# Chapter Three

## 3.1 Introduction:

The Boolean algebra was created by the English mathematician George Boole (1815-1864). The Boolean algebra codifies rules of relationship between mathematical quantities to one of two possible values: true or false, 1 or 0. So, all arithmetic operations performed with Boolean quantities have but one of two possible outcomes: either 1 or 0. There are three basic Boolean arithmetic operations:

- Boolean addition which is equivalent to the OR logic function, as well as parallel switch contacts;
- Boolean multiplication, which is equivalent to the AND function as well as series switch contacts;
- Boolean complementation which is equivalent to the NOT logic function.

## 3.2 Boolean arithmetic:

This section presents the basic relationship concerning the three basic Boolean arithmetic operations.

### 3.2.1 Boolean addition:

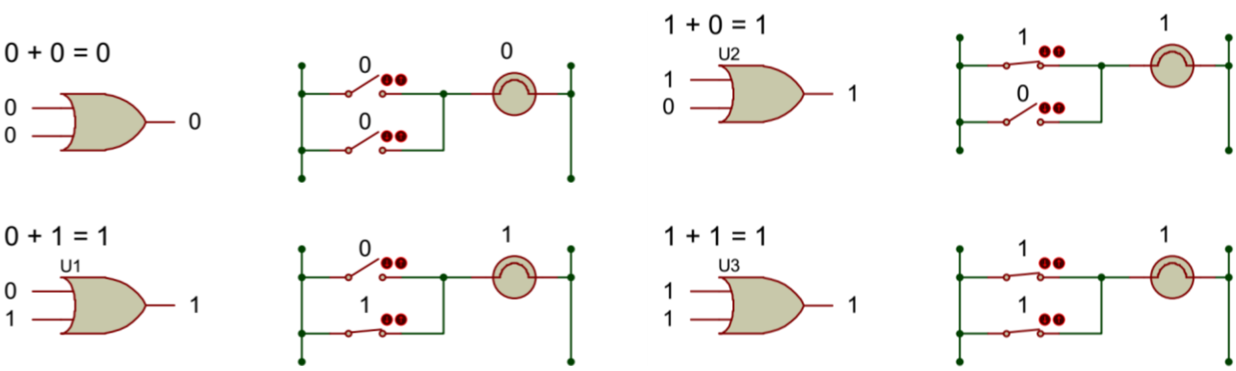
As we have already said, Boolean addition is equivalent to the OR logic function. Therefore, we have the following relationships:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 1




### Remark 3.1:

There is a difference between Boolean addition and binary addition; for binary addition we have the following relationships.

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10 (1 + 1 = 0 + report of 1).

### 3.2.2 Boolean multiplication:

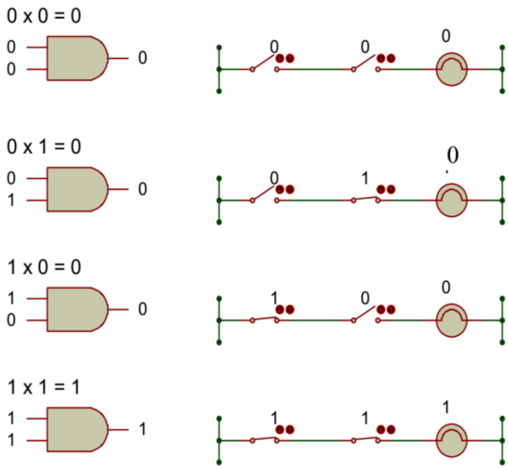
The Boolean multiplication is equivalent to the AND logic function:

0 x 0 = 0

0 x 1 = 0

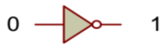
1 x 0 = 0

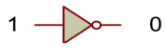
1 x 1 = 1




### 3.2.3 Boolean complementation:

The Boolean complementation is equivalent to the NOT logic function.

/0 = 1 


/1 = 0 


### 3.3 Boolean algebraic identities:

An identity is a statement that is true for all possible values of its variables. There are two groups of Boolean algebraic identities: additive identities and multiplicative identities.

### 3.3.1 Additive identities

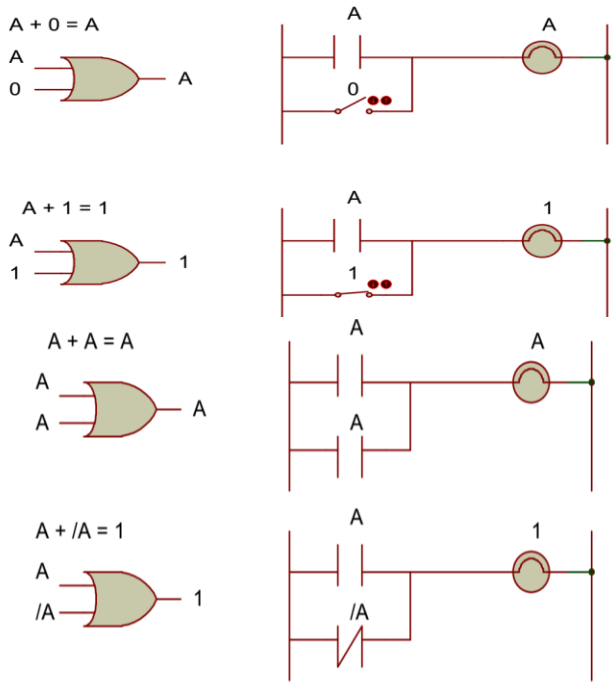
If A is a Boolean variable, then the following statements are always true.

A + 0 = A

A + 1 = 1

A + A = A

A + /A = 1




### 3.3.2 Multiplicative identities:

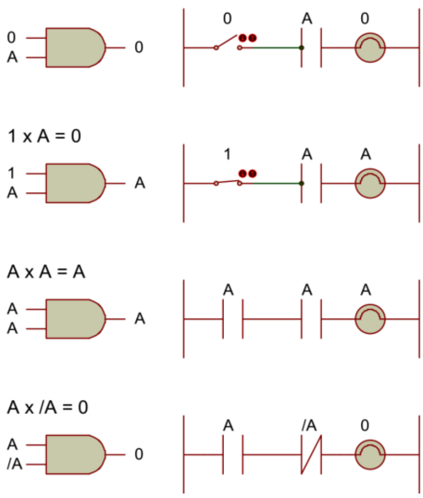
A being a Boolean variable, the following statements are always true.

0 x A = 0

1 x A = A

A x A = A

A x/A = 0




### Remark 3.2: Double complementation

Complementing a variable twice results in the original Boolean value.


### 3.4 Boolean algebraic properties:

Let us consider three Boolean Variables A, B and C. The following properties are true.

- **Commutative property:**

- Addition:

$A + B = B + A$

- Multiplication:

$A \times B = B \times A$

- **Associative property:**

- Addition:

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

- Multiplication:

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

- **Distributive property:**

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

### 3.5 Boolean rules for simplification:

There are several rules for Boolean algebra intended to be used in reducing complex Boolean expressions to their simplest forms. The simplification of the Boolean expressions of logic circuits brings many advantages:

- Higher operating speed (less delay time from input signal transition to output signal transition).
- Less power consumption (few IC used).
- Less cost.
- Greater reliability.

### 3.6 Boolean rules for simplification:

There are several rules for Boolean algebra intended to be used in reducing complex Boolean expressions to their simplest forms. The simplification of the Boolean expressions of logic circuits brings many advantages:

- Higher operating speed (less delay time from input signal transition to output signal transition).
- Less power consumption (few IC used).
- Less cost.
- Greater reliability.

#### 3.5.1 Rule 1: $A + AB = A$

$A + AB = A(1 + B)$

= A (1)

= A

### *3.5.2* Rule 2: $A+\bar{A}B=A+B$

$A+\bar{A}B=A+AB+\bar{A}B$ (Apply the previous rule to expand A term to A + AB)

$=A+B\left(A+\bar{A}\right)$ (Factorising B)

$=A+B(1)$ (Applying identity $A+\bar{A}=1$)

$=A+B$

### 3.5.3 Rule 3: $(A+B)(A+C)=A+BC$

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

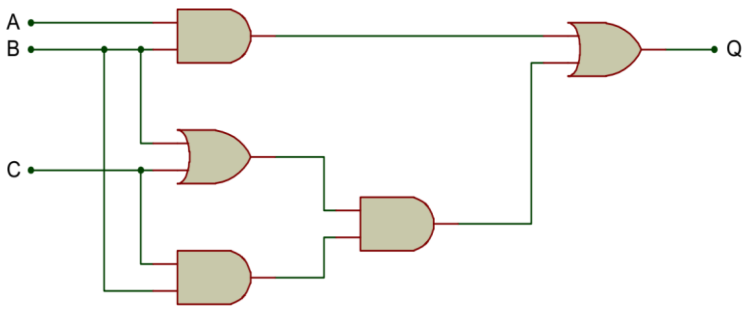
$=A+AC+AB+BC$ (Applying identity AA = A)

$=A+AB+BC$ (Applying A + AC = A)

$=A+BC$ (Applying A + AB = A)

## 3.6 Circuit simplification example:

Let us consider the following logic circuit.




1. Write the Boolean expression of the output Q:

$Q=AB+BC(B+C))$

2. Reduce this expression to its simplest form using the rules of Boolean algebra.

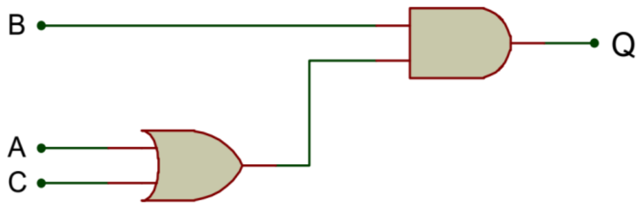
$Q=AB+BCB+BCC$

$=AB+BC+BC$ (Using the identity A.A =A)

$=AB+BC$ (Identity A.A = A)

$Q=B(A+C)$

3. Generate the schematic diagram of the simplest expression




### Remark 3.3

To convert Boolean expression to a gate circuit, you should evaluate the expression using standard order of operation:

- Multiplication before addition,
- Operation within parenthesis before anything else.

### Exercise 3.1:

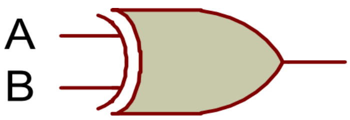
Simplify the following expressions using Boolean algebra and generate the schematic diagrams of the simplest expressions.

$$X = \overline{A}.\overline{B}.\overline{C} + \overline{A}.BC + ABC + A\overline{B}.\overline{C} + A.\overline{B}.C$$

$$Y = \left(B + \overline{C}\right)\left(\overline{B} + C\right) + \overline{\overline{A} + B + \overline{C}}$$

$$Z = \left(\overline{C + D}\right) + \overline{A}C\overline{D} + A\overline{B}.\overline{C} + \overline{A}..\overline{B}CD + A.C.\overline{D}$$

## 3.7 The exclusive-OR function




$A \oplus B$

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

## 3.8 De Morgan's theorem

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

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

De Morgan's theorem may be thought in terms of breaking a long bar symbol. When a long bar is broken, the operation directly implies the changes from addition to multiplication or vice versa, and the broken bar pieces remains over the individual variables.

### Remark 3.4:

When multiple layers of bar exist in an expression, you may only break one bar at a time.

**Example 3.1:**

Let us simplify the following expressions:

$\overline{A+\overline{BC}} = \overline{A}.\overline{\overline{BC}}$ (The superior bar broken)

$= \overline{A}.BC$

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

$= \overline{A}.BC$

## 3.9 Converting truth table into Boolean expression:

We can convert truth table into Boolean expression using one of the following methods:

- Sum of products (SOP)
- Product of sums (POS)

### 3.9.1 Sum of products:

Boolean expressions may be generated from truth table quite easily using the following steps:

- Determine which rows of the table have an output of 1;
- Write one product for each raw;
- Sum all the product terms.

This creates a Boolean expression representing the truth table as a whole.

**Example 3.2:**

Let's consider a logic circuit having the following truth table:

| A | B | C | Q | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Row 1 |
| 0 | 0 | 1 | 0 | Row 2 |
| 0 | 1 | 0 | 0 | Row 3 |
| 0 | 1 | 1 | 1 | Row 4 |
| 1 | 0 | 0 | 0 | Row 5 |
| 1 | 0 | 1 | 1 | Row 6 |
| 1 | 1 | 0 | 1 | Row 7 |
| 1 | 1 | 1 | 1 | Row 8 |

The rows 4, 6, 7 and 8 have an output of 1, each raw gives us a product. By summing those products, we obtain the following Boolean expression which is that of the output Q.

$Q = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$

**Exercise 3.2:**

Simplify the expression of the output Q treated in the example above using Boolean algebra and generate the schematic diagram of the simplest expression.

### 3.9.2 Product of sums:

Boolean expression may be also generated from truth table quite easily by determining which rows of the table have an output of 0, writing one sum term for each row and finally multiplying all the terms.

**Example 3.3:**

Let us consider a logic circuit having the following truth table.

| A | B | C | Q | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Row 1 |
| 0 | 0 | 1 | 1 | Row 2 |
| 0 | 1 | 0 | 1 | Row 3 |
| 0 | 1 | 1 | 1 | Row 4 |
| 1 | 0 | 0 | 1 | Row 5 |
| 1 | 0 | 1 | 1 | Row 6 |
| 1 | 1 | 0 | 1 | Row 7 |
| 1 | 1 | 1 | 0 | Row 8 |

The rows 1 and 8 have an output of 0; each row gives us a sum. The product of those sums gives us a Boolean expression which is that of the output of the logic circuit. In fact, we have:

$$\overline{Q} = \overline{A}.\overline{B}.\overline{C} + ABC$$

$$\overline{\overline{Q}} = Q$$

$$= \overline{\left(\overline{A}..\overline{B}..\overline{C}\right) + ABC}$$

$$= \overline{\left(\overline{A}..\overline{B}..\overline{C}\right)}.\overline{A.B.C}$$

$$= \left(\overline{\overline{A}} + \overline{\overline{B}} + \overline{\overline{C}}\right)\left(\overline{A} + \overline{B} + \overline{C}\right)$$

$$= (A + B + C)\left(\overline{A} + \overline{B} + \overline{C}\right)$$

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

In reality for each row having an output of 0, we should notice that we have but the

inverted output product ( $\overline{Q}$ ). By inverting that output ( $\overline{\overline{Q}}$ ), we obtain a sum using De Morgan’s theorem. Finally, the product of all those sums gives us the output of the logic circuit.

**Remark:**

Generally, the sum of products is more used than the product of sums to convert a truth table into Boolean expression. However, when a few number of rows have an output of 0, it is preferable to use the POS than to use the SOP.

**Exercise 3.3:**

Generate the logic diagram of the circuit treated the example 3.3.

**Exercise 3.4:**

Assuming that $A \oplus B = \overline{A}.B + A.\overline{B}$ , proof that $\overline{A \oplus B} = A.B + \overline{A}..\overline{B}$