

INTRODUCTION

Boolean Algebra was defined in 1847 by Georges Boole (1815-1864), an English physicist. It is an algebra applicable to logical reasoning which deals with functions with binary variables (only two values 1 or 0), i.e. it cannot be applied to systems with more than two states binary variables (only two values), i.e. it cannot be applied to systems with more than two states. Boolean algebra is used to manipulate logical values. Note that a logical value has only two possible states:

True (1) or False (0).

Several logical values can be combined to give a result, which is also a logical value.

DEFINITIONS

State: Logical states are represented by 0 and 1.

Variable: A quantity represented by a symbol, which can take two states (0 or 1).

Function: This represents a group of variables linked by logical operators.

Example

A large number of physical phenomena can be associated with a logical state Example: (stop, run) (open, close) (black, white) (forward, reverse) (on, off).

Logical state 1 is generally associated with the actuated state of the component.

1. COMBINATORIAL LOGIC

1.1 DEFINITION OF COMBINATORIAL LOGIC

Combinatorial logic uses logic functions to construct a combinatorial system.

A system is said "combinatorial "when it is of the open-loop type, i.e. none of the outputs is looped through as an input.

Each input combination corresponds to a single output. Combinatorial systems are the simplest and can be represented by a truth table indicating the corresponding output state for each input state.

Any logic function can be implemented using a small number of basic logic functions known as logic operators or gates. There are two types of operator:

- **Basic operators:** NOT, AND, OR
- **Derived operators:** NOT-AND (**NAND**) , NOT OR (**NOR**), Exclusive OR (**XOR**)
– Not Exclusive OR (**NXOR**).

MORGAN'S LAW

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

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

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

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

➤ *Generalisation of the MORGANE Theorem to N variables*

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

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

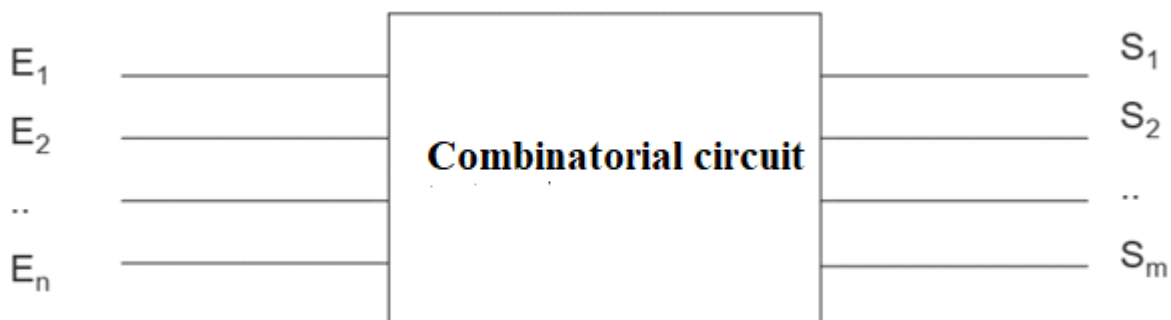
1.3. COMBINATORIAL CIRCUITS

A logic circuit (or combinational circuit) is an assembly of logic gates linked together to represent an algebraic expression.

A combinational circuit is a digital circuit whose outputs depend solely on its inputs. $S_i = F(E_i)$

$$S_i = F(E_1, E_2, \dots, E_n)$$

Combinatorial circuits are used to create other, more complex circuits.



ESTABLISHING THE TRUTH TABLE

Logic functions can be represented by truth tables. The truth table shows the output of a logic circuit as a function of the various combinations of input values.

- The number of columns is the total number of inputs and outputs
- The number of lines is 2^N , where "N" is the number of inputs,

A function with 3 inputs and 1 output is represented by a table with 4 columns and 8 rows.

A	B	C	Résultat
0	0	0	$\bar{A} \bar{B} \bar{C}$
0	0	1	$\bar{A} \bar{B} C$
0	1	0	$\bar{A} B \bar{C}$
0	1	1	$\bar{A} B C$
1	0	0	$A \bar{B} \bar{C}$
1	0	1	$A \bar{B} C$
1	1	0	$A B \bar{C}$
1	1	1	$A B C$

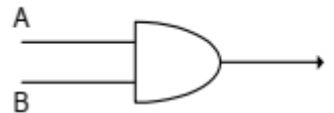
LOGICAL FUNCTION

This function links N logical variables with a set of basic logical operators.

- There are three basic operators in Boolean algebra: NOT , AND , OR .
- The value of a logic function is equal to 1 or 0, depending on the values of the logic variables.
- If a logic function has N logic variables $\rightarrow 2^n$ combinations \rightarrow the function has 2^n values.

TRUTH TABLES FOR BASIC OPERATORS**LOGICAL FUNCTION (AND)**Representation: $F = A * B$ ou $A \cdot B$ ou AB

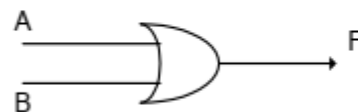
B	A	F
0	0	0
0	1	0
1	0	0
1	1	1



Symbole graphique

LOGICAL FUNCTION (OR)Representation: $F = A + B$

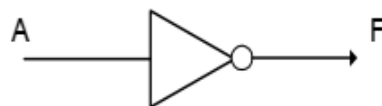
B	A	F
0	0	0
0	1	1
1	0	1
1	1	1



Symbole graphique

LOGICAL FUNCTION NOTRepresentation : $F = \bar{A}$

A	F
0	1
1	0



Symbole graphique

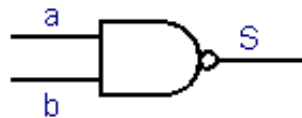
FONCTIONS DÉRIVÉES DES FONCTIONS FONDAMENTALES

LOGICAL FUNCTION NOT AND (NAND)

Representation : $F = A/B$

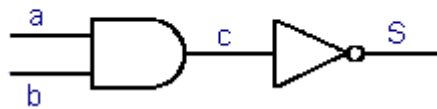
Truth table

A	B	A/B
0	0	1
0	1	1
1	0	1
1	1	0



NAND

A NAND circuit is obtained by connecting an AND gate and an inverter in series, as shown below



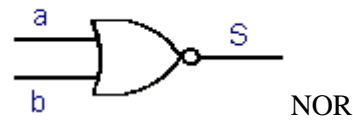
Decomposition of a NAND circuit

LOGICAL FUNCTION NOT OR(NOR)

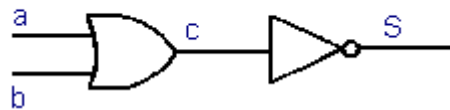
Representation : $F = A \downarrow B$

Truth table

A	B	$A \downarrow B$
0	0	1
0	1	0
1	0	0
1	1	0



A NOR circuit is obtained by connecting an OR gate and an inverter in series, as shown below



Decomposition of a NAND circuit

LOGICAL FUNCTION XOR (EXCLUSIVE OR)

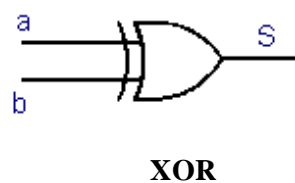
In the case of eXclusive OR, for $S = 1$, $A \text{ OR } B$ must be exclusively 1, i.e. the output will be equal to 1 if the two inputs are in different states (one equal to 1 and the other to zero).

We then write that $F = A \oplus B$ which is expressed as F equals A exclusive OR B.

The sign \oplus is the symbol for XOR (exclusive OR) in logic equations.

Truth table

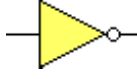
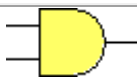
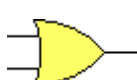
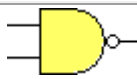
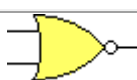
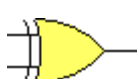
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



Note

- AND , OR , NAND , NOR gates can have more than two inputs.
- There is no exclusive OR with more than two inputs.

SUMMARY TABLE

Basic operators	NON (NOT) Inversion -  /a	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>/A</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	/A	0	1	1	0	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>A.B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	A.B	0	0	0	0	1	0	1	0	0	1	1	1	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>A+B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	A+B	0	0	0	0	1	1	1	0	1	1	1	1									
	A		/A																																														
	0		1																																														
1	0																																																
A	B	A.B																																															
0	0	0																																															
0	1	0																																															
1	0	0																																															
1	1	1																																															
A	B	A+B																																															
0	0	0																																															
0	1	1																																															
1	0	1																																															
1	1	1																																															
ET (AND) .  a.b																																																	
OU (OR) +  a+b																																																	
Derived operators	NAND (NOT-AND) /  a/b	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>A/B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	A/B	0	0	1	0	1	1	1	0	1	1	1	0	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>A↓B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	A↓B	0	0	1	0	1	0	1	0	0	1	1	0	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>A ⊕ B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	A ⊕ B	0	0	0	0	1	1	1	0	1	1	1	0
	A		B	A/B																																													
	0		0	1																																													
0	1	1																																															
1	0	1																																															
1	1	0																																															
A	B	A↓B																																															
0	0	1																																															
0	1	0																																															
1	0	0																																															
1	1	0																																															
A	B	A ⊕ B																																															
0	0	0																																															
0	1	1																																															
1	0	1																																															
1	1	0																																															
NOR (NOT-OR) ↓  a↓b																																																	
XOR ou exclusif ⊕  a ⊕ b																																																	
	NOT, NON	AND, ET	OR, OU																																														
	NAND, ET-NON	NOR, OU-NON	XOR, OUX																																														