

Chapitre 3 : Boolean Algebra

At the end of this course, the learner will be able to:

- Know the basic operations of Boolean algebra using their various properties.
- Comprehend the functioning of logic gates.
- Apply the set of theorems of Boolean algebra.
- Simplify logical functions using algebraic and graphical methods.

Prerequisites:

- Mathematics (Linear Algebra).
- Basic electricity.

1. Introduction

Boolean algebra is an algebraic structure that contains only two elements, which are commonly called logical variables. These variables can only have two states:

- 0: False
- 1: True

Or: open or closed, off or on, inactive or active, released or pressed. Like any other algebra, there are in Boolean Algebra operations, variables, and functions. These take the name of:

- Logical Operations
- Logical Variables
- Logical Functions

Therefore, it can be said that Boolean algebra is an algebra that operates on logical variables using logical operators to achieve a logical function.

2. Logical Operations

Six logical operations are defined:

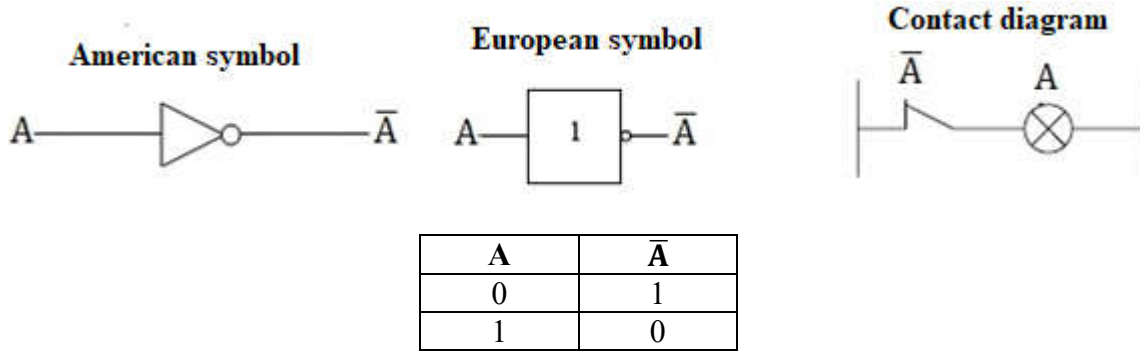
- Three main operations.
- Three secondary operations.

2.1. The main operations

a. NOT:

Commonly called inverter, with a single input and a single output, it is an operator that performs the complement of a logical variable A, noted:

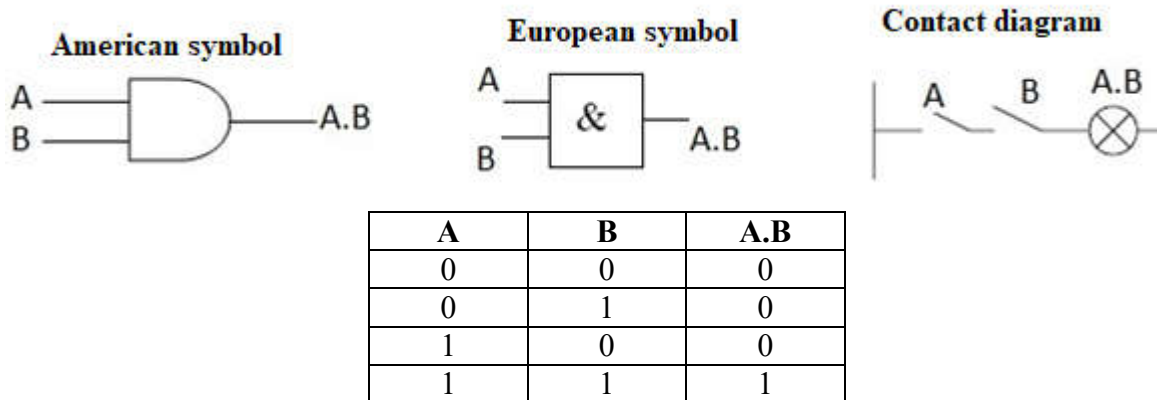
$$NOT A = \bar{A}$$



b. AND:

This is the logical product of two or more logical variables. The result of the operation is 1 when all the variables are 1. If A and B represent two logical variables, the result of the AND operation between these two variables is noted:

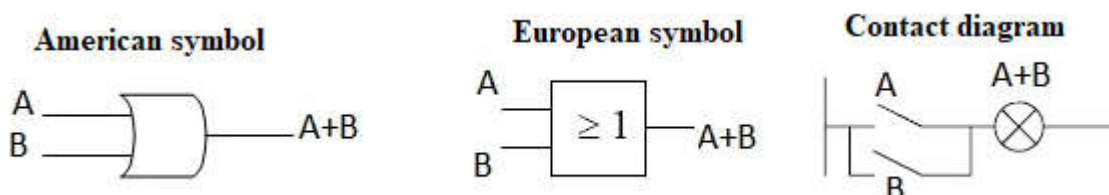
$$A \text{ AND } B = A.B$$



c. OR:

It is the logical sum of two or more logical variables; the result of the operation is 1 when at least one of the variables is equal to 1. If A and B represent two logical variables, the result of the OR operation between these two variables A and B is noted:

$$A \text{ OR } B = A + B$$



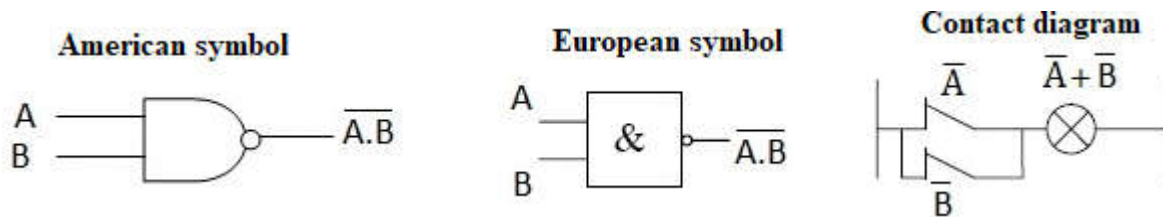
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

2.2.Secondary operations

a.NAND:

It is the complement of logical product of two logical variables A and B denoted:

$$A \text{ NAND } B = \overline{A \cdot B}$$

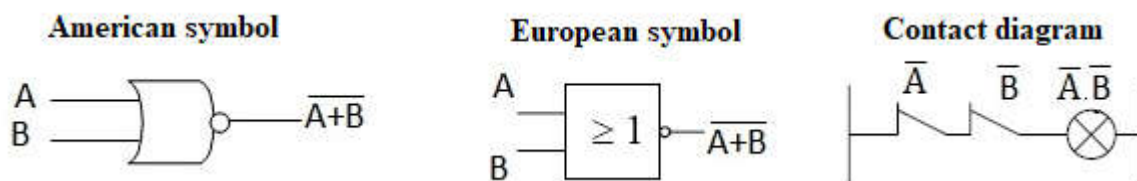


A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

b.NOR:

It is the equivalent of an OR operation followed by a NOT operation of the logical sum of two logical variables A and B noted:

$$A \text{ NOR } B = \overline{A + B}$$

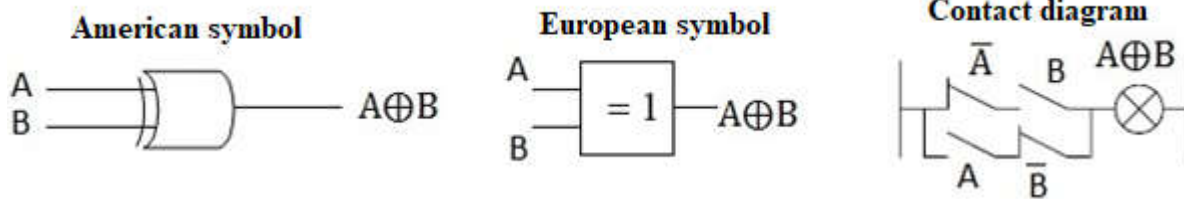


A	B	$\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

c. XOR:

This operation gives as result 1, if and only if one of the two variables is equal to 1, it is defined by:

$$A \text{ XOR } B = A \oplus B$$



A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

XOR is equal to 1 if and only if $A = 1$ or $B = 1$, but not simultaneously. An XOR operation provides an inequality comparator: XOR is 1 only if A and B are different. The complement of XOR corresponds to an equality detector.

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

3. De-Morgan's theorem

Two essential laws of Boolean algebra have been bequeathed to us by the mathematician De Morgan. De Morgan's theorems prove to be of great utility in simplifying expressions involving complements of variables in sums or products. Here are these two theorems:

Theorem:

- The complement of a logical product is equal to the sum of the complements.
- The complement of a logical sum is equal to the product of the complements.

$$\overline{\prod_{i=1}^n A_i} = \sum_{i=1}^n \bar{A}_i$$

$$\overline{\sum_{i=1}^n A_i} = \prod_{i=1}^n \bar{A}_i$$

De Morgan's theorems allow for the transformation of AND into OR and vice versa. The pair (AND, NOT) or the pair (OR, NOT) are therefore sufficient to express any combinatorial algebraic formula.

Examples :

Simplify using Morgan's theorem the following functions:

$$F_1 = \overline{\bar{A}B(A+B)} + \overline{A\bar{B}(A+\bar{B}+\bar{C})}$$

$$F_2 = \overline{(C + D)ACD(\overline{AC} + \overline{D})}$$

En appliquant le théorème de De-Morgan :

$$\begin{aligned} F_1 &= \overline{\overline{AB}(A + B) + \overline{AB}(A + \overline{B} + \overline{C})} = \overline{\overline{AB} + \overline{A}\overline{B} + \overline{AB}(\overline{A.B.C})} \\ &= \overline{\overline{A} + B + \overline{A}\overline{B} + (\overline{A.B.C})} \\ &= \overline{\overline{A}(1 + \overline{B}) + B + (\overline{A.B.C})} \\ &= \overline{\overline{A} + B + (\overline{A.B.C})} \\ &= \overline{\overline{A}(1 + B.C) + B} \\ \Rightarrow F_1 &= \overline{\overline{A} + B} \end{aligned}$$

$$\begin{aligned} F_2 &= \overline{(C + D)ACD(\overline{AC} + \overline{D})} \\ &= \overline{(\overline{C}\overline{D}) + \overline{ACD} + ((A + \overline{C}).D)} \\ &= \overline{(\overline{C}\overline{D}) + (\overline{ACD}) + ((A + \overline{C}).D)} \\ &= \overline{(\overline{C}\overline{D}) + (\overline{ACD}) + (AD + \overline{C}.D)} \\ &= \overline{(\overline{C}\overline{D}) + (\overline{ACD}) + (AD + \overline{C}.D)} \\ &= \overline{\overline{C}(\overline{D} + D) + \overline{ACD} + AD} \\ &= \overline{\overline{C}(\overline{D} + D) + AD(1 + \overline{C})} \\ &= \overline{\overline{C} + AD} \\ \Rightarrow F_2 &= AD + \overline{C} \end{aligned}$$

4. Properties of logical operators

The operators mentioned above respect the following laws, which can be easily verified through complete induction, meaning by checking all possible cases using truth tables.

4.1. Properties of AND and OR operators

Associativity	$A + (B + C) = (A + B) + C$ $A.(B.C) = (A.B).C$
Commutativity	$A + B = B + A$ $A..B = B.A$
Distributivity	$A + (B.C) = (A + B)(A + C)$ $A.(B + C) = (A.B) + (A.C)$
The neutral element.	$A + 0 = A$ $A..1 = A$
Complementarity	$A + \overline{A} = 1$ $A..\overline{A} = 0$
Involution of negation	$\overline{\overline{A}} = A$

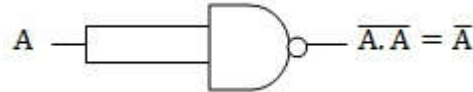
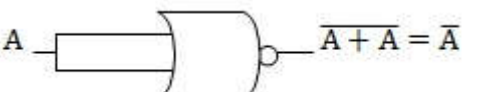
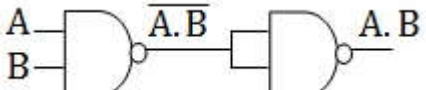
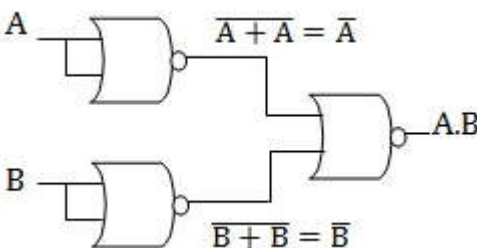
Invariance	$A + 1 = 1$ $A \cdot 0 = 0$
Idempotence	$A + A = A$ $A \cdot A = A$
The remarkable identities	$A + AB = A(1 + B) = A$ $A \cdot (A + B) = A + A \cdot B = A$ $(A + \overline{B})B = AB$ $\overline{A}B + B = A + B$ $(A + B)(B + C)(C + \overline{A}) = (A + B)(C + \overline{A})$

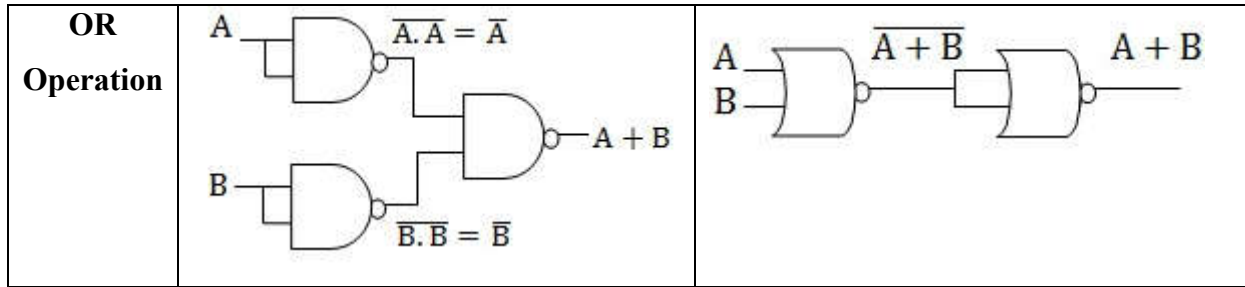
4.2. Properties of NAND and NOR Operators

Associativity	$\overline{(\overline{A \cdot B}) \cdot C} \neq \overline{A \cdot (\overline{B \cdot C})}$ $\overline{(\overline{A + B}) + C} \neq \overline{A + (\overline{B + C})}$	
Commutativity	$\overline{A + B} = \overline{B + A}$ $\overline{A \cdot B} = \overline{B \cdot A}$	
Distributivity	NAND	NOR
	$A + \overline{(B \cdot C)} \neq \overline{(A + B)(A + C)}$ $\overline{A \cdot (B + C)} \neq \overline{(A \cdot B) + (A \cdot C)}$ $\overline{A \cdot (B \cdot C)} \neq \overline{(A \cdot B) + (A \cdot C)}$ $\overline{A \cdot (BC)} \neq \overline{(A \cdot B) \cdot (A \cdot C)}$	$A + \overline{(B + C)} \neq \overline{(A + B) + (A + C)}$ $A + \overline{(B + C)} \neq \overline{(A + B) \cdot (A + C)}$ $\overline{A \cdot (B + C)} \neq \overline{(A + B) + (A + C)}$ $\overline{A \cdot (B + C)} \neq \overline{(A \cdot B) + (A \cdot C)}$

5. Universality of NAND and NOR operators

This is to perform the main operations using the NAND and NOR logic gates.

NOT Operation	Réalisation par la porte NAND	Réalisation par la porte NOR
		
AND operation		



6. Logical functions

It is a set of logical variables connected by logical operators (+, ., NOT, ...). A logical function can only take two values: 0 or 1. If the number of logical variables is equal to N , we say that the function is of order N .

6.1. Representations of logical functions

A logical function is a combination of logical variables connected by the AND, OR, and NOT operators. It can be represented by an algebraic writing or a truth table or a KARNAUGH table or a logic diagram.

6.1.1. algebraic expression

A logical function can be represented in algebraic form, which is a representation in the form of an expression. It can be expressed as a combination of logical sums and products:

$$F = BCD + ACD + \overline{BCD} + AB + \overline{AC}$$

6.1.2. Truth table

The most common representation of a logical function is the truth table. A function F of N variables is fully described by the statement of the set of combinations of input variables and the value of the function corresponding to each combination. This statement generally takes the form of a table with $N+1$ columns (N inputs + 1 output) and 2^N lines (on N bits, 2^N different values can be coded). The $(N+1)^{\text{th}}$ column contains the values that the function takes for each combination of variables.

Example :

Either:

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

Describe its truth table

Solution :

3 variables $\rightarrow 2^3$ combinations

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

6.1.3. Karnaugh Map

The Karnaugh map is a graphical tool that allows for a systematic simplification of a logical equation. It is a two-dimensional representation of a logical function and consists of 2^N cells. The cells are adjacent and symmetrical. Moving from one cell to another results in a change of only one bit. It provides a graphical method for simplifying logical functions. For a two-variable function, a Karnaugh map has 4 cells; for a three-variable function, it has 8 cells, and so on.

a. Karnaugh map for 2 variables A and B

		A	
		0	1
B	0		
	1		

b. Karnaugh map for 3 variables A, B and C

		AB			
		00	01	11	10
C	0				
	1				

c. Karnaugh map for 4 variables A, B, C and D

A B C D	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

d. Karnaugh map for 5 variables A, B, C, D and E

A B C D E	000	001	011	010	110	111	101	100
00	0	4	12	8	24	28	20	16
01	1	5	13	9	25	29	21	17
11	3	7	15	11	27	31	23	19
10	2	6	14	10	26	30	22	18
	A=0				A=1			

Example :

Represent the function using a Karnaugh map:

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

Solution :

We express f in the form of a sum of products of all logical variables:

$$\begin{aligned} F(A, B, C) &= A(\overline{B}\overline{C} + \overline{B}) + AC(B + C) \\ &= A\overline{B}\overline{C} + A\overline{B} + ACB + AC \\ &= A\overline{B}\overline{C} + A\overline{B}(C + \overline{C}) + ACB + AC(B + \overline{B}) \\ \Rightarrow F(A, B, C) &= A\overline{B}\overline{C} + A\overline{B}C + A\overline{B}\overline{C} + ABC \end{aligned}$$

Then we place 1 in the cells of the table that correspond to the combinations:

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

$F(110), F(101), F(100), F(111)$

Pour les autres combinaisons la fonction vaut 0

C \ AB	00	01	11	10
0	0	0	1	1
1	0	0	1	1

6.1.4. By a numerical expression

To simplify the representation of the function, it can be expressed in numerical form. This form indicates the decimal value corresponding to the binary combinations of the variables, for which the function equals 1.

Example :

$$F = \sum (0,1,3,5,7)$$

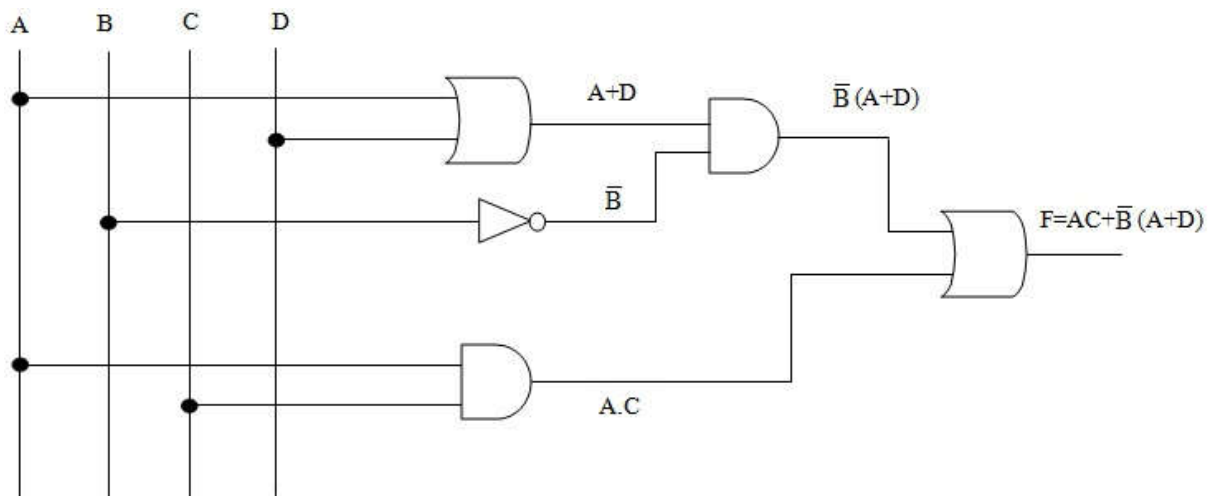
6.1.5. Representation by a Logic Diagram

It is a graphical method based on the representation of output equations by logic gates.

Example :

Represent the following equation using logic gates:

$$F = AC + \bar{B}(A + D)$$



7. Canonical forms

It is an equation that directly identifies each cell in the Karnaugh map containing a logical "1" or a logical "0." There are mainly two canonical forms, which are:

7.1. 1st canonical form

In the 1st canonical form, the function is expressed as a sum of all combinations of all logical variables for which the function is worth "1". Each term is called a minterm or fundamental product.

Example :

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

7.2. 2nd canonical form

In the 2nd canonical form, the function is expressed as a product of sums, including the all variables. Each term is called a maxterm or fundamental sum.

Example :

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

7.3. Calculation Methods

There are several methods for calculating the 1st and 2nd canonical forms of a function.

7.3.1. The Graphical Method

The logical function is represented by the Karnaugh map; the 1st canonical form corresponds to the sum of all combinations for which the function is 1. For the 2nd canonical form, we express \overline{f} using the Karnaugh map, then we complement the expression.

Example :

Determine the 1st canonical form of the function:

$$F = AC + B\overline{C}$$

The representation of the function f using the Karnaugh map is as follows:

C \ AB	00	01	11	10
0	0	1	1	0
1	0	0	1	1

a. The 1st canonical form:

$$F = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C}$$

b. The 2nd canonical form:

$$\overline{F} = \overline{\overline{A}\overline{B}\overline{C}} + \overline{\overline{A}B\overline{C}} + \overline{\overline{A}BC} + \overline{A\overline{B}\overline{C}}$$

$$\overline{\overline{F}} = \overline{\overline{\overline{A}\overline{B}\overline{C}} + \overline{\overline{A}B\overline{C}} + \overline{\overline{A}BC} + \overline{A\overline{B}\overline{C}}}$$

$$F = \overline{\overline{\overline{A}\overline{B}\overline{C}}}. \overline{\overline{\overline{A}B\overline{C}}}. \overline{\overline{\overline{A}BC}}. \overline{\overline{A\overline{B}\overline{C}}}$$

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

7.3.2. The algebraic method:**a. The 1st canonical form**

Each term of the sum is multiplied by the sum of the missing variable(s) and their complement(s).

b. The 2nd canonical form:

As for the graphical method, we first determine the expression of \overline{f} in the 1st canonical form, then we calculate its complement.

Example :

Put in the 1st and 2nd canonical form the following function:

$$F = AC + B\overline{C}$$

The 1st canonical form:

$$F = AC + B\overline{C}$$

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

$$\Rightarrow F = ABC + A\overline{B}C + AB\overline{C} + \overline{A}B\overline{C}$$

The 2nd canonical form:

$$\overline{F} = \overline{AC + B\overline{C}}$$

$$= \overline{(AC)(B\overline{C})}$$

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

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

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

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

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

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

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

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

8. Simplification of logic functions

To implement a logic function using an electronic circuit, it is necessary to ensure that the form of the function is minimal and this, in order to limit the number of circuits used to what is strictly necessary.

The objective of simplifying logical functions is to minimize the number of terms in order to obtain the simplest form of the function, or more precisely, its minimal form. Two simplification methods are used:

- Algebraic simplification.
- Graphical simplification using Karnaugh maps.

8.1. Case of Fully Defined Functions

These are functions for which their values are defined for all combinations of logical variables.

8.1.1. Algebraic Simplification

By applying the laws of Boolean algebra, the result of simplification depends on how the calculations have been carried out. It should be noted that the same term can be used several times for simplification.

Let the function be:

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

Simplify this function algebraically.

We can simplify F, by proceeding as follows:

$$\begin{aligned} F &= \overline{A}BC + A\overline{B}C + ABC + \overline{A}BC + \overline{A}BC + \overline{A}BC \\ &= (\overline{A}BC + \overline{A}BC) + (\overline{A}BC + \overline{A}BC) + (ABC + \overline{A}BC) \\ &= \overline{B}C(\overline{A} + A) + \overline{A}B(\overline{C} + C) + \overline{A}C(B + \overline{B}) \\ \Rightarrow F &= \overline{B}C + \overline{A}B + \overline{A}C \end{aligned}$$

The rules and properties of Boolean algebra allow for the simplification of functions, but algebraic simplification remains a relatively cumbersome method. It never allows us to know whether or not we end up with a minimal expression of the function. We can then use the KARNAUGH map method.

8.1.2. Simplification by Karnaugh Map

The Karnaugh map allows for visualizing a function and intuitively deriving a simplified function from it. Simplifying a logical function using the Karnaugh map is a highly effective graphical method. It is based on the principle that logical products corresponding to adjacent states can be simplified.

The simplification will consist in first representing the logical function by a Karnaugh map, and then grouping all the « 1 » values located in symmetric or adjacent cells.

The simplification rules using the Karnaugh map are as follows:

- The « 1 » values appearing in neighboring or symmetrical cells can be grouped.
- The grouping of two adjacent or symmetrical cells reduces the initial min-terms by one variable.
- Grouping 2^P adjacent or symmetrical cells reduces the initial minterms by P variables.
- Grouping must involve a power of 2 number of cells. The goal is always to group as many cells as possible.
- All « 1 » must be included in at least one grouping.
- The same cell can be used for different groupings.

Each obtained grouping represents a prime implicant.

A prime implicant that contains at least one "1" and cannot be included in any other prime implicant is called an essential prime implicant.

To obtain the minimal form, we first choose the essential prime implicants. Then, from the remaining prime implicants, we choose those that are necessary to completely cover the original function. If the minimal form consists only of essential prime implicants, then it is unique.

Example :

Simplify by the Karnaugh map the completely defined function:

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

C \ AB	00	01	11	10
0	0	1	1	0
1	1	1	0	0

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

Exercise :

Simplify using the Karnaugh map the following functions:

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

	AB	00	01	11	10
C					
0		0	1	0	0
1		1	1	1	1

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

$$F = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}CD + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}BCD + A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D + A\overline{B}C\overline{D} + ABC\overline{D}$$

	A B	00	01	11	10
C D					
00		1	0	0	1
01		1	1	1	1
11		0	1	1	0
10		1	0	0	1

$$F = BD + \overline{C}D + \overline{B}\overline{D} = D(B + \overline{C}) + \overline{B}\overline{D}$$

8.2. Case of incompletely defined functions

A function is incompletely defined when its value is indifferent or unspecified for certain combinations of input variables. This case occurs when some combinations are physically impossible. The value of the function in such cases is denoted by \emptyset . For the simplification of incompletely defined functions, the indifferent states are included in such a way as to maximize the size of the grouping. The \emptyset states may not necessarily be included in any grouping.

Example :

Simplify the incompletely defined function represented by the Karnaugh map below:

A B \ C D	00	01	11	10
00	0	0	Φ	1
01	0	0	1	1
11	1	1	Φ	Φ
10	1	1	0	Φ

$$F = \overline{A}C + \overline{A}C = A \oplus C$$

Exercise :

Find the simplest possible equation from the following table:

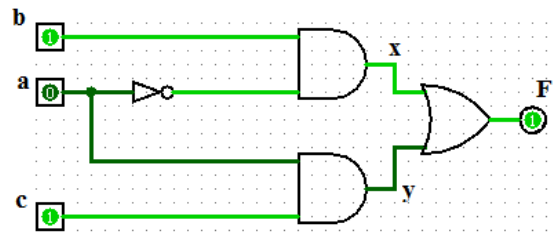
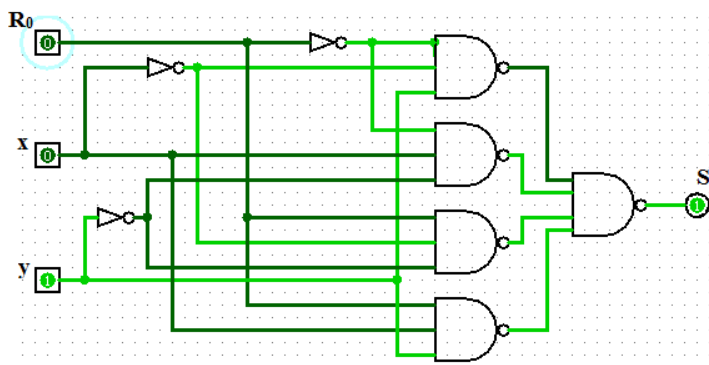
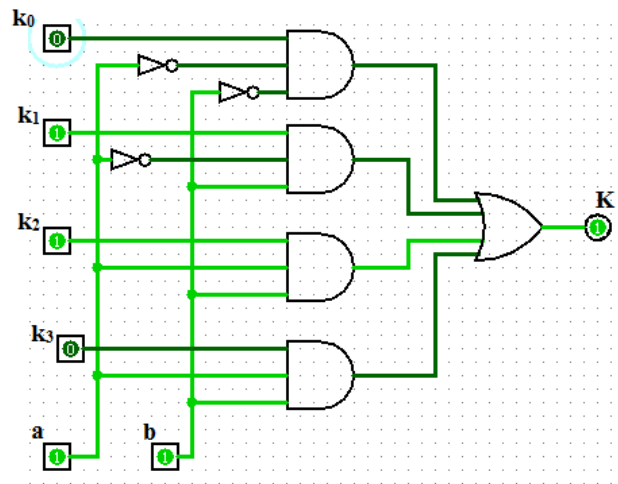
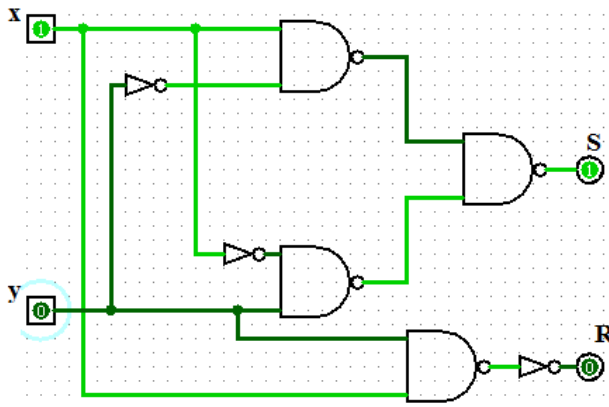
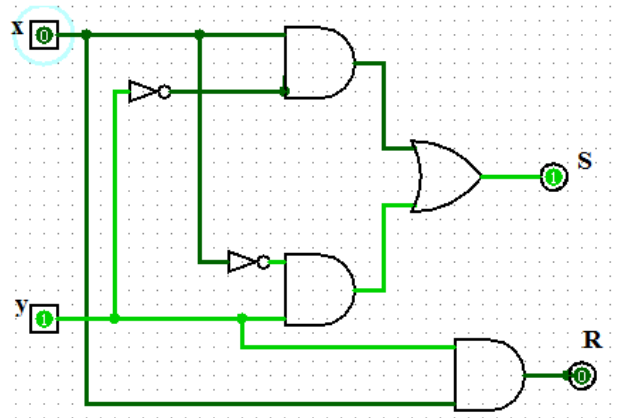
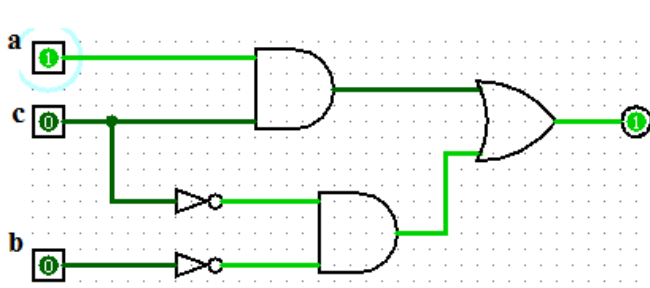
A B \ C D	00	01	11	10
00	1	\emptyset	\emptyset	1
01	0	1	1	0
11	1	0	0	1
10	\emptyset	0	0	\emptyset

After simplification, we find:

$$F = \overline{B}C + \overline{C}D + \overline{B}C = (B \oplus C) + \overline{C}D$$

Exercises :

Exercise 01 : For the following logic diagrams, find the output equations:



Exercise 02: Use Karnaugh maps to determine the simplest equation.

		ab			
c		00	01	11	10
	0	0	1	1	0
	1	1	0	0	1

R=

		ab			
cd		00	01	11	10
	00	0	1	1	0
	01	1	1	1	1
	11	0	1	1	0
	10	0	1	1	0

A =

		ab			
cd		00	01	11	10
	00	1	0	0	1
	01	0	0	0	0
	11	0	1	1	0
	10	1	0	0	1

B =

		ab			
cd		00	01	11	10
	00	0	0	0	0
	01	1	1	1	1
	11	0	0	0	0
	10	1	1	1	1

C =

		ab			
cd		00	01	11	10
	00	0	1	1	0
	01	1	1	1	1
	11	1	1	1	1
	10	0	1	1	0

D =

		ab			
cd		00	01	11	10
	00	0	0	0	0
	01	1	0	0	1
	11	1	0	0	1
	10	0	0	0	0

E =

		ab			
cd		00	01	11	10
	00	0	1	0	1
	01	1	1	1	1
	11	1	1	1	1
	10	0	1	0	1

F =

Exercise 03 : Simplify the following functions:

$$S_1 = A(A + B)$$

$$S_2 = (A + B)(\bar{A} + \bar{B})$$

$$S_3 = (\bar{A}\bar{B} + C)(A + \bar{B})C$$

$$S_4 = (A + B)C + \bar{A}(\bar{B} + C) + \bar{B}$$

$$S_5 = (A + B + C)(\bar{A} + \bar{B} + \bar{C}) + AB + BC$$

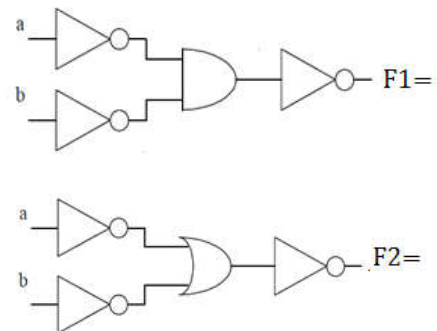
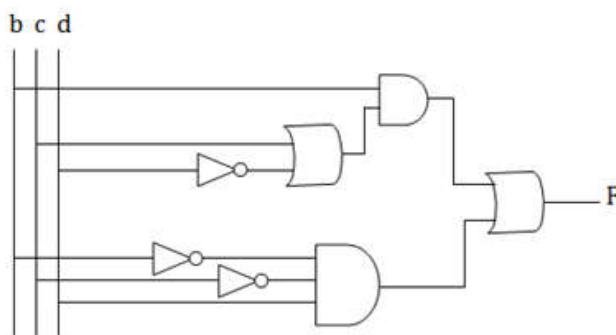
$$S_6 = A + \bar{A}\bar{B} + \bar{A}BC + \bar{A}BCD + \bar{A}BCD$$

$$S_7 = A + ABC + \bar{A}BC + \bar{A}B + AD + \bar{A}\bar{D}$$

$$S_8 = ABC\bar{C} + B(A + \bar{C}) + \bar{A} + B + \bar{A}\bar{C}$$

Exercise 04 :

Determine the output equation of the following logic diagrams.:



Exercise 05 :

Let's consider the Boolean function::

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

1. Represent y using a Karnaugh map..
2. Simplify the expression using the Karnaugh map method..
3. Provide the equation for \bar{Y} by grouping the cells with 0 in the table.
4. Complement \bar{Y} (to recover Y) by applying De-Morgan's theorems (we obtain a normal form in Π).
5. From the canonical forms in Σ and Π , and using the transformation properties, give the logic diagrams using exclusively NAND operators for one and NOR for the other.

BIBLIOGRAPHY

- [1] Électronique numérique en 26 fiches ; P. Mayé., Dunod, Paris 2010
- [2] Architecture et Technologie des Ordinateurs ; S. Tisserant., ESIL, 2003
- [3] Electronique Numérique, Systèmes électroniques numériques complexes, Modélisation et mise en œuvre, Cours et exercices corrigés ; N. Alexandre, D. Damien., Technosup, 2012.
- [4] Les Systèmes Logiques", Tome 1 ; N. Mansouri., EUMC
- [5] Architecture de l'ordinateur ; R. Strandh et I. Durand., Dunod, 2005.
- [6] Cours d'électronique numérique ; M. Siadat et C. Diou., novembre 2004

WEBOGRAPHIE

- [7] Les fonctions booléennes ; L. Lambin., <http://www.lelectronique.com>