School of Engineering and Computer Science

Mathematics Applied to Digital Engineering-2

(Logic Gates)

Kamel ATTAR
kamel.attar@epita.fr

Week **#4** — **5** ✦ 11/Mars/2024 ✦

★ Logic Gates ★

Basic logic functions

- The package **Truth Tables** and **Boolean Algebra** set out the basic principles of logic.

★ Logic Gates ★

Basic logic functions

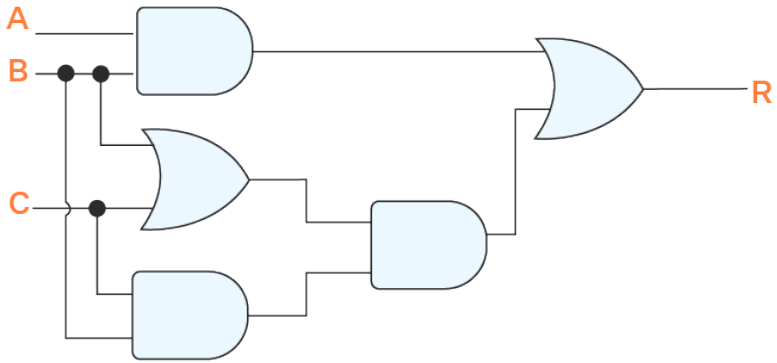
- The package **Truth Tables** and **Boolean Algebra** set out the basic principles of logic.
- Any Boolean algebra operation can be associated with an electronic circuit in which the inputs and outputs represent the statements of Boolean algebra.

★ Logic Gates ★

Basic logic functions

- The package **Truth Tables** and **Boolean Algebra** set out the basic principles of logic.
- Any Boolean algebra operation can be associated with an electronic circuit in which the inputs and outputs represent the statements of Boolean algebra.

- A logic gate is a building block for any digital circuit.
- These logic gates need to make the decision of combining various inputs according to some logical operation and produce an output.

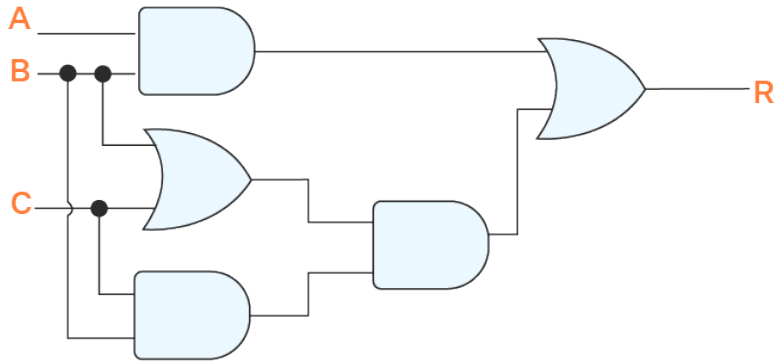


★ Logic Gates ★

Basic logic functions

- The package **Truth Tables** and **Boolean Algebra** set out the basic principles of logic.
- Any Boolean algebra operation can be associated with an electronic circuit in which the inputs and outputs represent the statements of Boolean algebra.

- A logic gate is a building block for any digital circuit.
- These logic gates need to make the decision of combining various inputs according to some logical operation and produce an output.

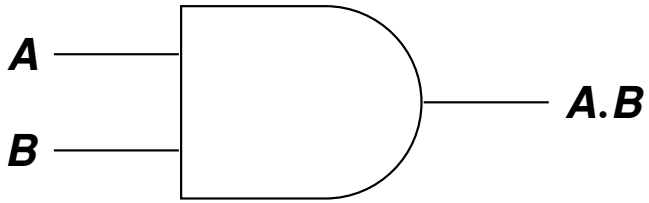


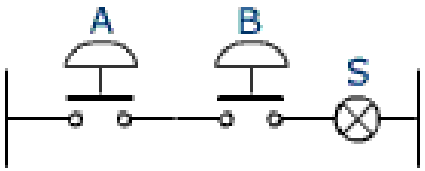
- Logic gates perform logical operations based on boolean algebra.
- Although, digital circuits may be complex, they may all be constructed from three basic devices. These are the **AND gate**, the **OR gate** and the **NOT gate**.

★ Basic logic functions ★

AND gate

- The AND gate is composed of two or more inputs and a single output, and performs logical multiplication.





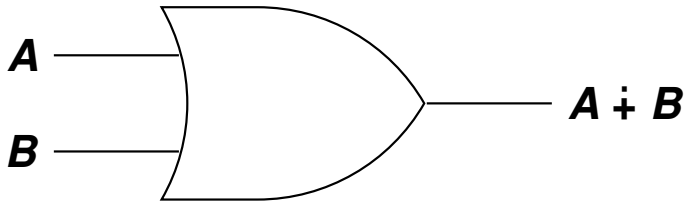
- In the AND operation, a result of **1** occurs when all the input variables are **1**.

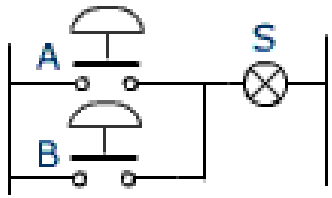
Truth Table of AND gate		
A	***B***	***S = A · B***
1	**1**	**1**
1	**0**	**0**
0	**1**	**0**
0	**0**	**0**

★ Basic logic functions ★

OR gate

- The OR gate is composed of two or more inputs and a single output, and performs logical addition.





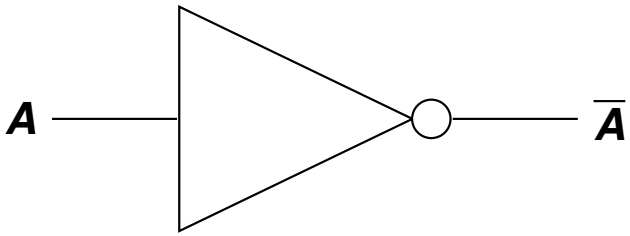
- According to the OR operation, a result of 1 is obtained when any of the input variable is 1.
- In addition, the OR operation produces a result of 0 only when all the input variables are 0.

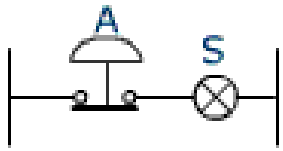
Truth Table of OR gate		
A	***B***	***S = A + B***
1	**1**	**1**
1	**0**	**1**
0	**1**	**1**
0	**0**	**0**

★ Basic logic functions ★

NOT gate

- The NOT operation is unlike the OR and AND operations. This operation can be performed on a single-input variable.
- The NOT gate performs a basic logic function called inversion or complementation.





- The purpose of this gate is to change one logic level (HIGH / LOW) to the opposite logic level.

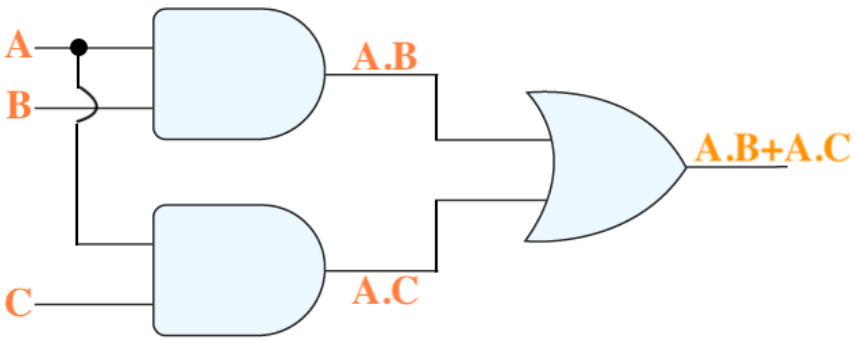
Truth Table of NOT gate	
A	$S = \overline{A}$
1	0
0	1

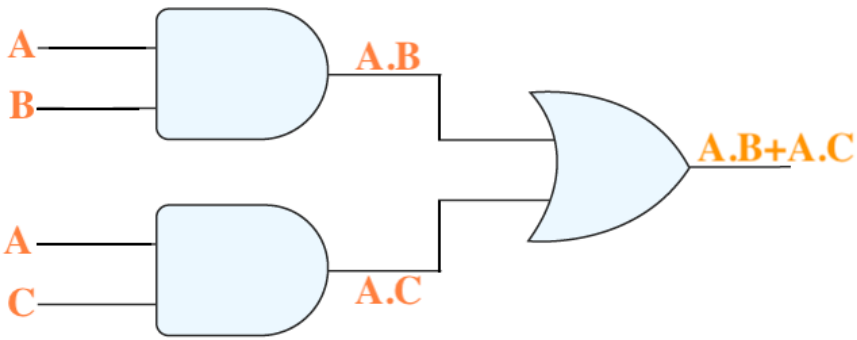
★ Logic circuits and boolean functions ★

Combinations of basic gates

- (**Logic circuits**) combinational circuits can be constructed using a combination of **inverters**, **OR** gates, and **AND** gates to produce a specific output from given inputs.
- When combinations of circuits are formed, some gates may share inputs. This is shown in one of two ways in depictions of circuits.
  - One method is to use branchings that indicate all the gates that use a given input.
  - The other method is to indicate this input separately for each gate.

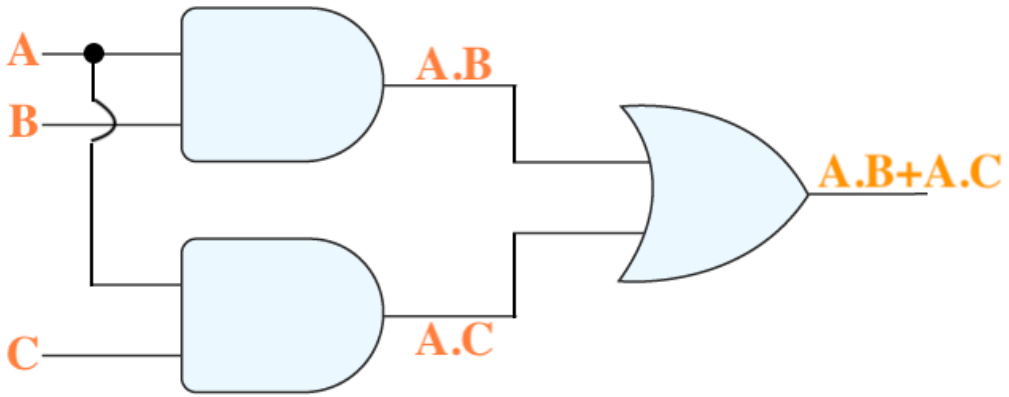
For example, let's consider the following circuit: It contains three inputs ***A***, ***B*** and ***C***





★ Logic circuits and boolean functions ★

Combinations of basic gates



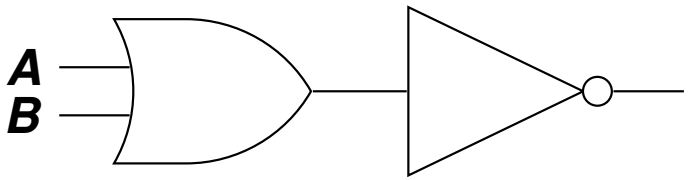
A	B	C	$f(A, B, C)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

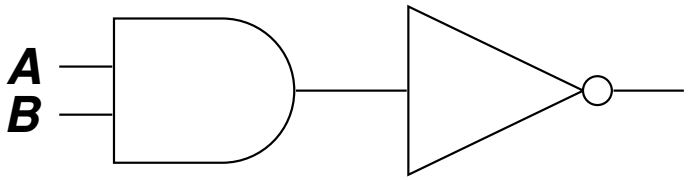
- Here the **boolean function** is $f(A, B, C) = A \cdot B + A \cdot C$
- According to the truth table the **disjunctive normal form** of the boolean function f is $f(A, B, C) = A \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C}$

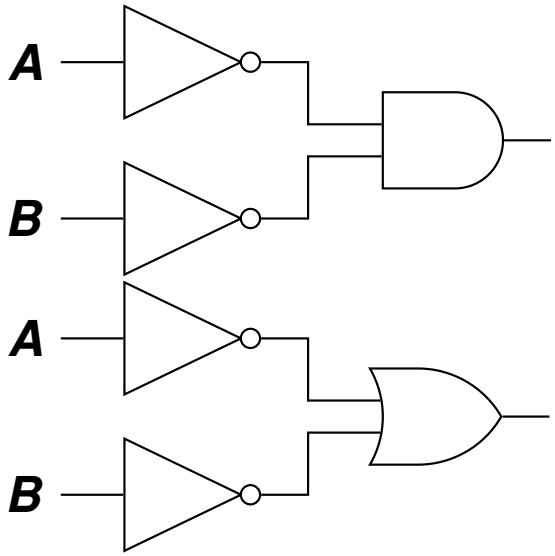
✎ **Exercise 1.** Use logic gates to represent these expressions and draw up the corresponding truth tables.

a. $A(\overline{B} + A)$ *b.* $A + \overline{B}C$ *c.* $B(A + (B + C))$ *d.* $\overline{AB + C}$ *e.* $AB + \overline{C}$

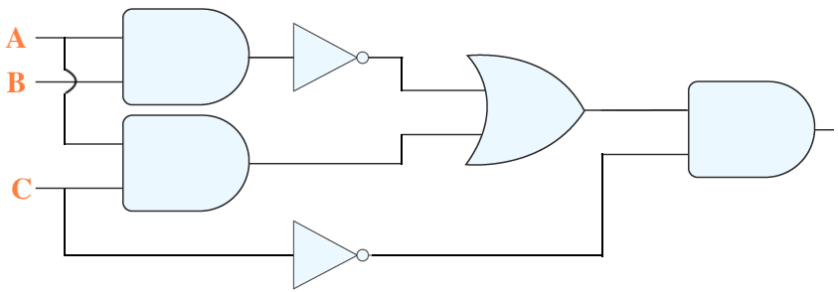
✎ **Exercise 2.** Investigate the relationship between the following circuits. Summarise your conclusions using Boolean expressions for the circuits.

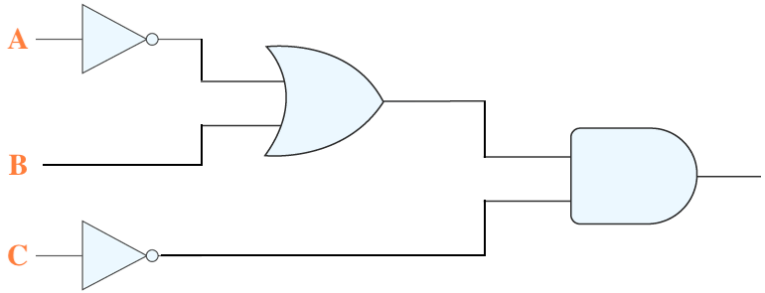


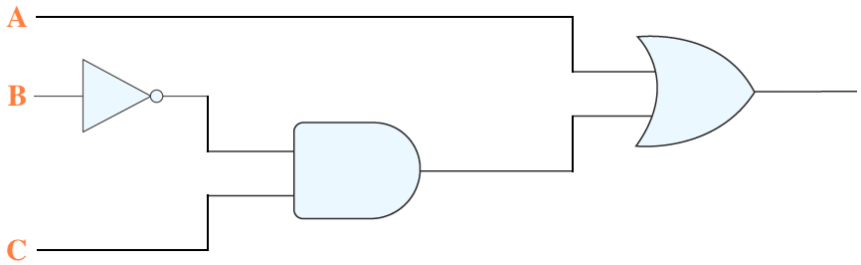


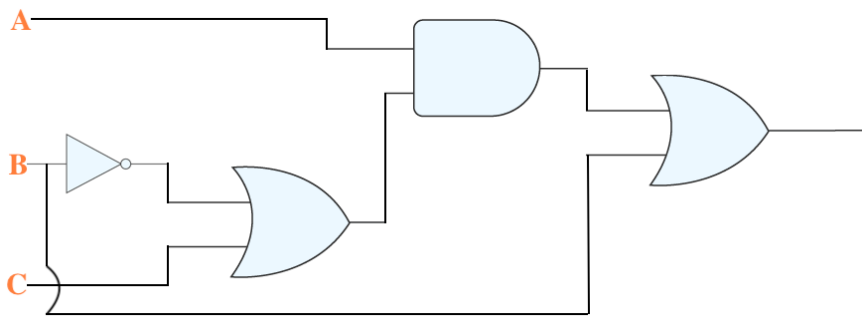


✎ **Exercise 3.** Write down the Boolean expression for each of the circuits below.





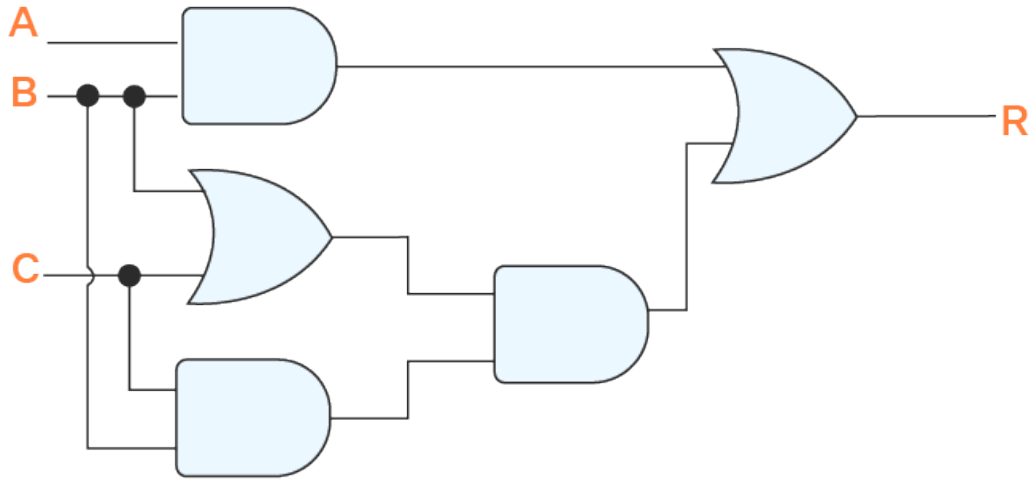




✎ **Exercise 4.** Design a logic circuit whose output is high only when a majority of inputs ***A***, ***B***, and ***C*** are low.

✎ Exercise 5.

a. Find the boolean function ***f*** that corresponds to the following logic circuit.



b. Establish a truth table for the Boolean function ***f***.

c. Find the **disjunctive normal form** of the boolean function ***f***

d. Simplify ***f*** using boolean algebra.

e. Design the simplified logic circuit.

★ Logic circuits and boolean functions ★

Logic circuits from a Specification

A burglar alarm system is to sound if a master switch is on and either a light beam is broken or a pressure pad is stood on.

Draw a logic diagram and a truth table for this system.

- Read the specification carefully. You should notice that it has three inputs. These are:
  - a master switch (M),
  - a light sensor (L),
  - a pressure pad (P).
- It has one output, an alarm bell (B).
- The bell should go to logic **1** if the master switch is at **1** and either the light beam goes to logic **0** or the pressure pad goes to logic **1**. This can be written in Boolean as

$$B = M \cdot (\overline{L} \dot{+} P)$$

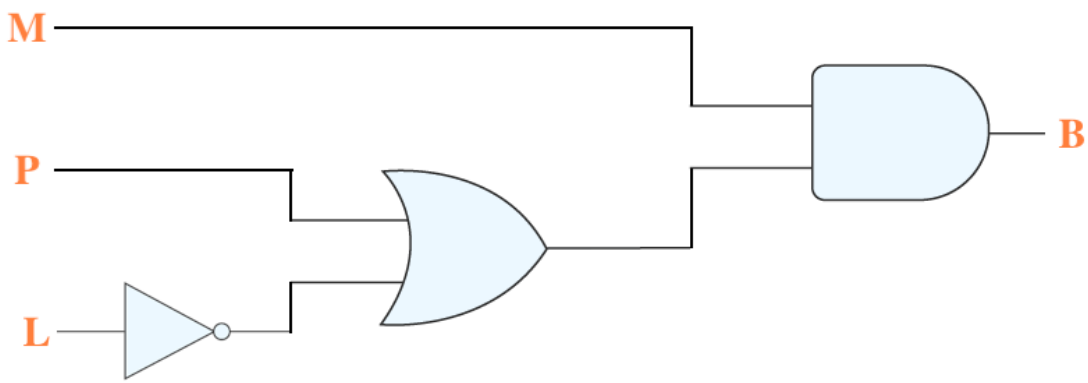
- The truth table for this system is shown in the next slide. All you have to do is read the specification carefully and then read across each row, one at a time, and decide whether the bell should be ringing or not.

★ Logic circuits and boolean functions ★

Logic circuits from a Specification

A burglar alarm system is to sound if a master switch is on and either a light beam is broken or a pressure pad is stood on.

M	*L*	*P*	*B*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



✎ Exercise 6.

a. A house doorbell is to ring (**R**) if a push button at the front door (**F**), a push button at the back door (**B**) or both buttons are operated. Draw a logic diagram and write a Boolean equation.

b. A lift motor is to start only when by closing the door, a switch is actuated (**S**) and a passenger has pressed a button (**B**). Prepare a truth table, a logic diagram and a Boolean equation for this system.

c. The driver of a dustcart is to be able to operate the loading claw by pressing a button (**A**), but only when the senior loader at the rear of the cart has pressed a button to give the 'all clear' (**B**). Draw a logic diagram and write a Boolean equation for this system.

d. An automatic central heating system is to heat the radiators (**R**) if the mains switch (**M**) is on, the timing control switch (**T**) is closed and the override button (**O**) is not selected. Draw a logic diagram, truth table and Boolean statement for this system.

e. A drill is to operate if an isolator is closed (**I**), a guard is in place (closing a micro switch) (**G**), either 'HI' (**H**) or 'LOW' (**L**) speed is selected and a foot pedal is operated (**FP**). Draw a suitable logic diagram for this system. Write the Boolean Expression.

f. A large hall has three temperature sensors (**S1**), (**S2**) and (**S3**). A logic system is to operate the radiator when any two of the temperature sensors fall below a preset level. Draw up a truth table for this system and draw a logic diagram.

g. A burglar alarm will operate if the mains switch (**M**) is on and either an electronic beam (**EB**) is broken, a pressure pad (**PP**) is stood on or a window (**W**) is opened. Draw a logic diagram for this system.

✎ Exercise 7.

Sometimes light fixtures are controlled by more than one switch. Circuits need to be designed so that flipping any one of the switches for the fixture turns the light on when it is off and turns the light off when it is on. Design circuits that accomplish this when there are two switches and when there are three switches.

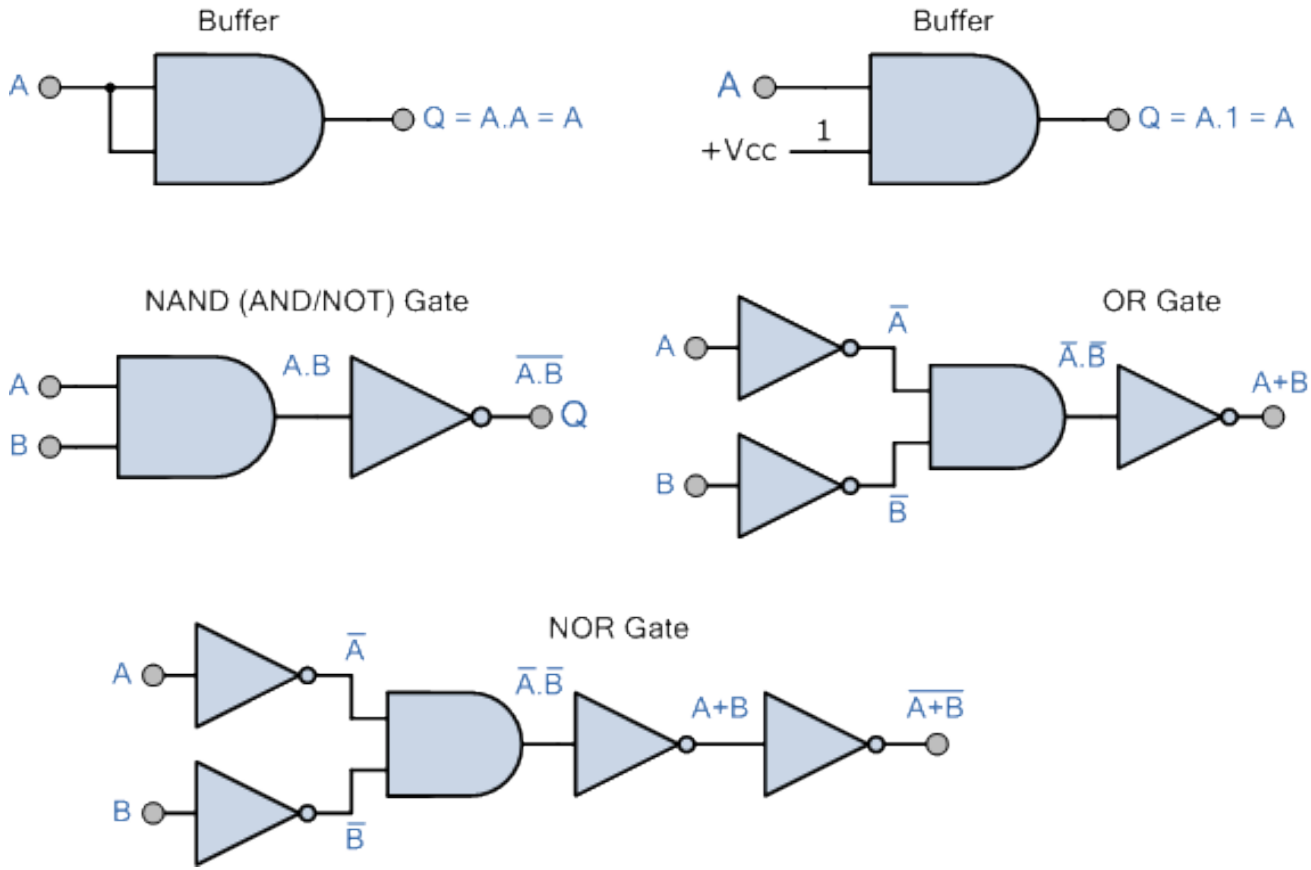
It is possible to produce every other Boolean function using just the set of **AND** and **NOT** gates since the **OR** function can be created using just these two gates.

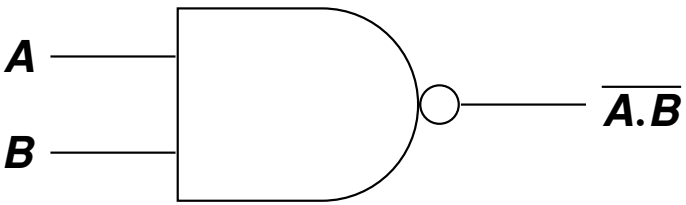
It is possible to produce every other Boolean function using just the set of **AND** and **NOT** gates since the **OR** function can be created using just these two gates.

Using just the **AND** and **NOT** set of logic gates we can create the following Boolean functions and equivalent gates.

Using just the **AND** and **NOT** set of logic gates we can create the following Boolean functions and equivalent gates.

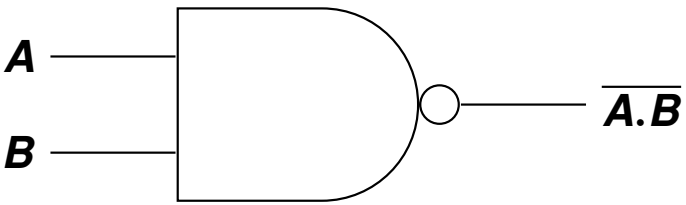
Universal Logic Gates **AND/NOT** Set Equivalents





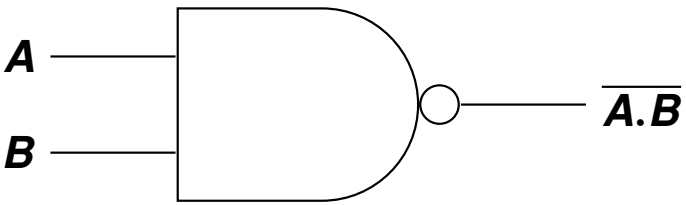
Truth Table of NAND gate		
A	B	$S = \overline{A \cdot B}$
1	1	0
1	0	1
0	1	1
0	0	1

- The term '**NAND**' is formed by the combination of NOT-AND and implies an **AND** function with an **inverted** output.



Truth Table of NAND gate		
A	B	$S = \overline{A \cdot B}$
1	1	0
1	0	1
0	1	1
0	0	1

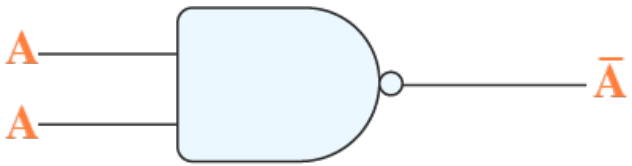
- The term '**NAND**' is formed by the combination of NOT-AND and implies an **AND** function with an **inverted** output.
- The logical operation of the **NAND** gate is such that the output is **LOW** (0) only when all the inputs are **HIGH** (1)



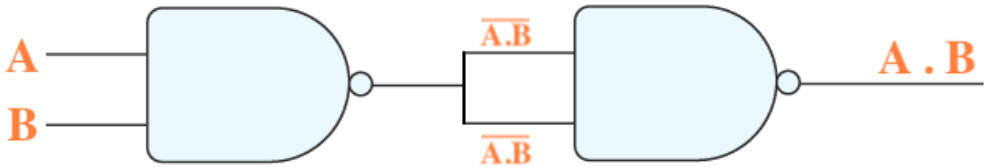
Truth Table of NAND gate		
A	B	$S = \overline{A \cdot B}$
1	1	0
1	0	1
0	1	1
0	0	1

- The term '**NAND**' is formed by the combination of NOT-AND and implies an **AND** function with an **inverted** output.
- The logical operation of the **NAND** gate is such that the output is **LOW** (0) only when all the inputs are **HIGH** (1)
- **NAND** has some very interesting properties. The most important of which is that it's a universal gate. That is, it may be employed to construct an **inverter**, an **AND** gate, an **OR** gate or any combination of these functions.

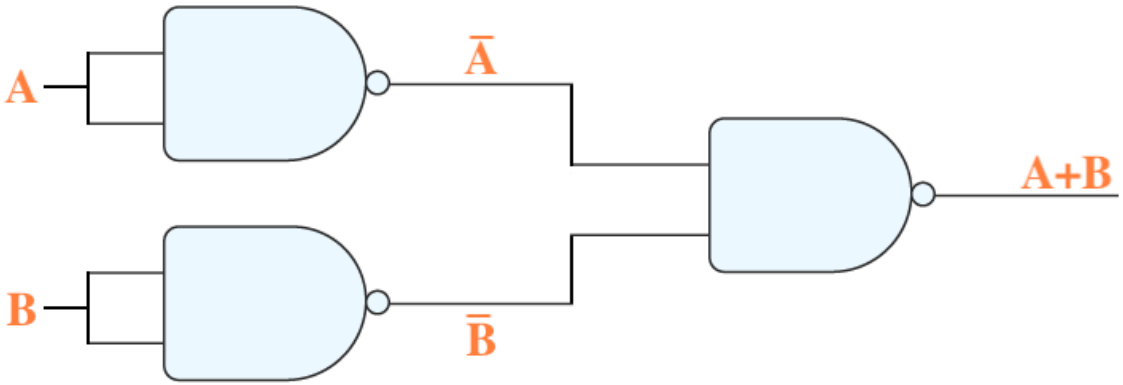
We can create a **NOT** gate by putting the same input twice into a **NAND**. This means we can do **NOT NAND** which is **AND**. Finally we can apply DeMorgan's law to get **OR**.



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

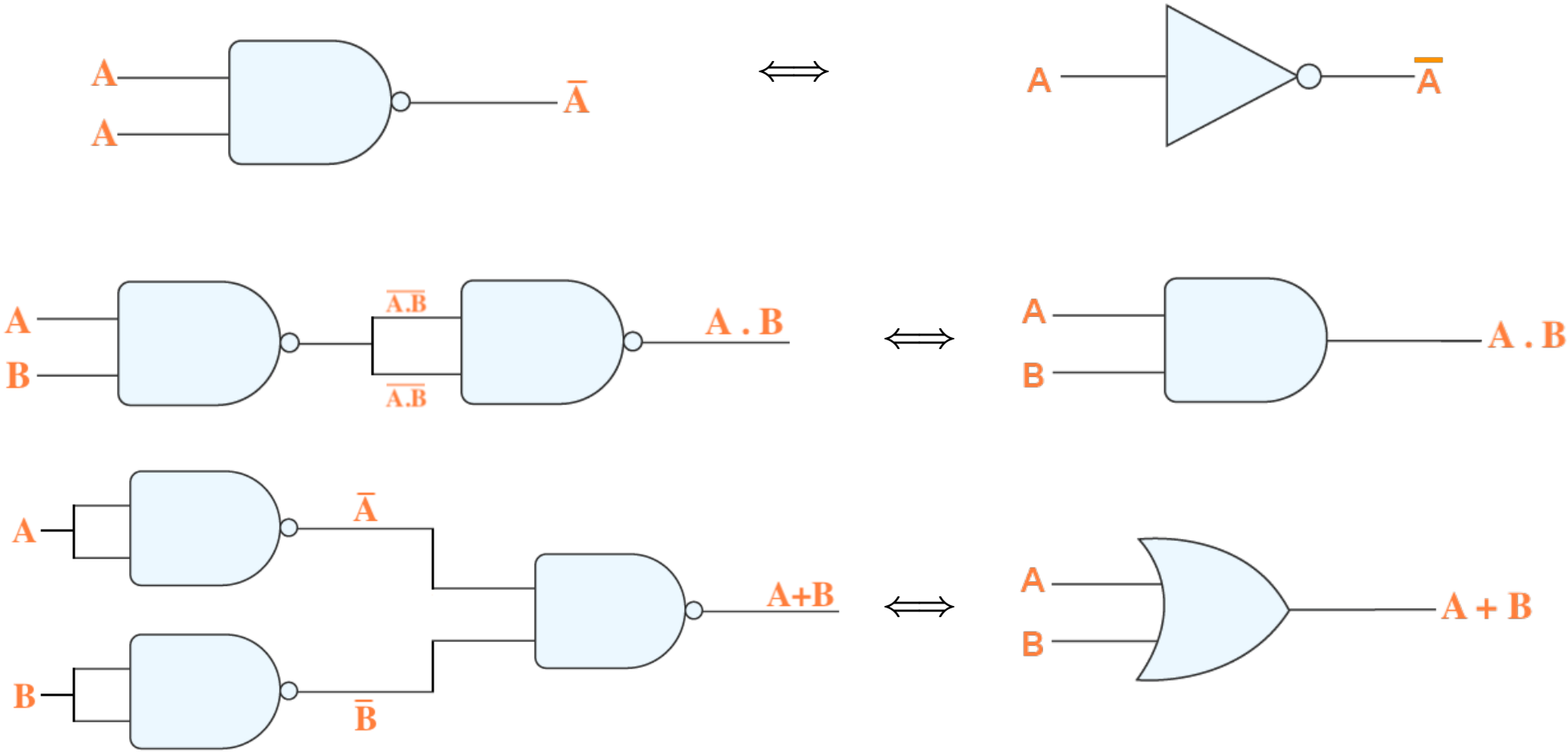


$$\overline{\overline{A \cdot B} \cdot \overline{A \cdot B}} = \overline{\overline{A \cdot B}} + \overline{\overline{A \cdot B}}$$
$$= (A \cdot B) + (A \cdot B)$$
$$= A \cdot B$$



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

We can create a **NOT** gate by putting the same input twice into a **NAND**. This means we can do **NOT NAND** which is **AND**. Finally we can apply DeMorgan's law to get OR.



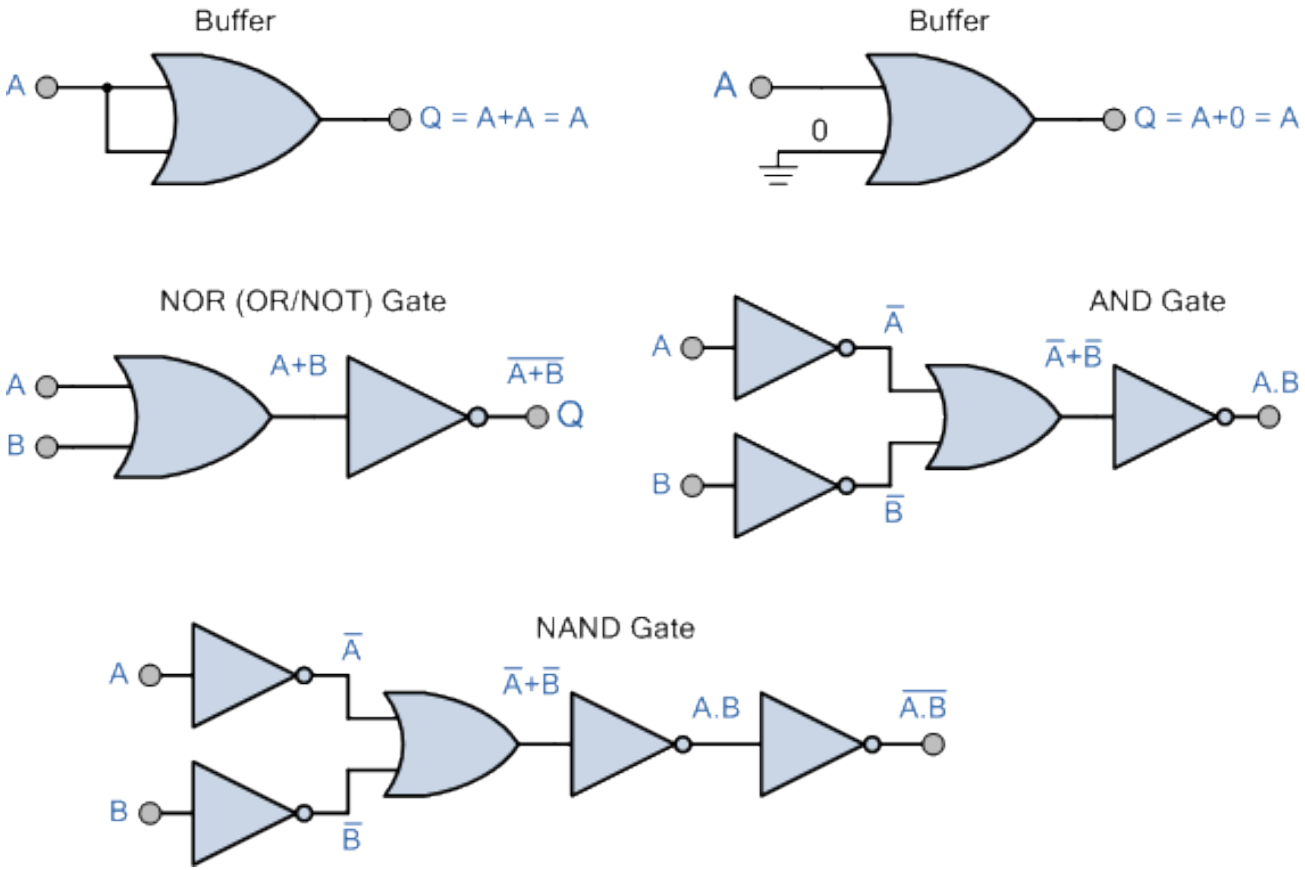
It is possible to produce every other Boolean function using just the set of **OR** and **NOT** gates since the **AND** function can be created using just these two gates.

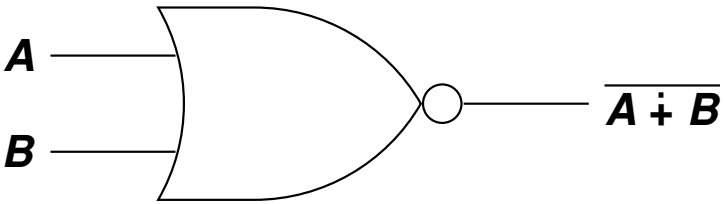
It is possible to produce every other Boolean function using just the set of **OR** and **NOT** gates since the **AND** function can be created using just these two gates.

Using just the **OR** and **NOT** set of logic gates we can create the following Boolean functions and equivalent gates.

Using just the **OR** and **NOT** set of logic gates we can create the following Boolean functions and equivalent gates.

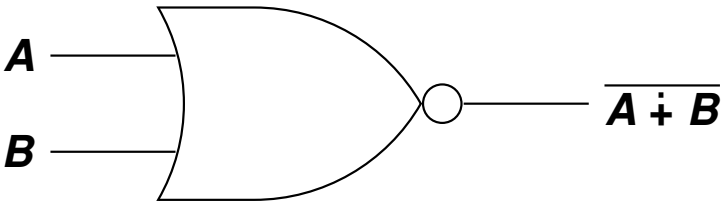
Universal Logic Gates **OR/NOT** Set Equivalents





Truth Table of NOR gate		
A	B	$S = \overline{A \dot{+} B}$
1	1	0
1	0	0
0	1	0
0	0	1

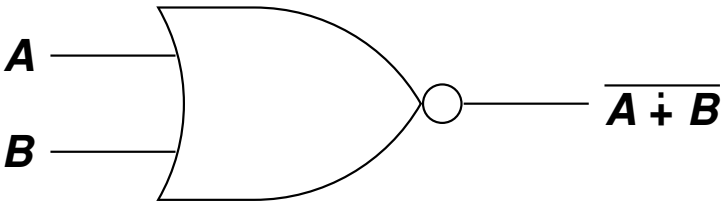
- The term '**NOR**' is formed by the combination of **NOT**-**OR** and implies an **OR** function with an inverted output.



Truth Table of NOR gate		
A	B	$S = \overline{A \dot{+} B}$
1	1	0
1	0	0
0	1	0
0	0	1

- The term '**NOR**' is formed by the combination of **NOT**-**OR** and implies an **OR** function with an inverted output.

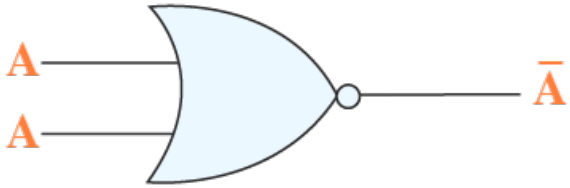
- The logical operation of the **NOR** gate is such that the output is **HIGH** (1) only when all the inputs are **LOW**.



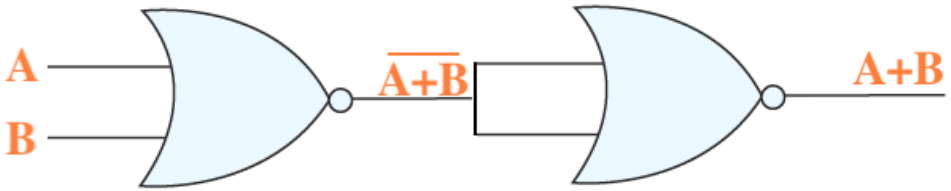
Truth Table of NOR gate		
A	B	$S = \overline{A + B}$
1	1	0
1	0	0
0	1	0
0	0	1

- The term '**NOR**' is formed by the combination of **NOT**-**OR** and implies an **OR** function with an inverted output.
- The logical operation of the **NOR** gate is such that the output is **HIGH** (1) only when all the inputs are **LOW**.
- Just like **NAND**, **NOR** is a universal gate. All gates can be made with **NOR**.

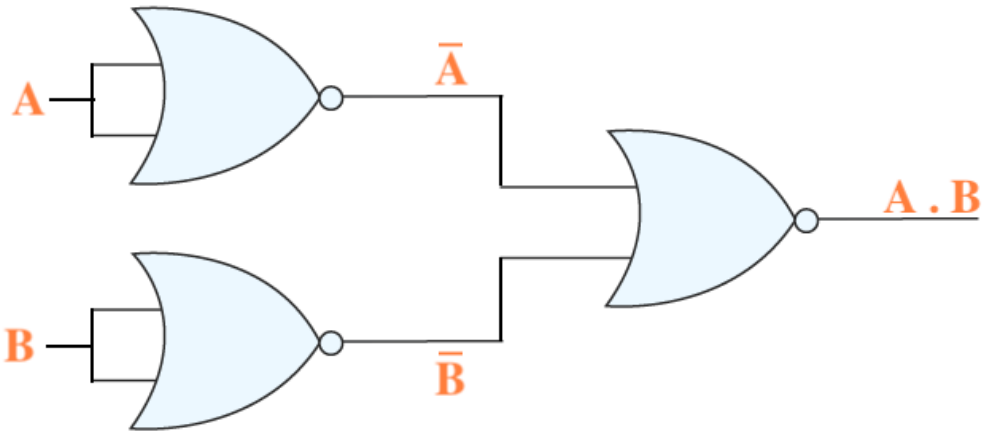
We can create a **NOT** gate by putting the same input twice into a **NOR**. This means we can do **NOT NOR** which is **OR**. Finally we can apply DeMorgan's law to get **AND**.



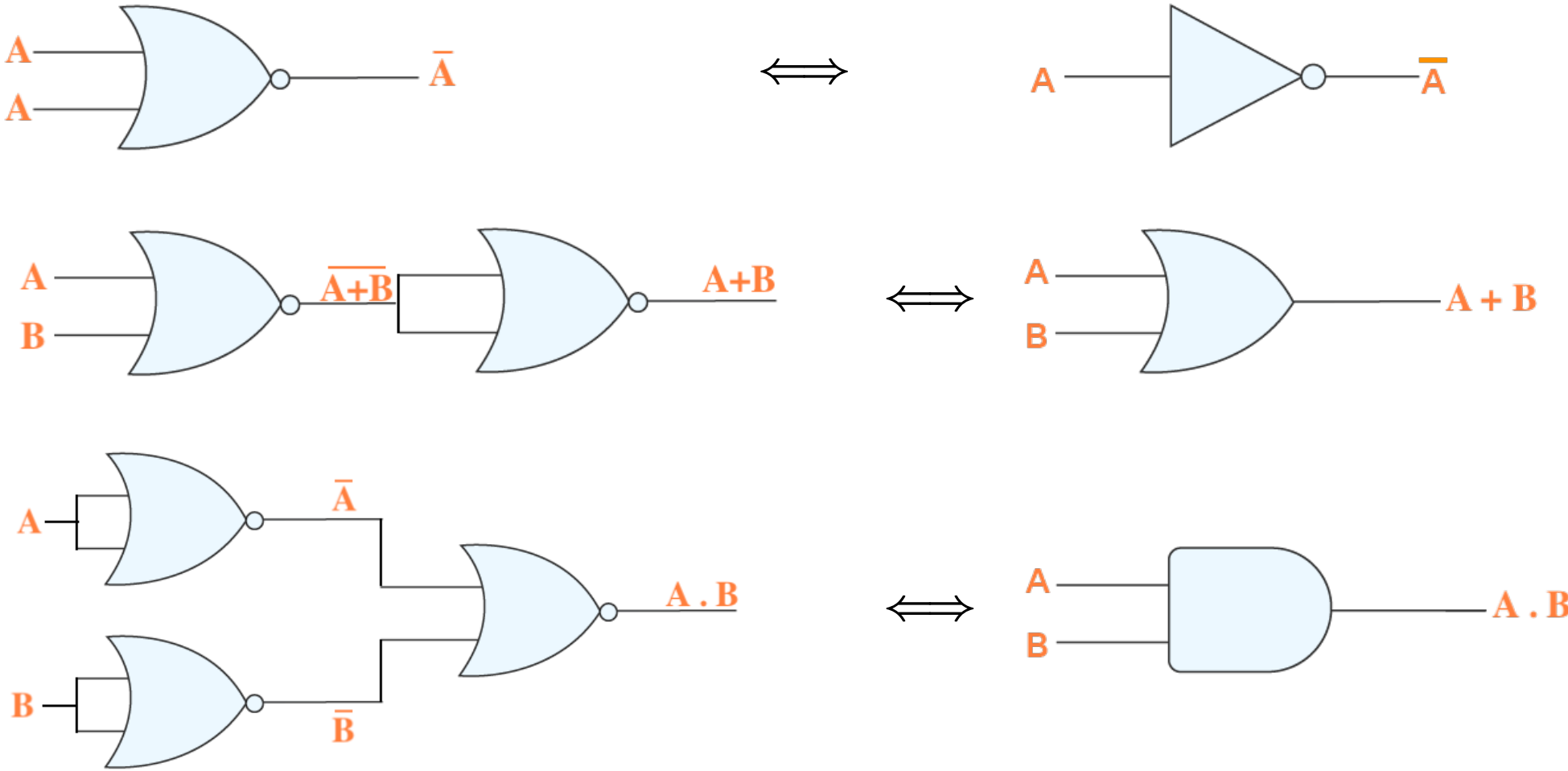
$$\overline{A \dot{+} A} = \overline{A} \cdot \overline{A}$$
$$= \overline{A}$$

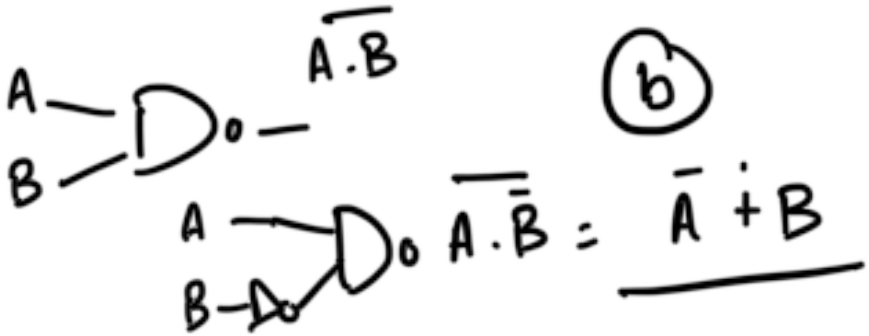


$$\overline{(A \dot{+} B) \dot{+} (A \dot{+} B)} = \overline{A \dot{+} B} \cdot \overline{A \dot{+} B}$$
$$= (A \dot{+} B) \cdot (A \dot{+} B)$$
$$= A \dot{+} B$$



$$\overline{(A \dot{+} A) \cdot (B \dot{+} B)} = \overline{A} \dot{+} \overline{B}$$
$$= \overline{\overline{A}} \cdot \overline{\overline{B}}$$
$$= A \cdot B$$


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

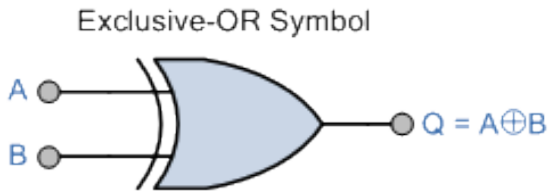


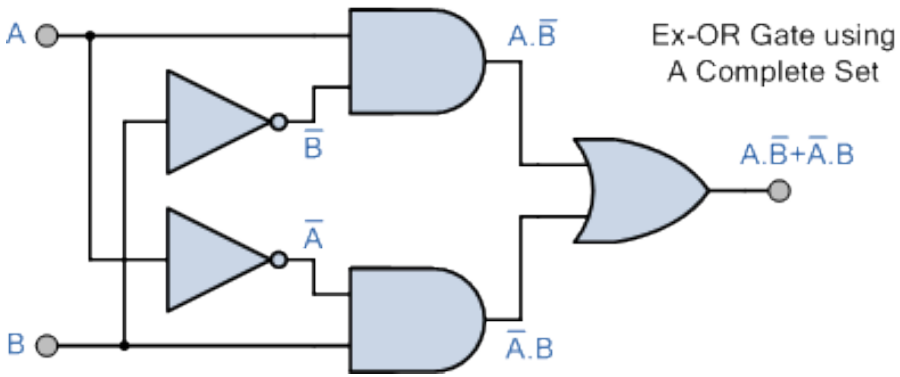
✎ **Exercise 8.** Design circuits for each of the following using only **NAND** gates and then **NOR** gates. a. $\overline{A}B$ b. $\overline{A} + B$ c. $\overline{A} \cdot \overline{B} + \overline{B}$

- Using the full **AND**, **OR** and **NOT** set of logic gates we can create the Boolean expressions for the Exclusive-OR (Ex-OR) gate as shown.

- Using the full **AND**, **OR** and **NOT** set of logic gates we can create the Boolean expressions for the Exclusive-OR (Ex-OR) gate as shown.
- However, because of their functional importance, these gates are treated as basic gates with their own unique symbols.

- Using the full **AND**, **OR** and **NOT** set of logic gates we can create the Boolean expressions for the Exclusive-OR (Ex-OR) gate as shown.
- However, because of their functional importance, these gates are treated as basic gates with their own unique symbols.
- The exclusive-OR is an 'inequality' function and the output is HIGH (1), when the inputs are not equal to each other. Conversely, the exclusive-NOR is an 'equality' function and the output is HIGH (1) when the inputs are equal to each other.



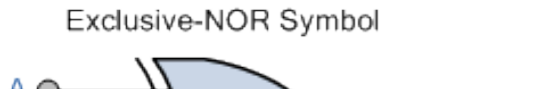


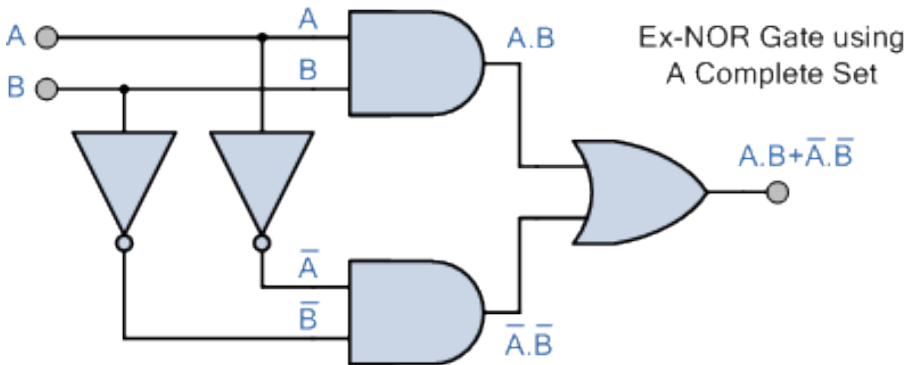
Truth Table of XOR gate		
A	***B***	$A \oplus B$
1	**1**	**0**
1	**0**	**1**
0	**1**	**1**
0	**0**	**0**
$A \oplus B = \overline{A}B + A\overline{B}$		

- These gates are usually formed from the combination of the other logic gates already discussed (**AND**, **OR** and **NOT**).

- These gates are usually formed from the combination of the other logic gates already discussed (**AND**, **OR** and **NOT**).
- However, because of their functional importance, these gates are treated as basic gates with their own unique symbols.

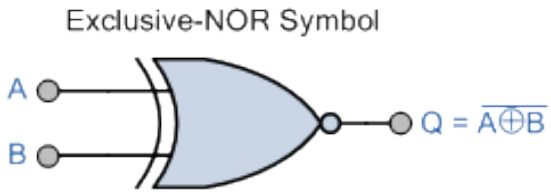
- These gates are usually formed from the combination of the other logic gates already discussed (**AND**, **OR** and **NOT**).
- However, because of their functional importance, these gates are treated as basic gates with their own unique symbols.
- The exclusive-OR is an 'inequality' function and the output is HIGH (1), when the inputs are not equal to each other. Conversely, the exclusive-NOR is an 'equality' function and the output is HIGH (1) when the inputs are equal to each other.

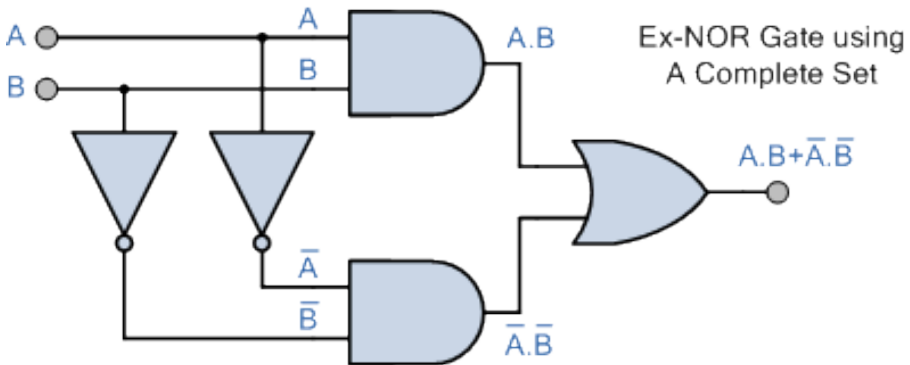




Truth Table of XNOR gate		
A	***B***	$A \odot B$
1	**1**	**1**
1	**0**	**0**
0	**1**	**0**
0	**0**	**1**
$A \odot B = AB + \overline{A}\,\overline{B}$		

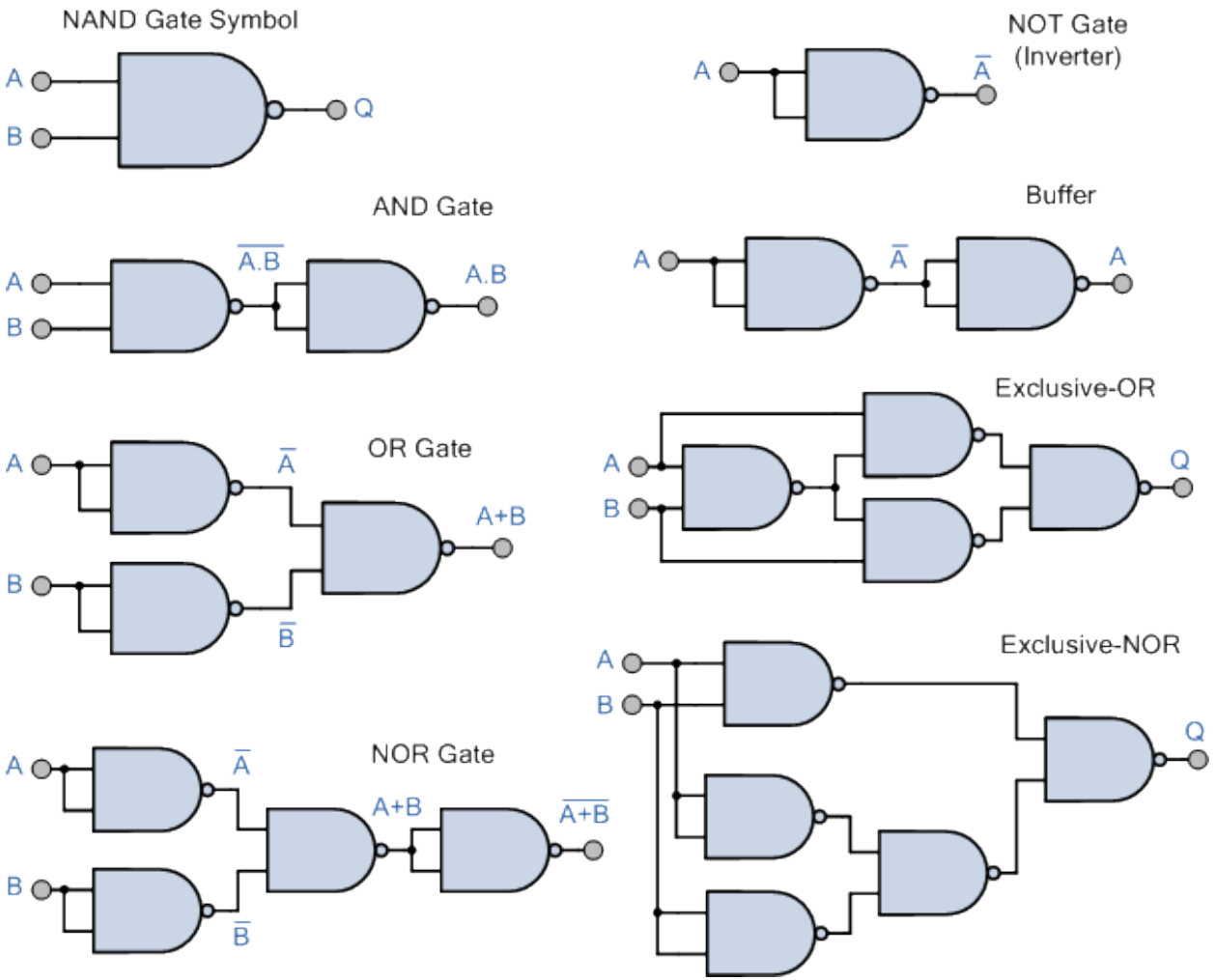
- These gates are usually formed from the combination of the other logic gates already discussed (**AND**, **OR** and **NOT**).
- However, because of their functional importance, these gates are treated as basic gates with their own unique symbols.
- The exclusive-OR is an 'inequality' function and the output is HIGH (1), when the inputs are not equal to each other. Conversely, the exclusive-NOR is an 'equality' function and the output is HIGH (1) when the inputs are equal to each other.
- Note that neither the Exclusive-OR gate or the Exclusive-NOR gate can be classed as a universal logic gate as they can not be used on their own or together to produce any other Boolean function.



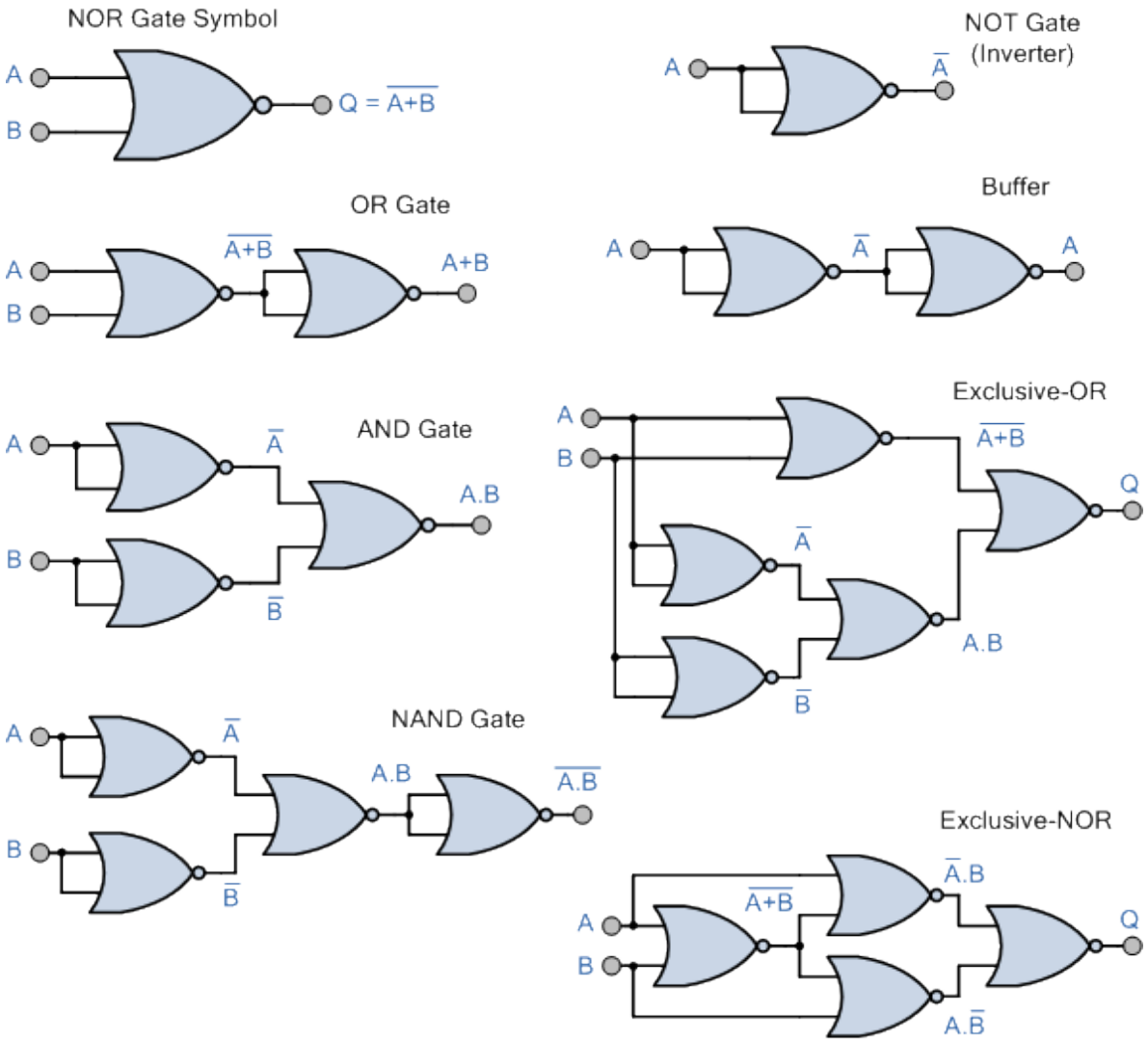


Truth Table of XNOR gate		
A	***B***	***A*** $\odot$ ***B***
1	**1**	**1**
1	**0**	**0**
0	**1**	**0**
0	**0**	**1**
$A \odot B = AB + \overline{A}\,\overline{B}$		

Universal Logic Gates using only NAND Gates

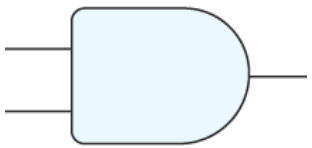
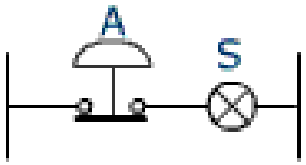
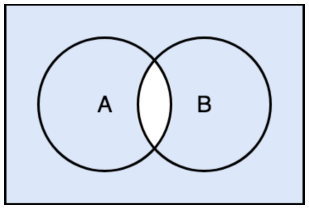


Universal Logic Gates using only NOR Gates

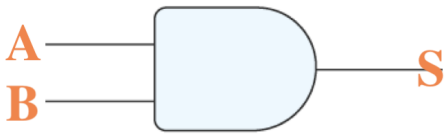


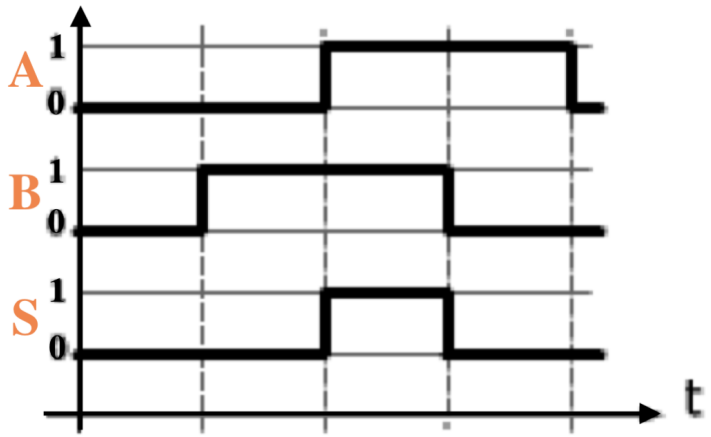
★ Chronogram ★

- We have seen different representations of logical functions:

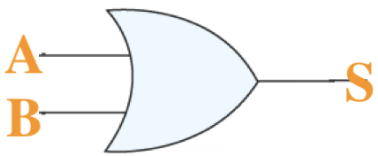
Logic gates	Electrical diagrams	Boolean equations	Truth tables	VENN Diagrams
		$(\overline{A} + B)(\overline{C} + B)$	A | $\overline{A}$ 1 | 0 0 | 1	

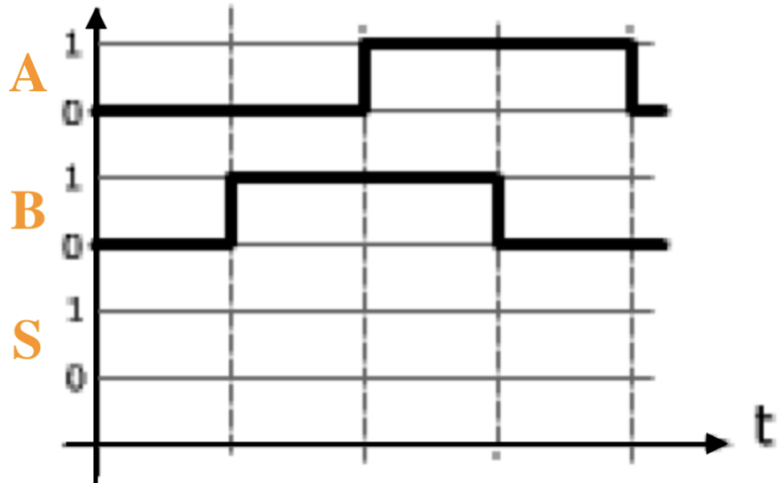
- There is another representation that should help you become even more familiar with logic functions: the **chronogram**.
- A **chronogram** is a temporal diagram, it represents the evolution of logic signals over time.

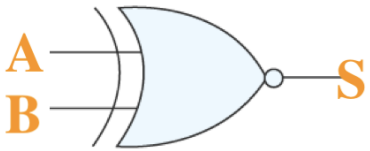


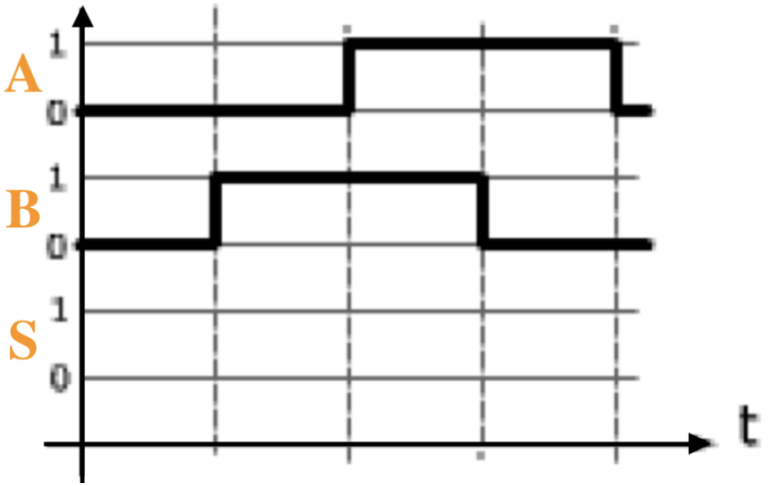


★ Chronogram ★

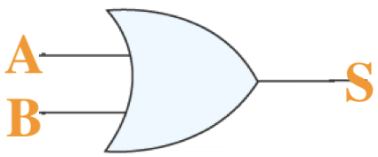


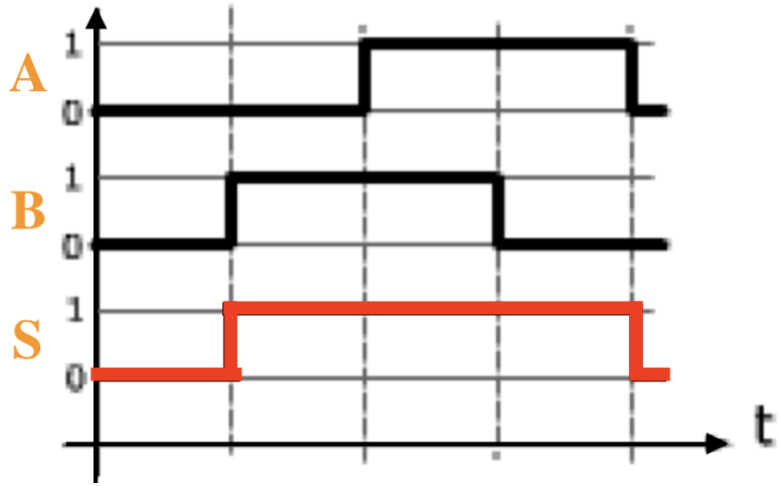


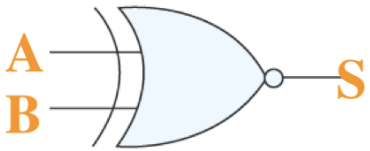


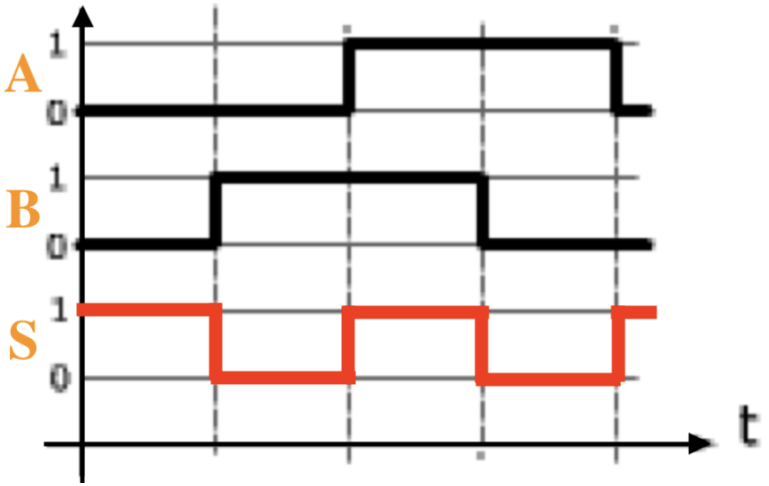


★ Chronogram ★

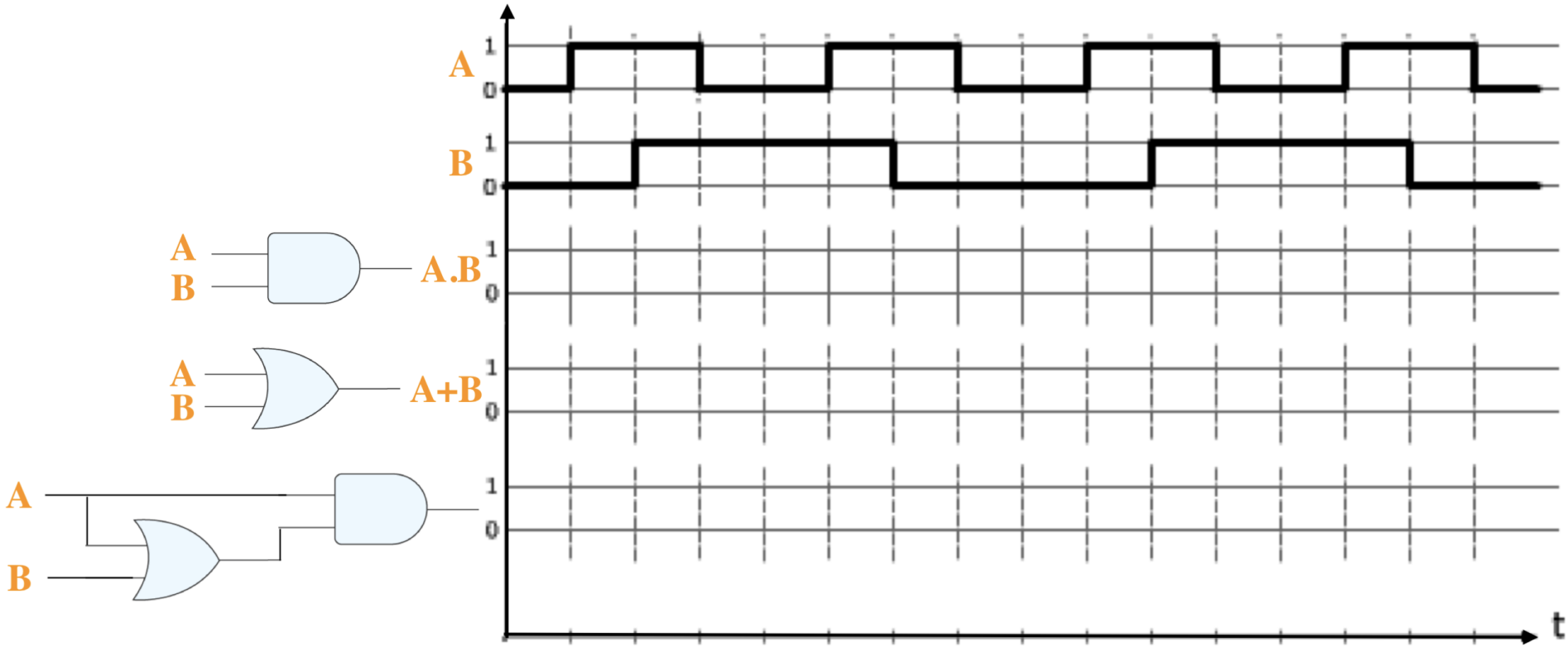






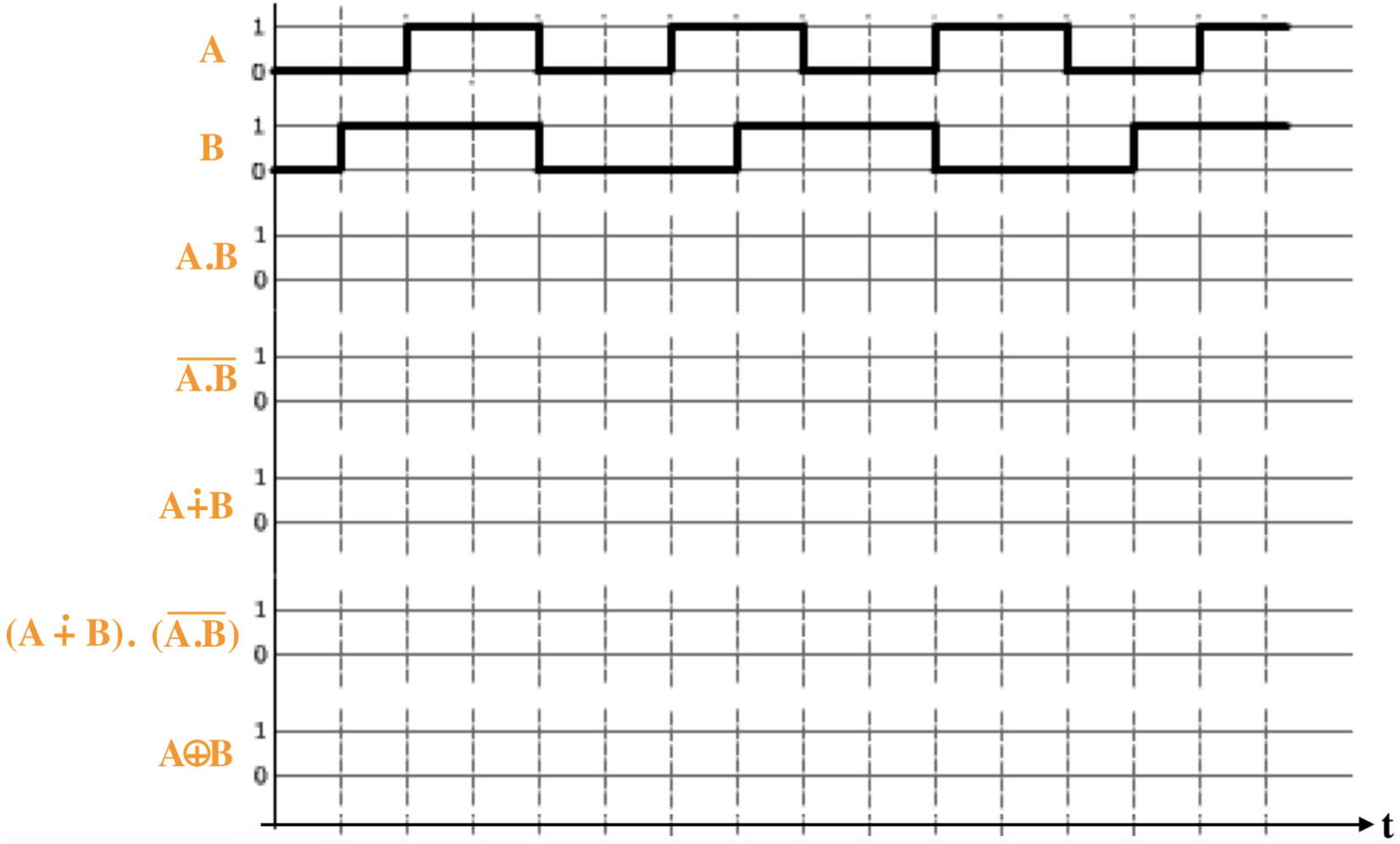


✎ **Exercise 9.** a. Draw the wave diagrams of the following circuits taking into account signals ***A*** and ***B***.



b. Prove by Boolean algebra that $A \cdot (A \dot{+} B) = A$

✎ Exercise 10. a.



b. Prove by Boolean algebra that $(A \dot{+} B) \cdot \overline{A \cdot B} = A \oplus B$


Thank you! Questions?