

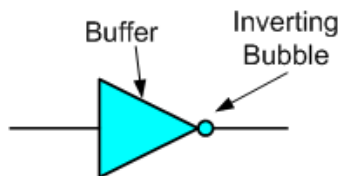
Universal Gates

Lesson Objectives:

In addition to AND, OR, and NOT gates, other logic gates like NAND and NOR are also used in the design of digital circuits.

The NOT circuit inverts the logic sense of a binary signal.

The small circle (bubble) at the output of the graphic symbol of a NOT gate is formally called a negation indicator and designates the logical complement.



The objectives of this lesson are to learn about:

1. Universal gates - NAND and NOR.
2. How to implement NOT, AND, and OR gate using NAND gates only.
3. How to implement NOT, AND, and OR gate using NOR gates only.
4. Equivalent gates.
5. Two-level digital circuit implementations using universal gates only.
6. Two-level digital circuit implementations using other gates.

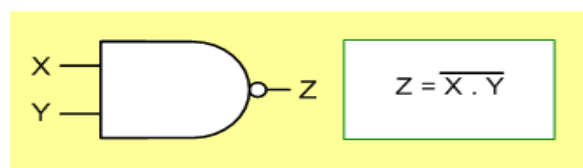
NAND Gate:

The **NAND** gate represents the complement of the AND operation. Its name is an abbreviation of **NOT AND**.

The graphic symbol for the NAND gate consists of an **AND symbol** with a **bubble** on the output, denoting that a complement operation is performed on the output of the AND gate.

The truth table and the graphic symbol of NAND gate is shown in the figure.

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0



The truth table clearly shows that the NAND operation is the complement of the AND.

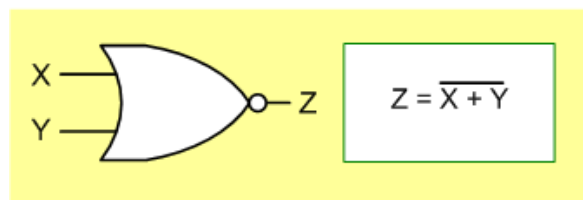
NOR Gate:

The **NOR** gate represents the complement of the OR operation. Its name is an abbreviation of **NOT OR**.

The graphic symbol for the NOR gate consists of an **OR** symbol with a **bubble** on the output, denoting that a complement operation is performed on the output of the OR gate.

The truth table and the graphic symbol of NOR gate is shown in the figure.

X	Y	NOR
0	0	1
0	1	0
1	0	0
1	1	0



The truth table clearly shows that the NOR operation is the complement of the OR.

Universal Gates:

A universal gate is a gate which can implement any Boolean function without need to use any other gate type.

The NAND and NOR gates are universal gates.

In practice, this is advantageous since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families.

In fact, an AND gate is typically implemented as a NAND gate followed by an inverter not the other way around!!

Likewise, an OR gate is typically implemented as a NOR gate followed by an inverter not the other way around!!

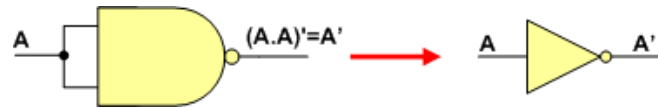
NAND Gate is a Universal Gate:

To prove that any Boolean function can be implemented using only NAND gates, we will show that the AND, OR, and NOT operations can be performed using only these gates.

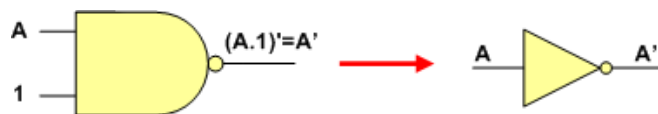
Implementing an Inverter Using only NAND Gate

The figure shows two ways in which a NAND gate can be used as an **inverter (NOT gate)**.

1. All NAND input pins connect to the input signal **A** gives an output **A'**.



2. One NAND input pin is connected to the input signal **A** while all other input pins are connected to logic **1**. The output will be **A'**.



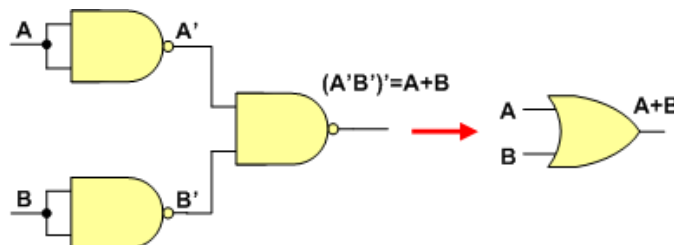
Implementing AND Using only NAND Gates

An **AND gate** can be replaced by NAND gates as shown in the figure (The AND is replaced by a NAND gate with its output complemented by a NAND gate inverter).



Implementing OR Using only NAND Gates

An **OR gate** can be replaced by NAND gates as shown in the figure (The OR gate is replaced by a NAND gate with all its inputs complemented by NAND gate inverters).



Thus, the NAND gate is a universal gate since it can implement the AND, OR and NOT functions.

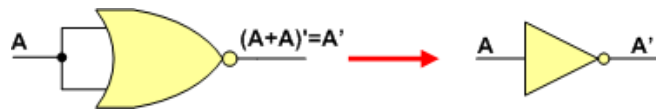
NAND Gate is a Universal Gate:

To prove that any Boolean function can be implemented using only NOR gates, we will show that the AND, OR, and NOT operations can be performed using only these gates.

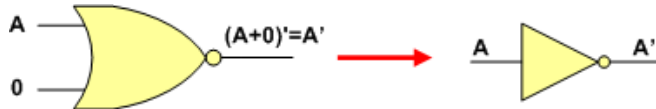
Implementing an Inverter Using only NOR Gate

The figure shows two ways in which a NOR gate can be used as an **inverter (NOT gate)**.

1. All NOR input pins connect to the input signal **A** gives an output **A'**.



2. One NOR input pin is connected to the input signal **A** while all other input pins are connected to logic **0**. The output will be **A'**.



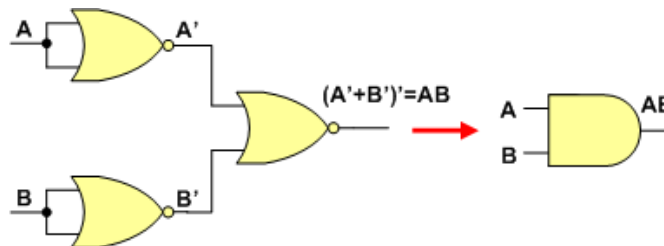
Implementing OR Using only NOR Gates

An **OR gate** can be replaced by NOR gates as shown in the figure (The OR is replaced by a NOR gate with its output complemented by a NOR gate inverter)



Implementing AND Using only NOR Gates

An **AND gate** can be replaced by NOR gates as shown in the figure (The AND gate is replaced by a NOR gate with all its inputs complemented by NOR gate inverters)

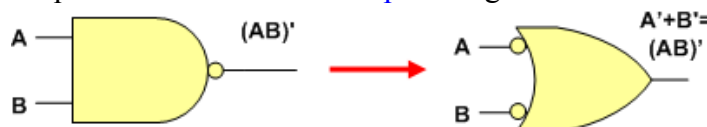


Thus, the NOR gate is a universal gate since it can implement the AND, OR and NOT functions.

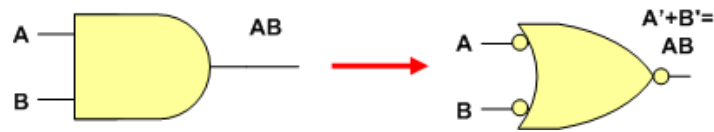
Equivalent Gates:

The shown figure summarizes important cases of gate equivalence. Note that bubbles indicate a complement operation (inverter).

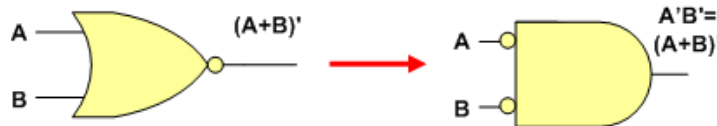
A **NAND gate** is equivalent to an **inverted-input OR gate**.



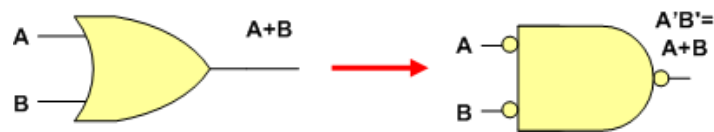
An **AND** gate is equivalent to an **inverted-input NOR** gate.



A **NOR** gate is equivalent to an **inverted-input AND** gate.



An **OR** gate is equivalent to an **inverted-input NAND** gate.



Two **NOT** gates in series are same as a **buffer** because they cancel each other as $A'' = A$.



Two-Level Implementations:

We have seen before that Boolean functions in either SOP or POS forms can be implemented using 2-Level implementations.

For SOP forms AND gates will be in the **first level** and a single OR gate will be in the **second level**.

For POS forms OR gates will be in the **first level** and a single AND gate will be in the **second level**.

Note that using inverters to complement input variables is not counted as a level.

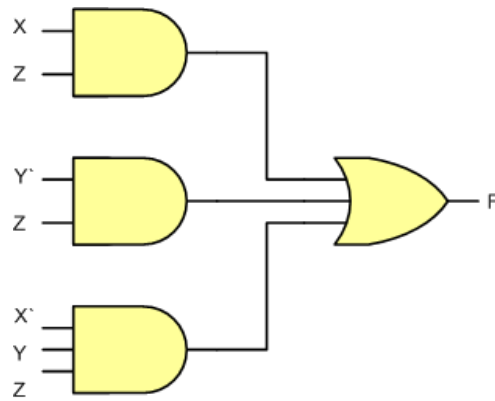
We will show that SOP forms can be implemented using only NAND gates, while POS forms can be implemented using only NOR gates.

This is best explained through examples.

Example 1: Implement the following SOP function

$$F = XZ + Y'Z + X'YZ$$

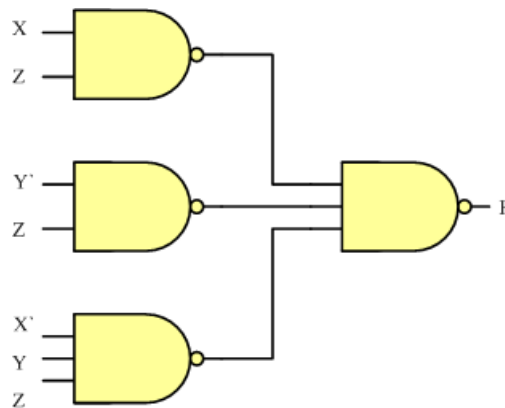
Being an SOP expression, it is implemented in 2-levels as shown in the figure.



Introducing two successive inverters at the inputs of the OR gate results in the shown equivalent implementation. Since two successive inverters on the same line will not have an overall effect on the logic as it is shown before.

[\(see animation in authorware version\)](#)

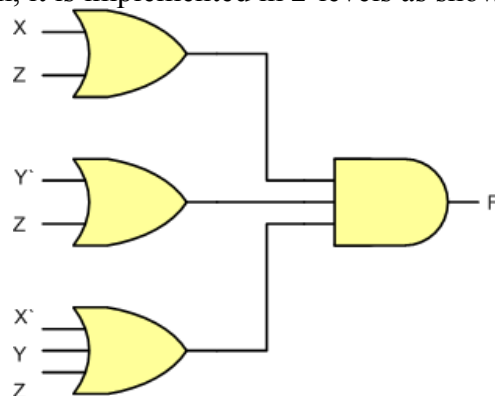
By associating one of the inverters with the output of the first level AND gate and the other with the input of the OR gate, it is clear that this implementation is reducible to 2-level implementation where both levels are NAND gates as shown in Figure.



Example 2: Implement the following POS function

$$F = (X+Z)(Y'+Z)(X'+Y+Z)$$

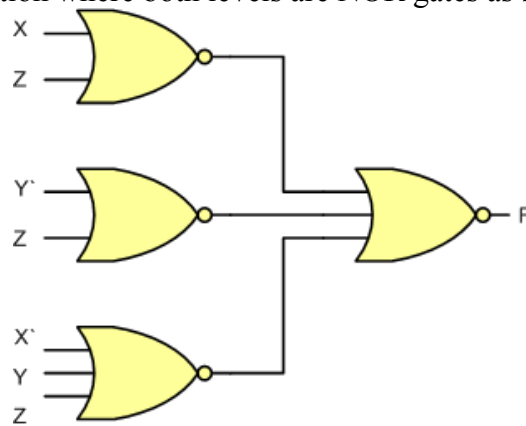
Being a POS expression, it is implemented in 2-levels as shown in the figure.



Introducing two successive inverters at the inputs of the AND gate results in the shown equivalent implementation. Since two successive inverters on the same line will not have an overall effect on the logic as it is shown before.

(see animation in authorware version)

By associating one of the inverters with the output of the first level OR gates and the other with the input of the AND gate, it is clear that this implementation is reducible to 2-level implementation where both levels are NOR gates as shown in Figure.



There are some other types of 2-level combinational circuits which are

- NAND-AND
- AND-NOR,
- NOR-OR,
- OR-NAND

These are explained by examples.

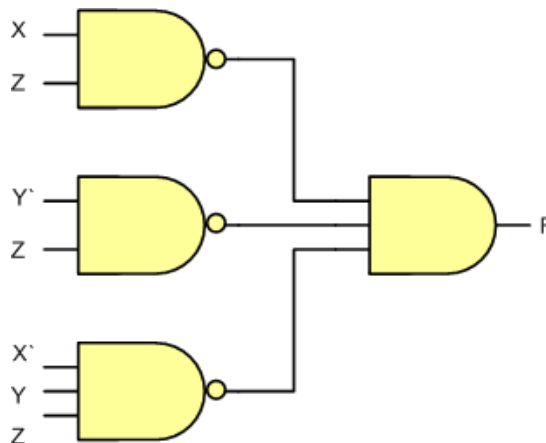
AND-NOR functions:

Example 3: Implement the following function

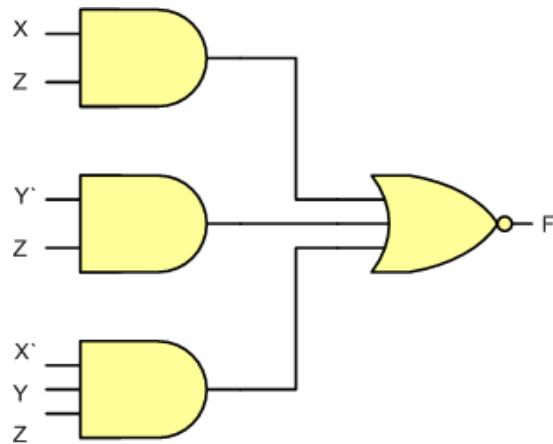
$$F = \overline{XZ} + \overline{YZ} + \overline{XYZ} \text{ or}$$

$$\overline{F} = XZ + YZ + XYZ$$

Since F' is in SOP form, it can be implemented by using NAND-NAND circuit. By complementing the output we can get F , **or** by using *NAND-AND* circuit as shown in the figure.



It can also be implemented using *AND-NOR* circuit as it is equivalent to NAND-AND circuit as shown in the figure. (see animation in authorware version)



OR-NAND functions:

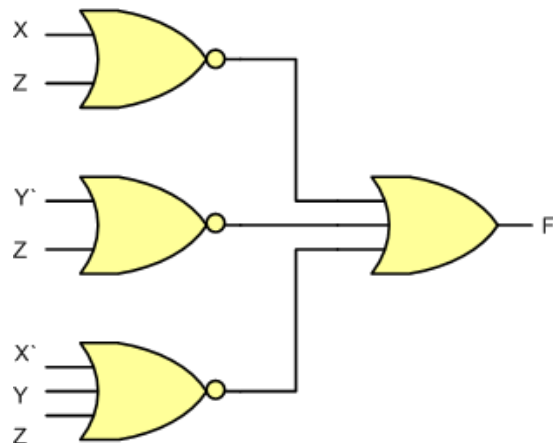
Example 4: Implement the following function

$$F = (X + Z) \cdot (\overline{Y} + Z) \cdot (\overline{X} + Y + Z) \text{ or}$$

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

Since F' is in POS form, it can be implemented by using NOR-NOR circuit.

By complementing the output we can get F , or by using **NOR-OR** circuit as shown in the figure.



It can also be implemented using **OR-NAND** circuit as it is equivalent to NOR-OR circuit as shown in the figure. [\(see animation in authorware version\)](#)

