Chapter 04 :
Part 01:
Boolean Algebra

Introduction: Boole Algebra

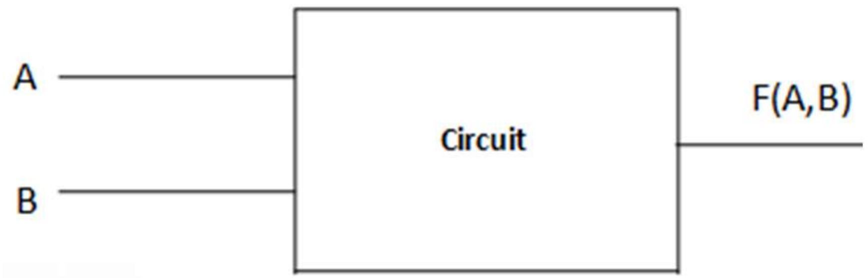
- Boolean algebra (named after George Boole, 1815 - 1864) is a mathematical theory that proposes to translate electrical signals (in two states) into mathematical expressions. It is a means of designing electronic circuits that perform complex operations, where elementary signals are defined by logical variables and their processing is governed by logical functions. Methods such as truth tables are used to define the desired operations and to transcribe the result into an algebraic expression.

Introduction: Boole Algebra

- Introduction
- Today, Boolean algebra finds numerous applications in computer science and the design of digital electronic circuits, such as memories, computing circuits, microprocessors, etc.
- Digital machines are composed of a set of electronic circuits, each providing a well-defined logical function (addition, comparison, etc.).

Boole Algebra

- We signify by B the set consisting of two elements called truth values *TRUE* and *FALSE*. This set is also represented as B = {1, 0}. On this set, two operations can be defined: **AND** and **OR**, and a transformation called complement, inversion, or **negation**.



- The function F(A, B) could be the sum of A and B, or the result of the comparison of A and B, or another function.

Boole Algebra

- To design and implement this circuit, a mathematical model of the function performed by the circuit is obligatory.
- This model must take into account the binary system.
- The mathematical model used is that of Boolean algebra.

Boole Algebra

- Examples of two-state systems:
  - A switch is either open or not open (closed).
  - A lamp is either on or not on (off).
  - A door is either open or not open (closed).
- Note: The following conventions can be used:
  - YES → TRUE (true)
  - NO → FALSE (false)
  - YES → 1 (High Level)
  - NO → 0 (Low Level)

Boole Algebra

- Definitions and Conventions Logical Level:

When studying a logical system, it is essential to specify the level of operation.

Level	**Positive Logic**	**Negative Logic**
H (Hight)	**1**	**0**
L (Low)	**0**	**1**

Example:

Positive Logic: Lamp on: 1, (L. Posi: 1)

Negative Logic: Lamp off: 0, (L. Neg: 0)

Logical operators of Boolean algebra

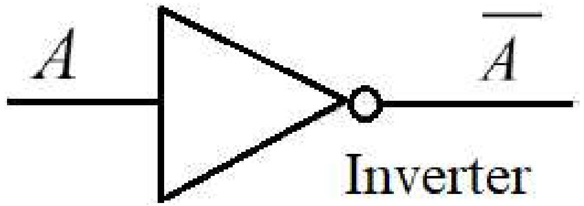
- Definition:
- We denote by B the set consisting of two elements called truth values {TRUE, FALSE}.
- This set is also represented as $B = \{1, 0\}$. On this set, two laws (operations or functions) can be defined:
- **AND** and **OR**, and a transformation called **NOT**, complement, inversion, or negation.
- These operations are referred to as the basic logical operations.

Basic logical operators

- **NOT**: (Negation): is a unary operator (operates on a single variable) that serves to reverse the value of a variable. The opposite of a variable "A" is TRUE if and only if A is FALSE.
- The negation of A is denoted :

  $\bar{A}$ (read as: A bar).

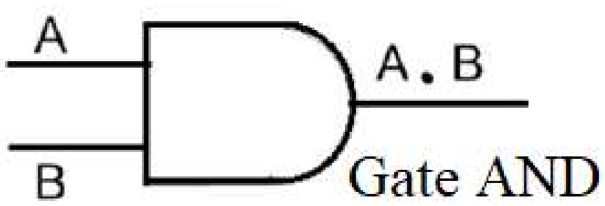
$F(A) = \text{NOT } A = \bar{A}$



A	$\bar{A}$
0	1
1	0

Basic logical operators

- **(AND)** operator:
- The AND operator is a binary operator (two variables) that aims to perform the logical product between two Boolean variables.
- AND performs the conjunction between two variables.
- The AND operator is defined
- by: **F(A,B)= A . B**



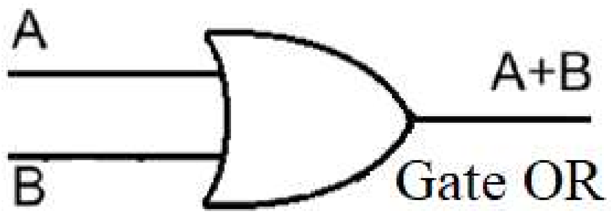
A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

Basic logical operators

- **(OR)** operator

•The OR operator is a binary operator (two variables) that aims to perform the logical sum between two Boolean variables. OR performs the disjunction between two variables. The OR operator is defined as: **F(A,B)=A+B** (it should not be confused with arithmetic addition).

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1



Basic logical operators

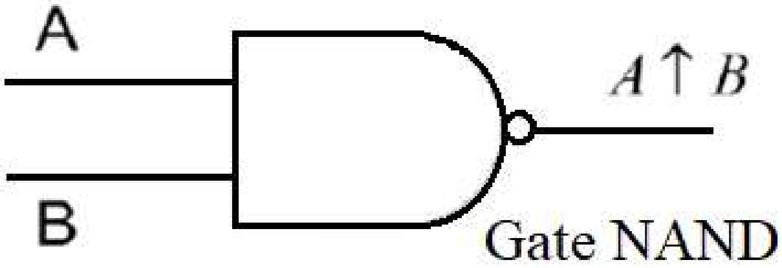
- Notes:
- In the definition of the AND, OR operators, we have delivered the basic definition with two logical variables.
- The AND operator can perform the product of multiple logical variables (e.g., A . B . C . D).
- The OR operator can also perform the logical sum of multiple logical variables (e.g., A + B + C + D).
- In an expression, parentheses can also be used.

Other logical operators

- The NOT-AND operator (NAND abbreviation) associates a result that has the value TRUE only if at least one of the two operands has the value FALSE. It is defined as follows:

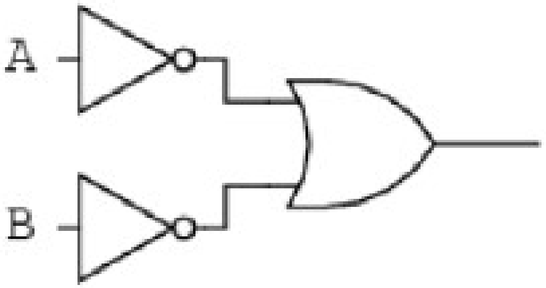
$$F(A,B) = \overline{A \, . \, B}$$

$$F(A,B) = A \uparrow B$$



A	B	$\overline{A \bullet B}$
0	0	1
0	1	1
1	0	1
1	1	0

Or:



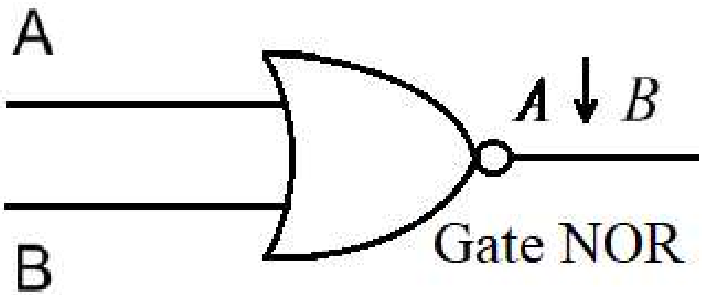
Other logical operators

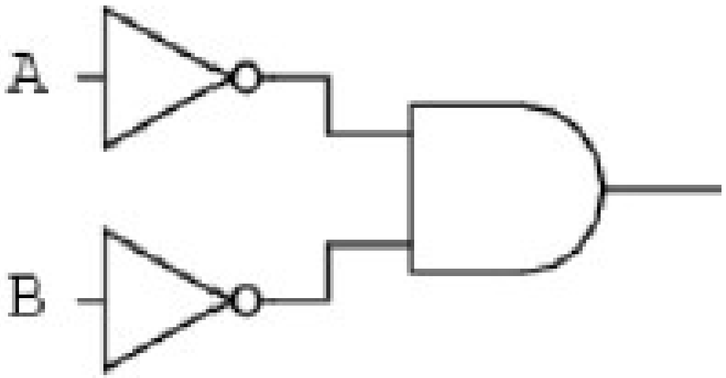
The NOT-OR operator (NOR Abbreviation) associates a result that has the value TRUE only if both operands have the value FALSE. It is defined as follows:

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

$$F(A,B) = \overline{A+B}$$

$$F(A,B) = A \downarrow B$$



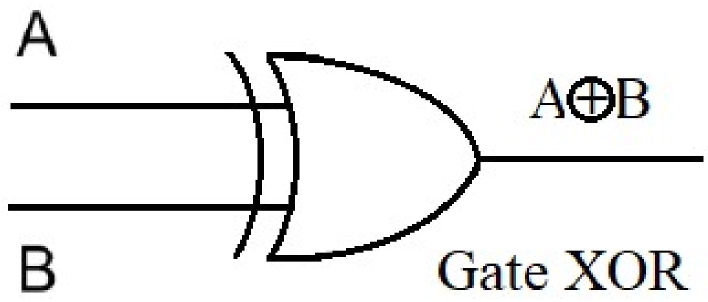


Other logical operators

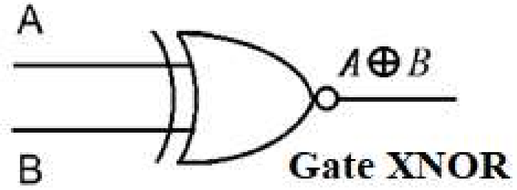
- The exclusive OR operator, often called XOR (eXclusive OR), associates a result that has the value TRUE only if the two operands have distinct values. It is defined as follows: $F(A,B)=A\oplus B$

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

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0



Other logical operators

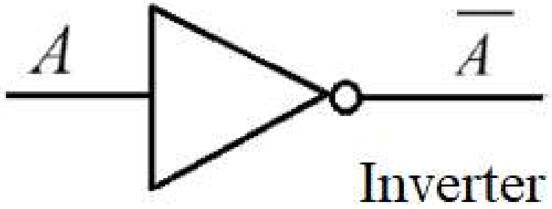


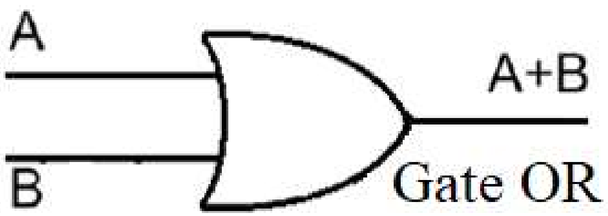
- Note: It can be noted that:
- A non XOR B is denoted as $A \otimes B$ and read as:

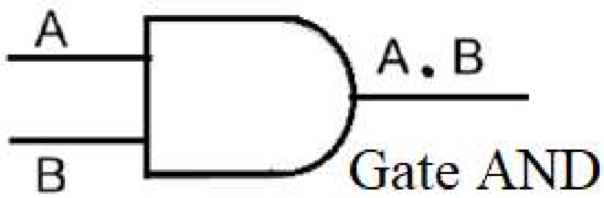
A XNOR B. The AND operator can perform the logical product of multiple variables (e.g., A . B . C . D). The OR operator can also perform the logical sum of multiple logical variables (e.g., A + B + C + D). In an expression, parentheses can also be used. The gates AND, OR, NAND, NOR can have more than two inputs. There is no exclusive OR with more than two inputs.

logic Gates

- A logic gate is a fundamental electronic circuit that enables the implementation of a basic logical operator function.

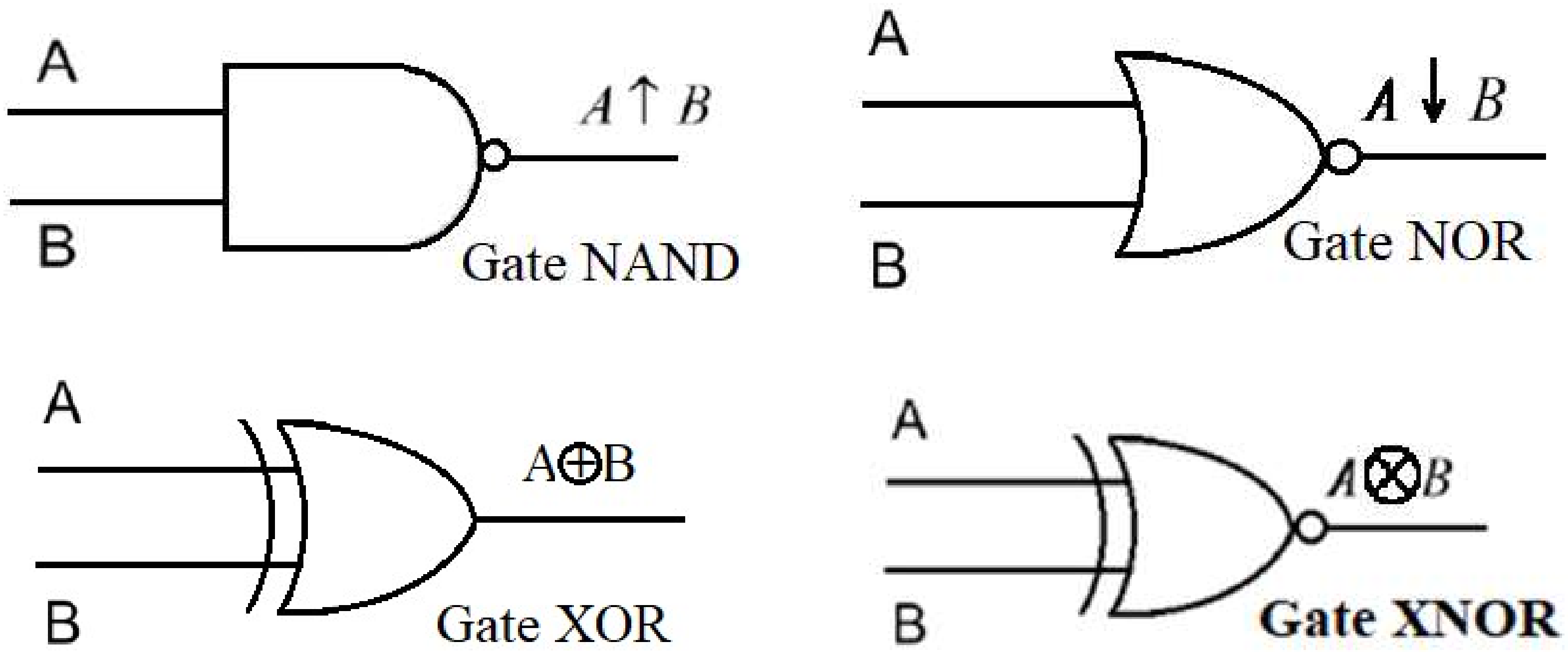






logic Gates

Note: The AND, OR, NAND, NOR gates can have more than two inputs. There is no exclusive OR **(XOR)** with more than two inputs.



Fundamental laws of Boolean algebra

- Properties of the "OR" operator:
  - ✓ $a + 1 = 1$ Absorbing element
  - ✓ $a + 0 = a$ Neutral element
  - ✓ $a + a = a$ Idempotence
  - ✓ $a + \bar{a} = 1$ Complementarity
  - ✓ $a + b = b + a$ Commutativity
  - ✓ $a + b + c = (a + b) + c = a + (b + c)$ Associativity

Fundamental laws of Boolean algebra

Properties of the "AND" operator:

- ✓ $a * 0 = 0$ Absorbing element
- ✓ $a * 1 = a$ Neutral element
- ✓ $a * a = a$ Idempotence
- ✓ $a * \bar{a} = 0$ Complementarity
- ✓ $a * b = b * a$ Commutativity
- ✓ $a * b * c = (a * b) * c = a * (b * c)$ Associativity

Fundamental laws of Boolean algebra

- Properties of the « NOT » operator:
  - ✓ $\neg \bar{a} = a$
  - ✓ $\bar{a} + a = 1$
  - ✓ $\bar{a} * a = 0$
- Distributive property:
  - ✓ $a * (b + c) = a * b + a * c$
  - ✓ $(a + b) * (c + d) = a * c + a * d + b * c + b * d$
  - ✓ $a + (b * c) = (a + b) * (a + c)$

Fundamental laws of Boolean algebra

- Propriété de l'opérateur "NAND"
  - ✓ $A\uparrow 0=1$
  - ✓ $A\uparrow 1=\bar{A}$
  - ✓ $A\uparrow B=B\uparrow A$
  - ✓ $(A\uparrow B)\uparrow C\neq A\uparrow(B\uparrow C)$
- Propriété de l'opérateur "NOR"
  - ✓ $A\downarrow 0= \bar{A}$
  - ✓ $A\downarrow 1=0$
  - ✓ $A\downarrow B=B\downarrow A$
  - ✓ $(A\downarrow B)\downarrow C\neq A\downarrow(B\downarrow C)$

Fundamental laws of Boolean algebra

- Note 1:
- All these formulas allow for the simplification of logical functions, meaning to eliminate logical operators as much as possible (without, of course, altering the initial function).
- Note 2:
- NAND and NOR are universal (complete) operators, meaning that by using them, any logical function can be expressed. To achieve this, it is sufficient to express the basic operators (NOT, AND, OR) with NAND and NOR.

Fundamental laws of Boolean algebra

- **Exemple :**

Implementation of basic operators using NOR gates.

$$A = A + A$$

Fundamental laws of Boolean algebra

- **Exemple :**

Implementation of basic operators using NOR gates.

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

Fundamental laws of Boolean algebra

- **Exemple :**

Implementation of basic operators using NOR gates.

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

Fundamental laws of Boolean algebra

- **Exemple :**

Implementation of basic operators using NOR gates.

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

$$A + B =$$

Fundamental laws of Boolean algebra

- **Exemple :**

Implementation of basic operators using NOR gates.

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

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

Fundamental laws of Boolean algebra

- **Exemple :**

Implementation of basic operators using NOR gates.

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

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

Fundamental laws of Boolean algebra

- **Exemple :**

Implementation of basic operators using NOR gates.

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

$$A + B = \overline{\overline{A + B}} = \overline{A \downarrow B} = (A \downarrow B) \downarrow (A \downarrow B)$$

Fundamental laws of Boolean algebra

- **Exemple :**

Implementation of basic operators using NOR gates.

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

$$A + B = \overline{\overline{A + B}} = \overline{A \downarrow B} = (A \downarrow B) \downarrow (A \downarrow B)$$

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

Fundamental laws of Boolean algebra

- **Exemple :**

Implementation of basic operators using NOR gates.

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

$$A + B = \overline{\overline{A + B}} = \overline{A \downarrow B} = (A \downarrow B) \downarrow (A \downarrow B)$$

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

Fundamental laws of Boolean algebra

- **Exemple :**

Implementation of basic operators using NOR gates.

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

$$A + B = \overline{\overline{A + B}} = \overline{A \downarrow B} = (A \downarrow B) \downarrow (A \downarrow B)$$

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

Fundamental laws of Boolean algebra

- **Exemple :**

Implementation of basic operators using NOR gates.

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

$$A + B = \overline{\overline{A + B}} = \overline{A \downarrow B} = (A \downarrow B) \downarrow (A \downarrow B)$$

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

Fundamental laws of Boolean algebra

- Duality of Boolean algebra: Any logical expression remains true if we replace: AND with OR, OR with AND, 1 with 0, and 0 with 1.
- **Exemple :**

$$A + 1 = 1 \Rightarrow A\,.\,0 = 0$$

$$A + \bar{A} = 1 \Rightarrow A.\,\bar{A} = 0$$

Fundamental laws of Boolean algebra

- De Morgan's Theorem: The complemented logical sum of two variables is equal to the logical product of the complements of the two variables.

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

- The complemented logical product of two variables is equal to the logical sum of the complements of the two variables.

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

Fundamental laws of Boolean algebra

- Generalization of De Morgan's Theorem to n variables.

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

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

Fundamental laws of Boolean algebra

- **Exemple :**
- Let S be simplified

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

Fundamental laws of Boolean algebra

- **Exemple :**
- Let S be simplified

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

Fundamental laws of Boolean algebra

- **Exemple :**
- Let S be simplified $S = (X + \overline{Y}).(X + Y) + Z.(\overline{X} + Y)$
- By distributivity $S = (X + \overline{Y}).X + (X + \overline{Y}).Y + Z.(\overline{X} + Y)$

Fundamental laws of Boolean algebra

- **Exemple :**
- Let S be simplified

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

- By distributivity

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

- By distributivity

$$S = X.X + \overline{Y}.X + X.Y + \overline{Y}.Y + Z.\overline{X} + Z.Y$$

Fundamental laws of Boolean algebra

- **Exemple :**
- Let S be simplified $S = (X + \overline{Y}).(X + Y) + Z.(\overline{X} + Y)$
- By distributivity $S = (X + \overline{Y}).X + (X + \overline{Y}).Y + Z.(\overline{X} + Y)$
- By distributivity $S = X.X + \overline{Y}.X + X.Y + \overline{Y}.Y + Z.\overline{X} + Z.Y$
- By Idempotence (x.x=x) $S = X + \overline{Y}.X + X.Y + \overline{Y}.Y + Z.\overline{X} + Z.Y$

Fundamental laws of Boolean algebra

- **Exemple :**
- Let S be simplified $S = (X + \overline{Y}).(X + Y) + Z.(\overline{X} + Y)$

- By distributivity $S = (X + \overline{Y}).X + (X + \overline{Y}).Y + Z.(\overline{X} + Y)$
- By distributivity $S = X.X + \overline{Y}.X + X.Y + \overline{Y}.Y + Z.\overline{X} + Z.Y$
- By Idempotence (x.x=x) $S = X + \overline{Y}.X + X.Y + \overline{Y}.Y + Z.\overline{X} + Z.Y$
- By Complementarity y.y=0 $S = X + \overline{Y}.X + X.Y + Z.\overline{X} + Z.Y$

Fundamental laws of Boolean algebra

- **Exemple :**
- Let S be simplified

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

- By distributivity

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

- By distributivity

$$S = X.X + \overline{Y}.X + X.Y + \overline{Y}.Y + Z.\overline{X} + Z.Y$$

- By Idempotence (x.x=x)

$$S = X + \overline{Y}.X + X.Y + \overline{Y}.Y + Z.\overline{X} + Z.Y$$

- By Complementarity y.y=0

$$S = X + \overline{Y}.X + X.Y + Z.\overline{X} + Z.Y$$

- By remarkable identity (1.x=x)

$$S = 1.X + \overline{Y}.X + X.Y + Z.\overline{X} + Z.Y$$

Fundamental laws of Boolean algebra

- **Exemple :**
- Let S be simplified

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

- By distributivity

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

- By distributivity

$$S = X.X + \overline{Y}.X + X.Y + \overline{Y}.Y + Z.\overline{X} + Z.Y$$

- By Idempotence (x.x=x)

$$S = X + \overline{Y}.X + X.Y + \overline{Y}.Y + Z.\overline{X} + Z.Y$$

- By Complementarity y.y=0

$$S = X + \overline{Y}.X + X.Y + Z.\overline{X} + Z.Y$$

- By remarkable identity (1.x=x)

$$S = 1.X + \overline{Y}.X + X.Y + Z.\overline{X} + Z.Y$$

- By distributivity

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

Fundamental laws of Boolean algebra

- **Exemple :**
- Let S be simplified $S = (X + \overline{Y}).(X + Y) + Z.(\overline{X} + Y)$

- By distributivity $S = (X + \overline{Y}).X + (X + \overline{Y}).Y + Z.(\overline{X} + Y)$
- By distributivity $S = X.X + \overline{Y}.X + X.Y + \overline{Y}.Y + Z.\overline{X} + Z.Y$
- By Idempotence (x.x=x) $S = X + \overline{Y}.X + X.Y + \overline{Y}.Y + Z.\overline{X} + Z.Y$
- By Complementarity y.y=0 $S = X + \overline{Y}.X + X.Y + Z.\overline{X} + Z.Y$
- By remarkable identity (1.x=x) $S = 1.X + \overline{Y}.X + X.Y + Z.\overline{X} + Z.Y$
- By distributivity $S = X.(1 + \overline{Y} + Y) + Z.\overline{X} + Z.Y$
- By remarkable identity (on + then *) $S = X + Z.\overline{X} + Z.Y$

Logic Circuit

Logic Circuit: Logigram (logic diagram)

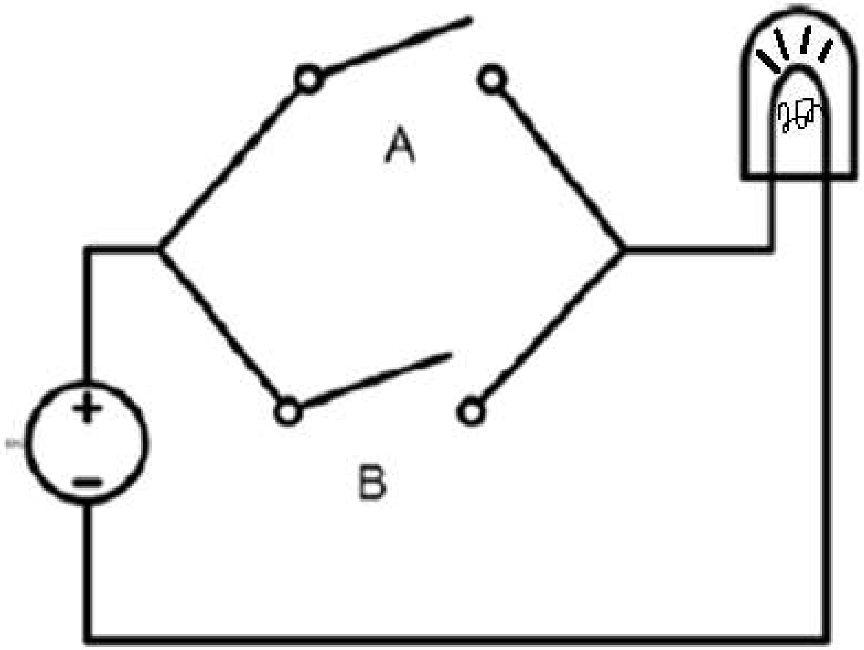
- Concept of a Logic Circuit (Flowchart)

The connection established between Boolean algebra and logic circuits dates back to the early 20th century. It was a true revolution whose consequences we are well aware of today.

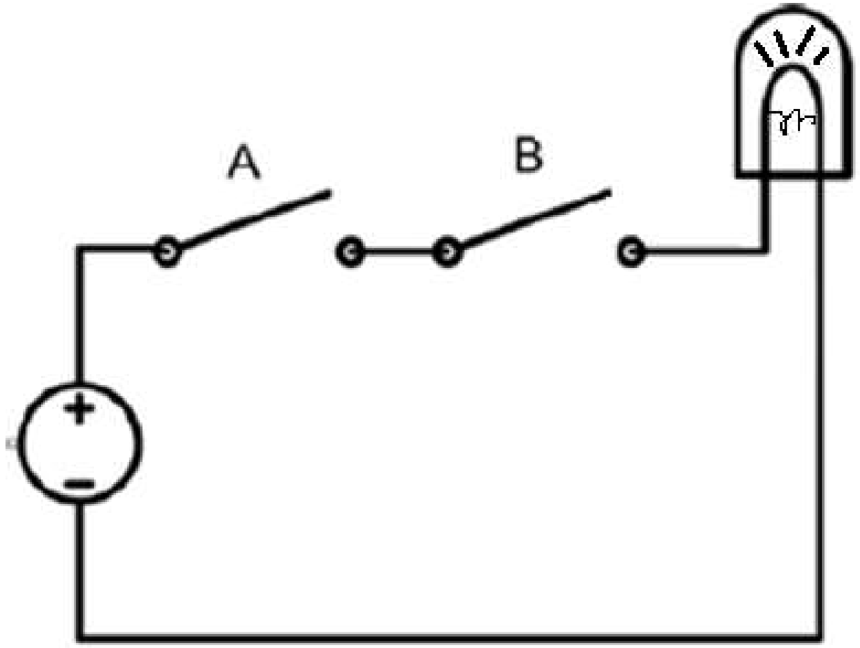
- It initially involved an application to relay circuits (a kind of buttons). If relays responded to the same command (variable), then a logical function could express its general operation:

Logic Circuit: Logigram (logic diagram)

- Examples of application to relay circuits:

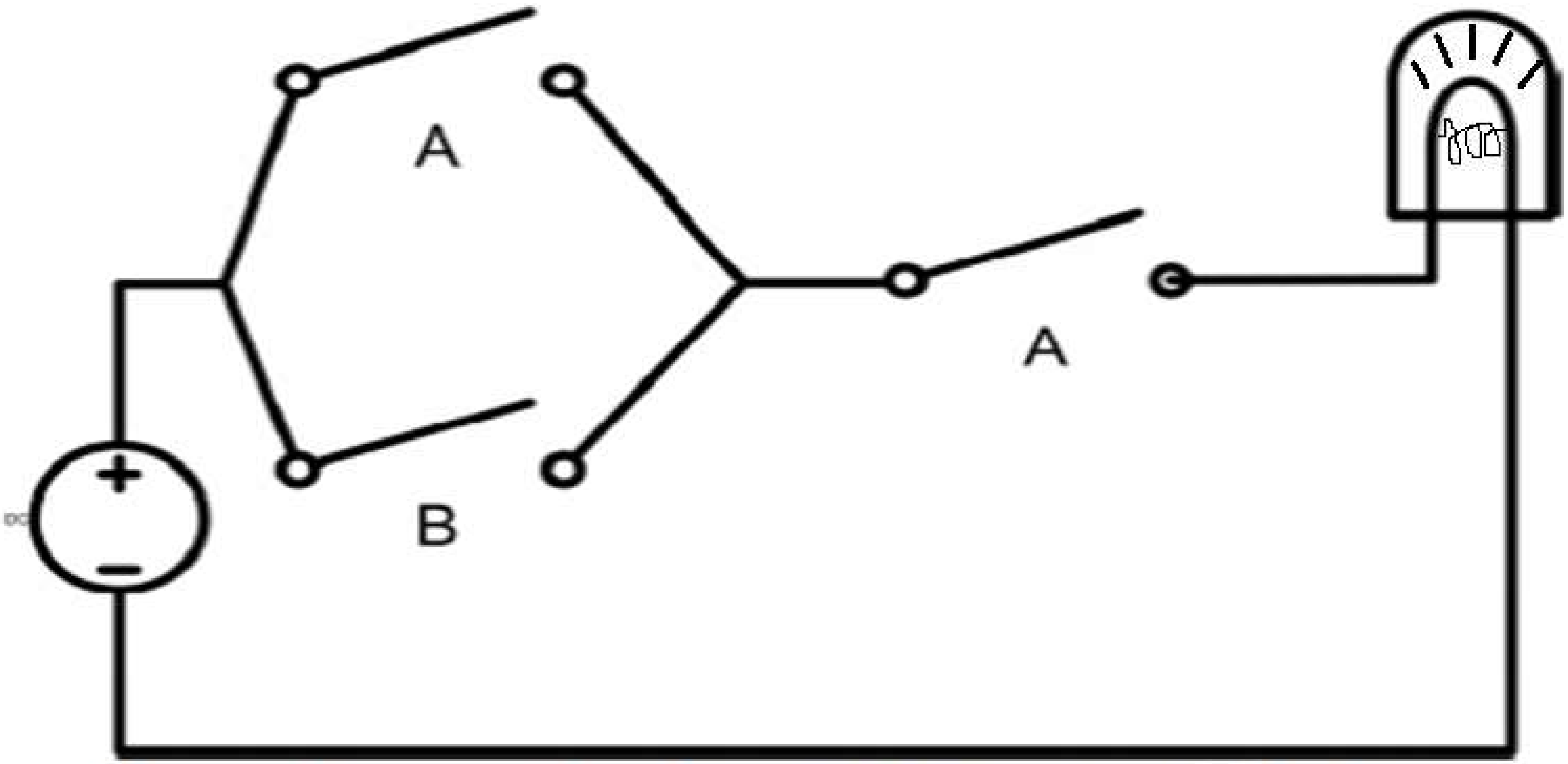


The lamp lights up if A OR B is closed.



The lamp lights up if A AND B are closed.

Logic Circuit: Logigram (logic diagram)



The lamp lights up if (A OR B) AND A is closed. Therefore, the lamp lights up if A is closed, since: A(A+B)=A.

Logic Circuit

A logic circuit is the translation of a logical function into an electronic schematic.

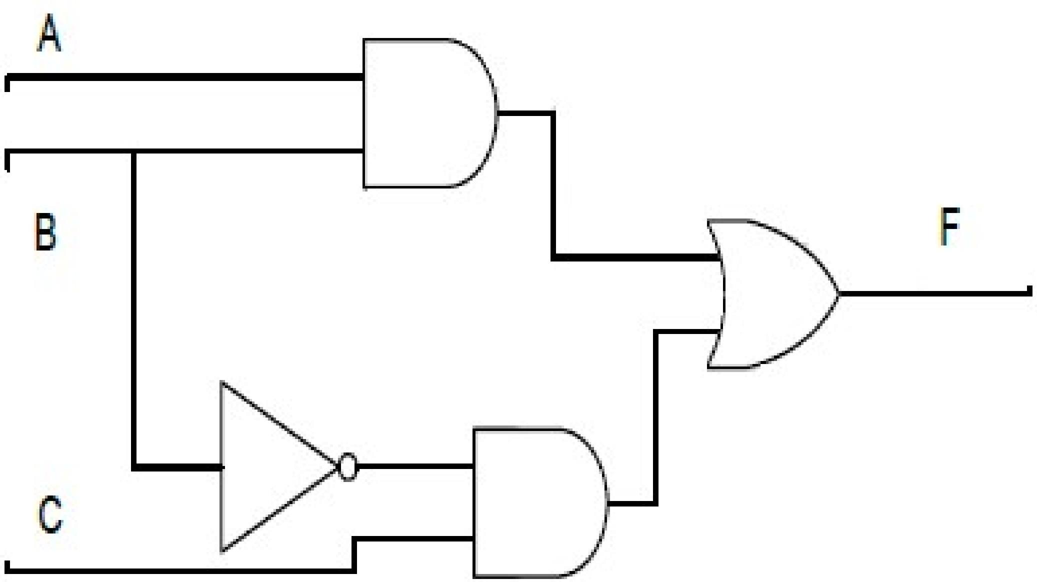
The principle involves replacing each logical operator with the corresponding logic gate.

Logic gates are the basic elements with which we can express any logical function. The variables of a logical function become the inputs of the circuit, and this circuit outputs the value of the logical function based on the input values.

Logic Circuit

- We can connect logic gates to each other to implement a logical function. On the contrary, finding the logical function implemented by a circuit allows us to manipulate it for potential simplifications.
- **Exemple :**

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



Logical function

- Function that connects N logical variables with a set of basic logical operators. There are three basic operators: NOT, AND, OR. The value of a logical function is equal to: 1 or 0 based on the values of the logical variables.
- If a logical function has N logical variables→ 2^n combinations → the function has 2^n values.
- The 2^n combinations are represented in a table called a truth table (TT),

Logical function

- Example of a logical function

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

- The function has: 3 variables,

2^3 combinations

The truth table:

A table representing the values taken by a Boolean expression for each possible combination of its inputs.

A	B	C		F
0	0	0		0
0	0	1		1
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		0
1	1	1		1

Logical function

- Textual definition of a logical function

- Generally, the definition of how a system operates is provided in textual format.
- Thus, to study and implement such a system, we must have its mathematical model (logical function)
- → it is necessary to derive (deduce) the logical function from the textual description.

Logical function

- **Exemple :**

  - ✓ A security lock opens based on three keys.
  - ✓ The operation of the lock is defined as follows:
  - ✓ The lock is open if at least two keys are used.
  - ✓ The lock remains closed in other cases.

- Provide the circuit diagram that controls the opening of the lock.

Logical function

- The system has three inputs:
  each input represents a key.
- We will correspond each key to a logical variable:
  key1 $\to$ A, key2 $\to$ B, key3 $\to$ C

- If key1 is used, then variable A = 1, otherwise A = 0.
- If key2 is used, then variable B = 1, otherwise B = 0.
- If key3 is used, then variable C = 1, otherwise C = 0.

Logical function

- Le système possède une seule sortie qui correspond à l'état de la serrure (ouverte ou fermé).
- On va correspondre une variable **S** pour designer la **sortie**
- **S=1** si la serrure est **ouverte**,
- **S=0** si elle est **fermée** ,

$$S=F(A,B,C)=\begin{cases} F(A,B,C)=1 \text{ si au moins deux clés sont introduites} \\ F(A,B,C)=0 \text{ sinon .} \end{cases}$$

Logical function

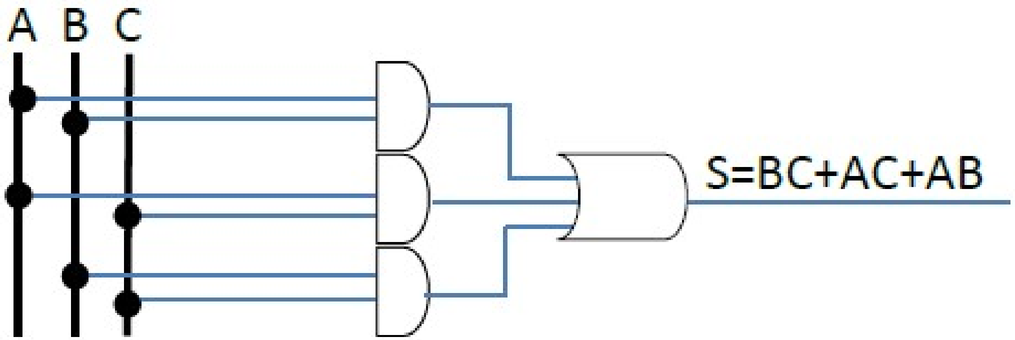
- Thus, the truth table of the system will be as follows: According to the truth table, we have:

S=F(A, B, C)= $\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

S=BC+AC+AB

- Then the corresponding circuit

  is as follows:



A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	**1**
1	0	0	0
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Logical function

- Canonical Form of a Logical Function:

- The canonical form of a function is referred to as the form where each term of the function includes all variables.
- It is called canonical because it is unique for each function (however, a canonical expression is not necessarily optimal).

Logical function

- According to the duality principle essential to Boolean algebra, two canonical forms are identified:
- 1- Sum of products (called disjunctive canonical form or the sum of minterms). A sum of products is in canonical form if all variables appear in all terms of the products that compose it.
- 2- Product of sums (called conjunctive canonical form or product of maxterms). A product of sums is in canonical form if all variables appear in all terms of sums that compose it.

Logical function

Line	A	B	C	F	*Minterms* m_i	*Maxterms* M_i
0	0	0	0	0	$\bar{A}\bar{B}\bar{C}$	$A + B + C$
1	0	0	1	0	$\bar{A}\bar{B}C$	$A + B + \bar{C}$
2	0	1	0	0	$\bar{A}B\bar{C}$	$A + \bar{B} + C$
3	0	1	1	1	$\bar{A}BC$	$A + \bar{B} + \bar{C}$
4	1	0	0	0	$A\bar{B}\bar{C}$	$\bar{A} + B + C$
5	1	0	1	1	$A\bar{B}C$	$\bar{A} + B + \bar{C}$
6	1	1	0	1	$AB\bar{C}$	$\bar{A} + \bar{B} + C$
7	1	1	1	1	ABC	$\bar{A} + \bar{B} + \bar{C}$

Logical function

- From this, we deduce the expression of F in disjunctive canonical form (sum of minterms) and its condensed representations:

$$F(A,B,C) = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$F(A, B, C) = m_3 + m_5 + m_6 + m_7$$

F	*Minterms* m_i
0	$\bar{A}\bar{B}\bar{C}$
0	$\bar{A}\bar{B}C$
0	$\bar{A}B\bar{C}$
1	$\bar{A}BC$
0	$A\bar{B}\bar{C}$
1	$A\bar{B}C$
1	$AB\bar{C}$
1	ABC

- From this, we deduce the expression of F in conjunctive canonical form (product of maxterms) and its condensed representations:

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

$$F(A, B, C) = M_0.M_1.M_2.M_4$$

Logical function

There is another representation of the canonical forms of a logical function, and this representation is called the numeric form.

R or Σ: to indicate the disjunctive form.

P or Π: to indicate the conjunctive form.

If we take the function from the previous example:

$$F(A,B,C) = \sum(3,5,6,7) = \mathrm{R}(011,101,110,111)$$
$$= \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC.$$

$$F(A,B,C) = \overline{\prod}(0,1,2,4) = \mathrm{P}(000,001,010,100)$$
$$= (A+B+C)(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+C)$$

Logical function

Note:

We can always reduce any logical function to one of the canonical forms. This involves adding the missing variables in terms that do not contain all variables (non-canonical terms), using the rules of Boolean algebra:

- Multiply a term by an expression that equals 1.
- Add to a term an expression that equals 0.
- Subsequently, perform distribution.

Logical function

•The first and second canonical forms are equivalent.
It is possible to transition from a disjunctive canonical expression to a conjunctive canonical expression (and vice versa) by considering the missing terms in the dual.

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

Logical function

- Simplification of logical functions
- Why?
- To use the fewest possible components;
- To simplify the wiring diagram as much as possible by reducing the number of logic gates used → reducing the circuit cost.
- Therefore, it is necessary to find the minimal form of the considered logical expression, and for that, we must:
  - Reduce the number of terms in a function;
  - Reduce the number of variables in a term.

Logical function

- Three methods:
  1. Algebraic (using properties and theorems)
  2. Graphic (Karnaugh maps; ...)
  3. Programmable (Quine-McCluskey method)

- Algebraic Simplification This method does not have a specific approach; its principle is to apply the rules of Boolean algebra to eliminate variables or terms.

Logical function

Thus, this simplification technique relies on the use of fundamental theorems and properties of Boolean algebra.

After finding the algebraic expression of the function, the next step is to minimize the number of terms in a function to obtain a smaller circuit, hence easier to construct with reduced cost.

Algebraic simplification is based on various actions; however, when the function is more complex (beyond three variables), this simplification method becomes less authentic.

Logical function

- Supprimer les associations de termes multiples.
- Mettre en facteur des variables pour éliminer plusieurs termes.
- Mettre en facteur des variables pour faire apparaître des termes inclus.
- Ajouter un terme qui existe déjà à une expression logique.

Logical function

- Eliminate associations of multiple terms.
- Factorize variables to eliminate multiple terms.
- Factorize variables to reveal included terms.
- Add a term that already exists to a logical expression.

Logical function

- Some fundamental rules:
- Rule 1: In a sum, all multiples of a fundamental term can be eliminated. X + XY = X.
- Rule 2: (Absorption): In the sum of a term and a multiple of its complement, the complement can be eliminated.

$$X + \bar{X}Y = X + Y$$

- Rule 3: Assembly terms using the rules of Boolean algebra.

$$
\begin{aligned}
ABC + AB\bar{C} + A\bar{B}CD &= AB(C + \bar{C}) + A\bar{B}CD \\
&= AB + A\bar{B}CD = A(B + \bar{B}(CD)) \\
&= A(B + CD) = AB + ACD
\end{aligned}
$$

Logical function

- Rule 4: Add an existing term to an expression.

$$ABC + \bar{A}BC + A\bar{B}C + AB\bar{C} =$$
$$= ABC + \bar{A}BC + ABC + A\bar{B}C + ABC + AB\bar{C}$$
$$= BC + AC + AB$$

- Rule 5: It is possible to eliminate an unnecessary term (an extra term), meaning it is already included in the union of the other term

$$\begin{aligned} F(A,B,C) &= AB + \bar{B}C + AC \\ &= AB + \bar{B}C + AC\,(B + \bar{B}) \\ &= AB + \bar{B}C + ACB + A\bar{B}C \\ &= AB\,(1 + C) + \bar{B}C\,(1 + A) \\ &= AB + \bar{B}C \end{aligned}$$

Logical function

- Rule 6: It is preferable to simplify the canonical form using the minimum number of terms.

$$F(A,B,C) = R(2,3,4,5,6,7)$$

$$\overline{F(A,B,C)} = R(0,1) = \overline{A}.\overline{B}.\overline{C} + \overline{A}.\overline{B}.C$$

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

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

$$F(A,B,C) = \overline{\overline{F(A,B,C)}} = \overline{\overline{A + B}} = A + B$$

Logical function

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

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

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

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

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

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

Logical function

- Example 1: Simplify the following expression using the rules of Boolean algebra.

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

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

$$= B+C+A.B+B.C$$

$$= B.(1+A)+C.(1+B)$$

$$= B+C$$

Logical function

- Example 2: Simplify the following expression using the rules of Boolean algebra.

$$
\begin{aligned}
F(A,B,C) &= \overline{\overline{A+B}+\overline{\overline{A}+\overline{C}}}+\overline{A+C} \\
&= (A+B).(\overline{A}+\overline{C})+\overline{A}.\overline{C} \\
&= A.\overline{A}+A.\overline{C}+\overline{A}.B+B.\overline{C}+\overline{A}.\overline{C} \\
&= \overline{C}.(A+\overline{A}+B)+\overline{A}.B \quad \text{Because : } A+\overline{A}=1 \text{ et } 1+B=1 \\
&= \overline{A}.B+\overline{C}
\end{aligned}
$$

Logical function

- Example 3: Simplify the following expression using the rules of Boolean algebra.

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

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

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

$$= A.\overline{D} + B.(\overline{D} + D.\overline{C}) \qquad X + \bar{X}Y = X + Y$$

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

End of Part 01
of Chapter 04