Badji-Mokhtar University. Annaba
Faculty of Technology
Department of Computer Science

Chapter 4: Boolean Algebra and Combinational circuits

Dr. Khelifi Hakima

S1- 2025/2026

Lessons Objectives

- ✓ Definition
- ✓ Fundamental concepts
- ✓ Laws of the Boolean Algebra
- ✓ Truth table and Logic gates
- ✓ Design steps of a combinational circuit

Definition

Definition

- Boolean algebra was introduced in 1847 by Georges Boole.
- It is a branch of mathematics that deals with operations on logical values with binary variables.
- These variables are represented as binary numbers to represent truths: 1 = true and 0 = false.
- It forms the basis for designing digital circuits and performing logical operations.

Fundamental concepts

Fundamental concepts

- **State**: Logical states are represented by 0 and 1.
- **Boolean variable**: can take one of two possible values: 0 or 1, representing false and true. These variables are used in Boolean algebra to represent logical statements or conditions.
- **Boolean function:** is an expression formed using Boolean variables, constants (0 or 1), and logical operations (AND, OR, NOT, etc.). 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).

Laws of the Boolean Algebra

Laws of the Boolean Algebra

- **Identity Law** :
  - ✓ **AND**: A.1 = A (AND with 1 leaves the variable unchanged)
  - ✓ **OR**: A + 0 = A (OR with 0 leaves the variable unchanged)
- **Commutative Law**: operating Boolean variables A and B is similar to operating Boolean variables B and A
  - ✓ **AND**: $A \cdot B = B \cdot A$
  - ✓ **OR**: $A + B = B + A$
- **Associative Law:** the order of performing Boolean operator is illogical as their result is always the same
  - ✓ **AND**: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
  - ✓ **OR**: $A + (B + C) = (A + B) + C$

Laws of the Boolean Algebra

- **Distributive Law** : AND distributes over OR, and OR distributes over AND
  - ✓ **AND over OR:** $A \cdot (B+C)=(A \cdot B)+(A \cdot C)$
  - ✓ **OR over AND:** $A+(B \cdot C)=(A+B) \cdot (A+C)$
- ✓ **Inversion Law**: the complement of the complement of any number is the number itself. Also, A variable ANDed with its complement is 0, and ORed with its complement is 1
  - ✓ $\overline{\overline{A}}$ **= A**
  - ✓ **AND:** $A \cdot \overline{A}=0$
  - ✓ **OR:** $A+ \overline{A} =1$
- **Null Law:** Multiplying by 0 always gives 0, adding 1 always gives 1
  - ✓ **AND:** $A \cdot 0=0$
  - ✓ **OR:** $A+1=1$

Laws of the Boolean Algebra

- **Absorption Law:**
  - ✓ A·(A+B)=A (A ANDed with the OR of A and B is just A)
  - ✓ A+(A·B)=A (A ORed with the AND of A and B is just A)
- **MORGAN'S LAW:**

1. The complement of the product (AND) of two Boolean variables is equal to the sum (OR) of the complement of each Boolean variable: $\overline{A \cdot B} = \overline{A} + \overline{B}$

2. The Complement of the sum (OR) of two Boolean variables is equal to the product(AND) of the complement of each Boolean variable: $\overline{A+B} = \overline{A} \cdot \overline{B}$

Laws of the Boolean Algebra

- **Example 1: $A \cdot (B+C) + A \cdot \overline{C}$**

- **Distributive Law:** $A \cdot B + A \cdot C + A \cdot \overline{C}$
- **Inversion Law:** $A \cdot C + A \cdot \overline{C} = A \cdot (C + \overline{C}) = A \cdot 1$
- **Identity Law:** $A \cdot 1 = A$
- **Absorption Law:** $A \cdot (B + A) = A$
- Final simplified expression = A

Truth table and Logic gates

Truth table and Logic gates

- Logic gate is a fundamental building block of digital circuits.
- It performs a basic logical function on one or more binary inputs to produce a single output.
- Based on the logic operation it carries out, the output is typically a binary value (0 or 1).
- There are several common types of logic gates: **NOT, AND, OR, NOT-AND (NAND) , NOT OR (NOR), Exclusive OR (XOR), Not Exclusive OR (NXOR).**

Truth table and Logic gates

- 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 rows is 2^N, where "N" is the number of inputs
- The value of a logic function is equal to 1 or 0, depending on the values of the logic variables.

Truth table and Logic gates

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

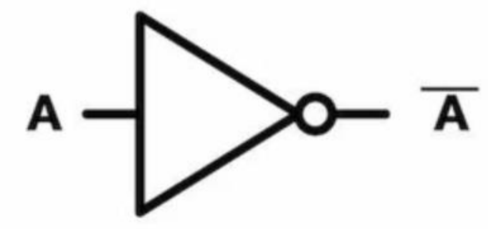
A	B	C	Result
0	0	0	$\bar{A}\,\bar{B}\,\bar{C}$
0	0	1	$\bar{A}\,\bar{B}\,C$
0	1	0	$\bar{A}\,B\,\bar{C}$
0	1	1	$\bar{A}\,B\,C$
1	0	0	$A\,\bar{B}\,\bar{C}$
1	0	1	$A\,\bar{B}\,C$
1	1	0	$A\,B\,\bar{C}$
1	1	1	$A\,B\,C$

Truth table and Logic gates

- The seven basic logic gates are **AND, OR, NOR, XOR, NOT, NAND, and NXOR.**

- **LOGICAL FUNCTION (NOT):** $F=\overline{A}$

A	F
0	1
1	0

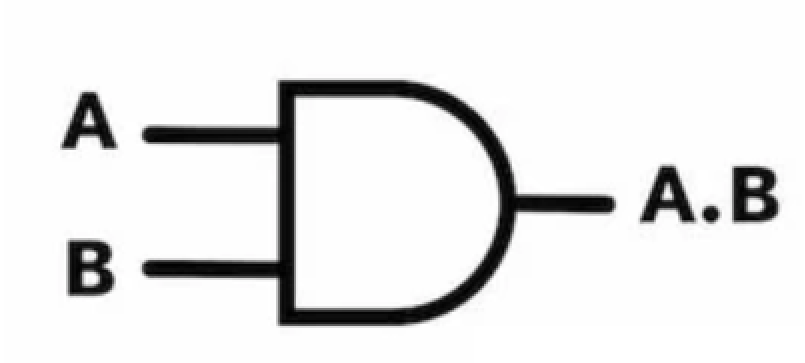


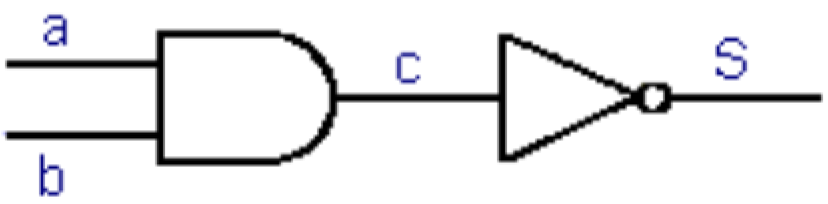
Truth table and Logic gates

- The seven basic logic gates are **AND, OR, NOR, XOR, NOT, NAND, and XNOR.**

- **LOGICAL FUNCTION AND:** F= A*B ou A· B ou AB

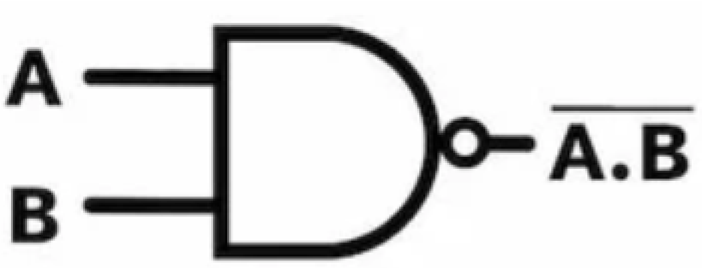
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1





- **LOGICAL FUNCTION NOT AND (NAND):** F= A/B ou $\overline{A \cdot B}$

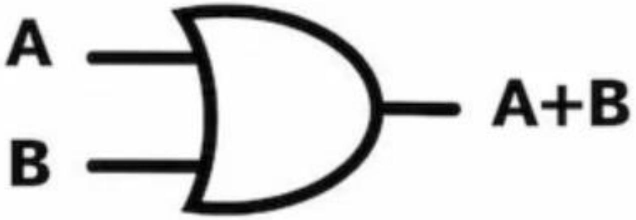
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



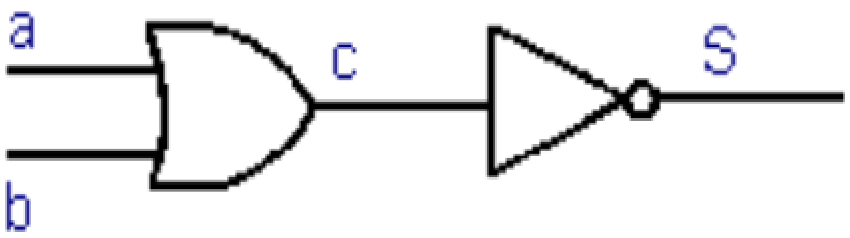
Truth table and Logic gates

- LOGICAL FUNCTION OR: $F = A + B$

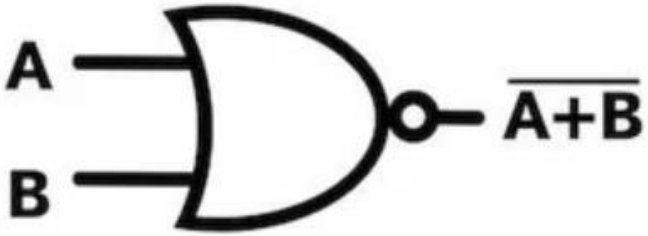
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1



- LOGICAL FUNCTION NOT OR(NOR): $F = \overline{A + B}$



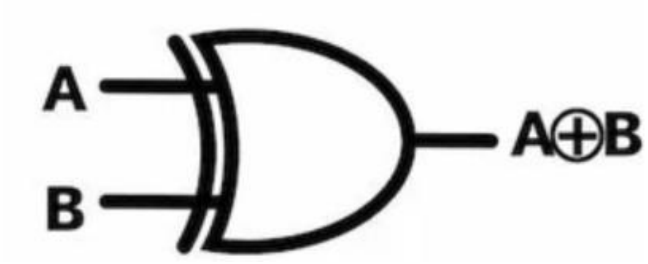
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



Truth table and Logic gates

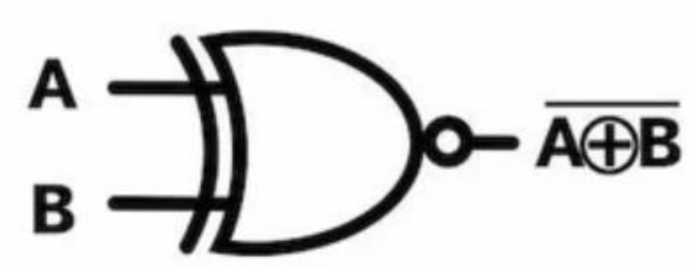
- **LOGICAL FUNCTION XOR (EXCLUSIVE OR):** $F = A \oplus B$

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



- **LOGICAL FUNCTION NXOR:** $F=\overline{A \oplus B}$

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



Truth table and Logic gates

- **Example: F(A, B) =$(A \cdot \overline{B}) + (A + B)$**

A	B	$\overline{B}$	$A \cdot \overline{B}$	A + B	F(A, B) =$(A \cdot \overline{B}) + (A + B)$
0	0	1	0	0	0
0	1	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Design steps of a combinational circuit

Design steps of a combinational circuit

- A combinational circuit is a digital circuit in which the output depends on the present input values.
- It is essentially a mapping from input values to output values based purely on Boolean algebra.
- It is an assembly of logic gates linked together to represent an algebraic expression.
- combinational circuits are used to create other, more complex circuits.

Design steps of a combinational circuit

- Components of a Combinational Circuit:

1. **Logic Gates**: The basic building blocks (AND, OR, NOT, NAND, NOR, XOR, NXOR).
2. **Input**: The variables fed into the circuit.
3. **Output**: The result after processing the inputs with logical operations.
4. **Interconnections**: Wires or connections that link the gates and inputs to outputs.

Design steps of a combinational circuit

- Steps to design a combinational circuit:
  - ✓ **Define the Problem**: understand the requirements of the circuit, including what the inputs and outputs represent and the relationship between them.
  - ✓ **Obtain Truth Table**: create a truth table that shows all possible combinations of input values and the corresponding output values.
  - ✓ **Simplify the Boolean Expression**: can use Boolean algebra to simplify the expressions.
  - ✓ **Select Logic Gates**: Based on the simplified Boolean expressions, choose the appropriate logic gates (AND, OR, NOT, NAND, NOR, XOR, etc.) to implement the circuit.
  - ✓ **Draw the Circuit Diagram**: use standard symbols for gates and connect the inputs, gates, and outputs.

Design steps of a combinational circuit

- Steps to design a combinational circuit:
- **Example**: **Full Adder circuit**
- ✓ **Define the Problem:** A Full Adder takes three binary inputs: **A, B** (the two data bits), and C_{in} (carry input). It produces two outputs: the **sum (S)** and the carry output (C_{out}).
- ✓ **Obtain Truth Table:**

A	B	C_{in}	sum (S)	Carry (C_{out})
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Design steps of a combinational circuit

- Steps to design a combinational circuit:
- **Example**: **Full Adder circuit**
- ✓ **Simplify the Boolean Expression :**
  - **Sum (S)**: The sum can be represented by the XOR operation: S= $A \oplus B \oplus C_{in}$
  - **Carry (C_{out})**: The carry output is generated by the following OR and AND operations: C_{out} =$(A \cdot B)+(B \cdot C_{in})+(A \cdot C_{in})$

A	B	C_{in}	sum (S)	Carry (C_{out})	$A \oplus B$	$A \oplus B \oplus C_{in}$	$A \cdot B$	$B \cdot C_{in}$	$A \cdot C_{in}$
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	0	0
0	1	0	1	0	1	1	0	0	0
0	1	1	0	1	1	0	0	1	0
1	0	0	1	0	1	1	0	0	0
1	0	1	0	1	1	0	0	0	1
1	1	0	0	1	0	0	1	0	0
1	1	1	1	1	0	1	1	1	1

Design steps of a combinational circuit

- Steps to design a combinational circuit:
- **Example**: **Full Adder circuit**

✓ **Select Logic Gates:**

✓ For the sum (S), use two XOR gates: First XOR gate for A⊕B, Second XOR gate for (A⊕B)⊕C_{in}.

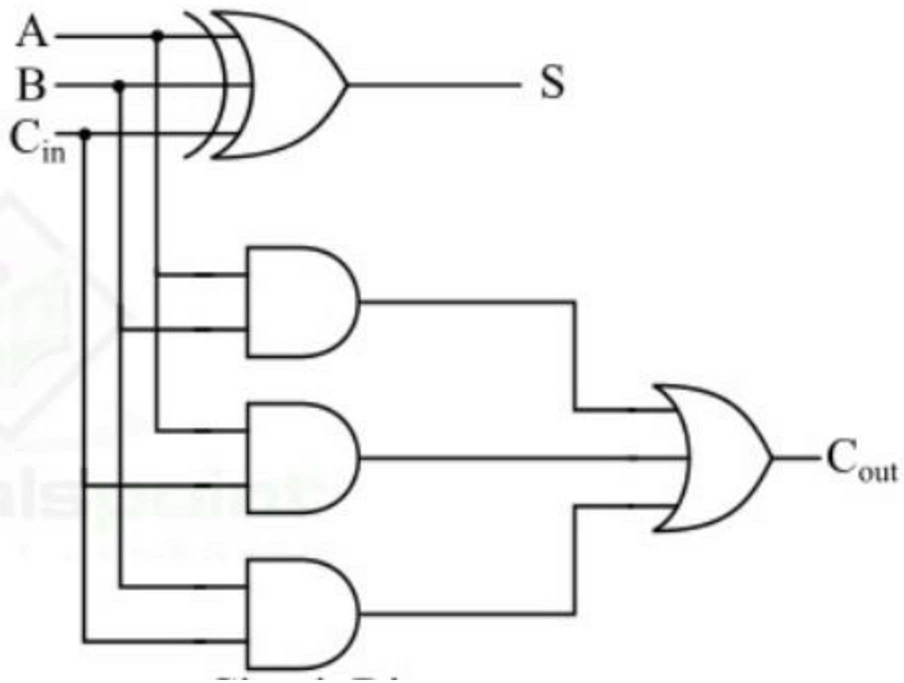
✓ For the carry (C_{out}), use three AND gates and one OR gate:

  ✓ Three AND gates to compute A·B, B· C_{in}, and A· C_{in},

  ✓ One OR gate to combine these AND outputs.

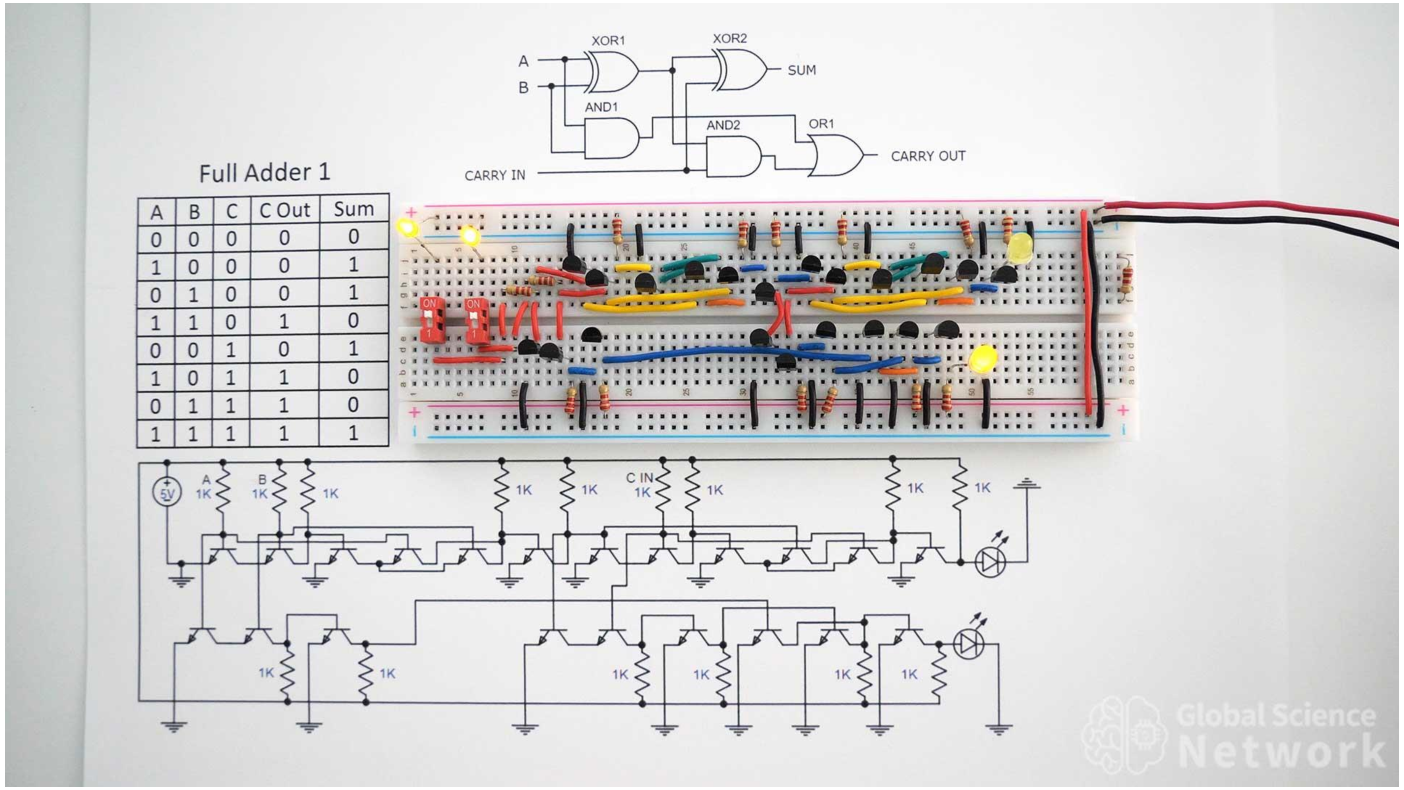
Design steps of a combinational circuit

- Steps to design a combinational circuit:
- **Example**: **Full Adder circuit**
- ✓ **Draw the Circuit Diagram:**



Design steps of a combinational circuit

- Steps to design a combinational circuit:
- **Example**: **Full Adder circuit**
- ✓ **Draw the Circuit Diagram:**



The end