR Gate	A	B	Q
	0	0	1
	0	1	0
	1	0	0
	1	1	0
Boolean Expression $Q = A + B$		Read as A OR B gives NOT-Q	

As well as the standard logic gates there are also two special types of logic gate function called an Exclusive-OR Gate and an Exclusive-NOR Gate. The actions of both of these types of gates can be made

using the above standard gates however, as they are widely used functions, they are now available in standard IC form and have been included here as reference.

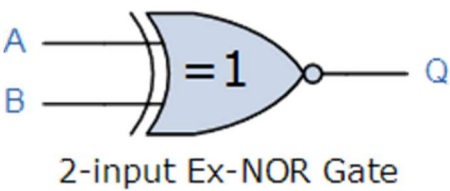
2-input EX-OR (Exclusive OR) Gate

For a 2-input Ex-OR gate, the output Q is true if EITHER input A or if input B is true, but NOT both giving the Boolean Expression of: ($Q = (A \text{ and NOT } B) \text{ or } (\text{NOT } A \text{ and } B)$).

Symbol	Truth Table		
 2-input Ex-OR Gate	A	B	Q
	0	0	0
	0	1	1
	1	0	1
	1	1	0
Boolean Expression $Q = A \oplus B$			

2-input EX-NOR (Exclusive NOR) Gate

For a 2-input Ex-NOR gate, the output Q is true if BOTH input A and input B are the same, either true or false, giving the Boolean Expression of: ($Q = (A \text{ and } B) \text{ or } (\text{NOT } A \text{ and } \text{NOT } B)$).

Symbol	Truth Table		
 2-input Ex-NOR Gate	A	B	Q
	0	0	1
	0	1	0
	1	0	0
	1	1	1
Boolean Expression $Q = A \odot B$			

Summary of 2-input Logic Gates

The following Truth Table compares the logical functions of the 2-input logic gates above.

Inputs		Truth Table Outputs For Each Gate					
A	B	AND	NAND	OR	NOR	EX-OR	EX-NOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

The following table gives a list of the common logic functions and their equivalent Boolean notation.

Logic Function	Boolean Notation
AND	$A.B$
OR	$A+B$
NOT	$\bar{A}$
NAND	$\overline{A.B}$
NOR	$\overline{A+B}$
EX-OR	$(A.B) + (\bar{A}.\bar{B})$ or $A \oplus B$
EX-NOR	$(A.B) + (\bar{A}.\bar{B})$ or $A \odot B$

2-input logic gate truth tables are given here as examples of the operation of each logic function, but there are many more logic gates with 3, 4 even 8 individual inputs. The multiple input gates are no different to the simple 2-input gates above, So a 4-input AND gate would still require ALL 4-inputs to be present to produce the required output at Q and its larger truth table would reflect that.