INTRODUCTION

It is difficult to imagine a world without digital computers nowadays. Computers are used in every spheres of life. We usually communicate each other with the instantaneous electronic mails (emails), all the transactions in the banks, hospitals, super markets, airlines, etc. are done with the help of digital computers. Computer is just one of the outcomes of the progress in digital electronics. It can be noted that even in large digital systems like computers, there are only few basic operations, which are repeated many times. The four most commonly used circuits in such systems are OR, AND and NOT Logic gates and FLIP-FLOP circuits. These Logic gates and circuits implement Boolean algebra, which was invented by G. Boole in mid 19th century. Hence it becomes a necessity for the today's engineers to learn these mathematical equations for logical circuits along with the logic circuits.

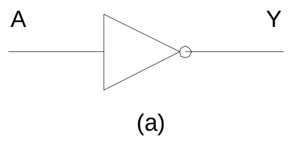
LOGIC GATES AND BOOLEAN ALGEBRA

Digital computers understand the language of 1s and 0s. This number system is also called as binary number system (bi means two). Note that the operation of digital circuits can be described in two corresponding voltage levels. The more positive level is denoted by high ($H = 1$) and the other is denoted by low ($L = 0$). And the logical operations are represented by true (T) and false (F). For instance, $H = 1 = T$ and $L = 0 = F$ is a positive logic whereas $H = 0 = F$ and $L = 1 = T$ is a negative logic.

A digital circuit having one or more input signals but only one output signal is called a gate. A gate, which implements Boolean algebraic equations, is called Logic gates. The most basic gates are NOT gate, OR gate and AND gate, we will discuss them one by one along with the Boolean identities pertaining to their logical operations.

8.2.1 NOT Gate

The NOT gate has a single input and a single output and its symbol and truth table (truth table contains a table of all possible input values and their corresponding outputs values) is shown in Figure 8.1. It performs the operation of inversion, i.e. the output of a NOT gate takes a 1 (high) state iff the input takes the 0 (low) state and vice-versa. A circuit, which performs a logic negation, is called a NOT circuit or inverter since it inverts the output with respect to the input.



A	Y
0	1
1	0

(b)

Figure 8.1 : (a) Symbol of NOT Gate, and (b) Truth Table

OR Gate

The OR gate has two or more inputs and a single output. The symbol and truth table for a two-input OR gate is shown in Figure 8.2. The output of an OR gate is in 1 state if any of the inputs is in the 1 state.

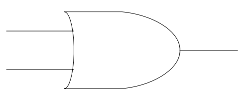


Figure 8.2 : (a) Symbol of OR Gate, and (b) Truth Table

AND Gate

The AND gate has two or more inputs and a single output. The symbol and truth table for a two-input AND is shown in Figure 8.3. The output of an AND gate is in 1 state iff all inputs are in the 1 state or the output will be zero if any of the inputs is zero.

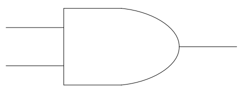


Figure 8.3 : (a) Symbol of AND Gate, and (b) Truth Table

A

The mathematics of binary number systems (1s and 0s) also known as Boolean Algebra was invented by George Boole in 1854. This branch of mathematics has become very important with the invention of digital computers in 1946. In this section, we will discuss some of the basic Boolean laws and theorems.

Boolean Identities for OR Operation

The OR operation in the Boolean algebra is denoted by (+). Following are the Boolean identities for the OR operation.

***A* + *B* = *B* + *A* (Commutative Law)**

It implies that the input *A* and *B* of the OR gate can be interchanged without changing the output *Y*. It can be justified from the truth table of two-input OR gate. For *A* = *B*, it is obvious that it doesn't matter when we interchange *A* and *B*. When *A* = 0 and *B* = 1, if we interchange *A* and *B*, then it will become the case of *A* = 1 and *B* = 0 and for both these case the output is 1. Hence it doesn't matter to the output if we interchange *A* and *B* inputs.

***A* + *B* + *C* = (*A* + *B*) + *C* = *A* + (*B* + *C*) (Associative Law)**

It means that the order of combining the input variables has no effect on the output variables. This can be verified from the truth table for three-input OR gate.

A* + *A* = *A

It means that any variable ORed with itself equals the variable. We can justify this Boolean identity by substituting the two possible values of *A*. For *A* = 0, 0 + 0 = 0 and for *A* = 1, 1 + 1 = 1 is true (refer to truth table for two-input OR gate).

***A* + 1 = 1**

If one input of the OR gate is high the output is high no matter what is the other input. For *A* = 0, 0 + 1 = 1 and for *A* = 1, 1 + 1 = 1 is true.

A* + 0 = *A

It means a Boolean variable ORed with 0 equals the variable. For $A = 0$, $0 + 0 = 0$ and $A = 1$, $1 + 0 = 1$ is true.

Boolean Identities for AND Operation

The AND operation is denoted by (.) in the Boolean algebra. Following are the Boolean identities pertaining to AND operation.

$A \cdot B = B \cdot A$ (Commutative Law)

It implies that the inputs of the AND gate can be interchanged without changing the output Y which can be justified from the truth table of two-input AND gate. Note that (.) is suppressed many times and we can write the Commutative law as $A\,B = B\,A$.

$A\,B\,C = (A\,B)\,C = A\,(B\,C)$ (Associative Law)

It means that the order of combining the input variables has no effect on the output variables. This can be verified from the truth table for three-input AND gate.

$A\,A = A$

It means that any variable ANDed with itself equals the variable. For $A = 0$, $0 \cdot 0 = 0$ and for $A = 1$, $1 \cdot 1 = 1$ is true (refer to truth table for two-input AND gate).

$A \cdot 1 = A$

If one input of the AND gate is high the output is equal to the input. For $A = 0$, $0.1 = 0$ and for $A = 1$, $1.1 = 1$ is true.

$A \cdot 0 = 0$

If one input of the AND gate is low the output is low irrespective of the other input. For $A = 0$, $0.0 = 0$ and for $A = 1$, $1.0 = 0$ is true.

$A\,(B + C) = A\,B + A\,C$ (Distributive Law)

We can write down the truth tables for LHS and RHS of the Boolean equation and verify they are same.

Boolean Identities for NOT Operation

The NOT operation of a Boolean variable A is denoted by its complement $\bar{A}$. Following are the Boolean identities pertaining to NOT operation.

$\bar{\bar{A}} = A$

It implies that the double complement of a Boolean variable is the variable itself. For $A = 0$, $\bar{A} = 1$, $\bar{\bar{A}} = 0$ and for $A = 1$, $\bar{A} = 0$, $\bar{\bar{A}} = 1$ is true.

$\bar{A} + A = 1$

It means that a variable ORed with its complement always equals 1. If $A = 0$, $\bar{A} = 1$ and $A + \bar{A} = 1$ and when $A = 1$, $\bar{A} = 0$ and $A + \bar{A} = 1$ is correct.

$A\,\bar{A} = 0$

It means that a variable ANDed with its complement always equals 0. For two possible values of A, $0.1 = 0$ when $A = 0$, $1.0 = 0$ when $A = 1$ is true.

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

Proof : $A + \bar{A}\,B = A\,(B + 1) + \bar{A}\,B = A\,B + A + \bar{A}\,B = (A + \bar{A})\,B + A = B + A$
$= A + B$

De Morgan’s Theorems

De Morgan’s First Theorem

It says that complement of sum equals the product of complements.

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

The LHS of the equation describes a NOR (NOT-OR) gate and the RHS of the equation describes an AND gate with inverted inputs and both the equations have the same truth table as shown in Figure 8.4.

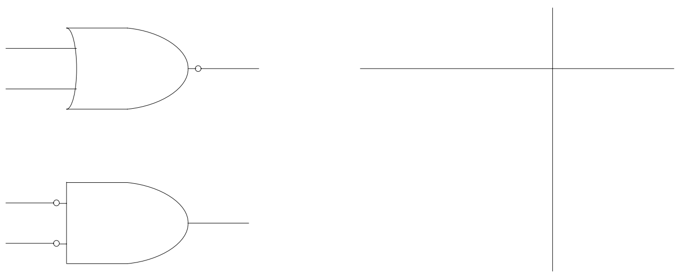


Figure 8.4 : (a) NOR Logic Gate, (b) AND Gate with Inverted Inputs, and (c) Truth Tble

De Morgan’s Second Theorem

It says that complement of product equals the sum of complements.

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

The LHS of the equation describes a NAND (NOT-AND) gate and the RHS of the equation describes an OR gate with inverted inputs and both the equations have the same truth table as shown in Figure 8.5.

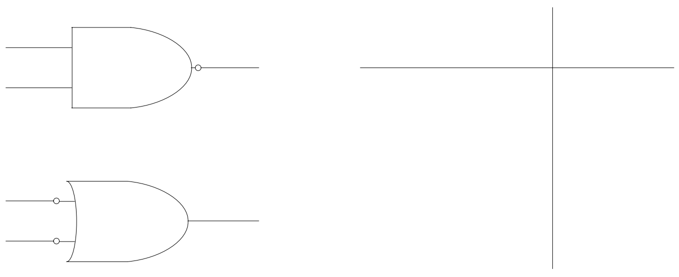


Figure 8.5 : (a) NAND Logic Gate, (b) OR Gate with Inverted Inputs, and (c) Truth Table

(a) Show that $A + A\ B = A$.

(b) Verify that $A + B\ C = (A + B)\ (A + C)$.