

Basic Gates and Functions

[Introduction](#)[Truth Tables](#)[Logic Gates](#)

- [AND gate](#)
- [OR gate](#)
- [NOT gate](#)
- [NAND gate](#)
- [NOR gate](#)
- [EOR gate](#)
- [ENOR gate](#)

[Example](#)[Problem](#)[Multiple Input Gates](#)[Tutorials with LabVIEW Simulations](#)[Gates and Functions Quiz](#)

Introduction

Boolean functions may be practically implemented by using electronic gates. The following points are important to understand.

- Electronic gates require a power supply.
- Gate **INPUTS** are driven by voltages having two nominal values, e.g. 0V and 5V representing logic 0 and logic 1 respectively.
- The **OUTPUT** of a gate provides two nominal values of voltage only, e.g. 0V and 5V representing logic 0 and logic 1 respectively. In general, there is only one output to a logic gate except in some special cases.
- There is always a time delay between an input being applied and the output responding.



Truth Tables

[Truth tables](#) are used to help show the function of a logic gate. If you are unsure about [truth tables](#) and need guidance on how to go about drawing them for individual gates or logic circuits then use the [truth table](#) section link.

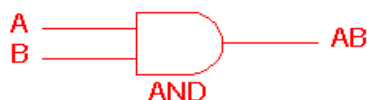
[Truth Tables](#)



Logic gates

Digital systems are said to be constructed by using logic gates. These gates are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR gates. The basic operations are described below with the aid of [truth tables](#).

AND gate



2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

The AND gate is an electronic circuit that gives a **high** output (1) only if **all** its inputs are high. A dot (.) is used to show the AND operation i.e. A.B. Bear in mind that this dot is sometimes omitted i.e. AB

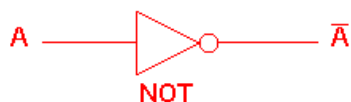
OR gate



2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

The OR gate is an electronic circuit that gives a high output (1) if **one or more** of its inputs are high. A plus (+) is used to show the OR operation.

NOT gate



NOT gate	
A	Ā
0	1
1	0

The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an *inverter*. If the input variable is A, the inverted output is known as NOT A. This is also shown as A', or A with a bar over the top, as shown at the outputs. The diagrams below show two ways that the NAND logic gate can be configured to produce a NOT gate. It can also be done using NOR logic gates in the same way.



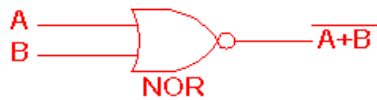
NAND gate



2 Input NAND gate		
A	B	Ā.B̄
0	0	1
0	1	1
1	0	1
1	1	0

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if **any** of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

NOR gate



2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if **any** of the inputs are high.

The symbol is an OR gate with a small circle on the output. The small circle represents inversion.

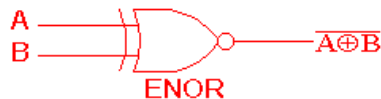
EXOR gate



2 Input EXOR gate		
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

The '**Exclusive-OR**' gate is a circuit which will give a high output if **either, but not both**, of its two inputs are high. An encircled plus sign (\oplus) is used to show the EOR operation.

EXNOR gate



2 Input EXNOR gate		
A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

The '**Exclusive-NOR**' gate circuit does the opposite to the EOR gate. It will give a low output if **either, but not both**, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion.

The NAND and NOR gates are called *universal functions* since with either one the AND and OR functions and NOT can be generated.

Note:

A function in *sum of products* form can be implemented using NAND gates by replacing all AND and OR gates by NAND gates.

A function in *product of sums* form can be implemented using NOR gates by replacing all AND and OR gates by NOR gates.

Table 1: Logic gate symbols

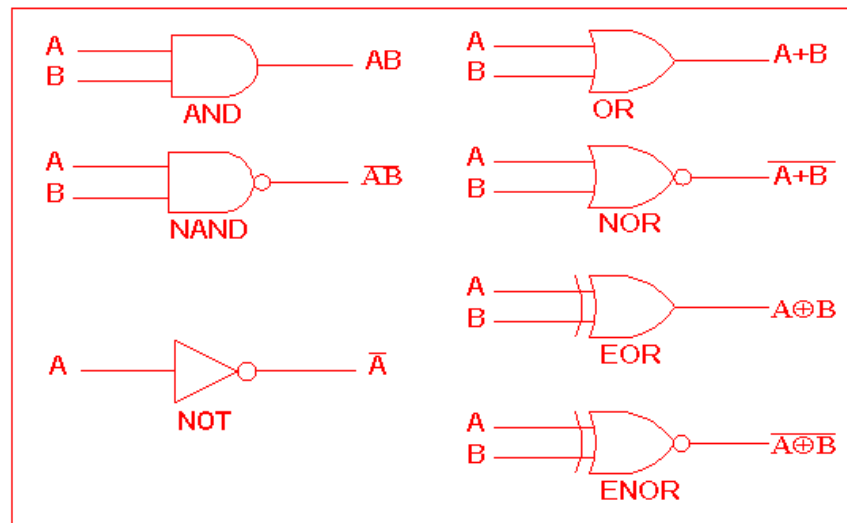


Table 2 is a summary truth table of the input/output combinations for the NOT gate together with all possible input/output combinations for the other gate functions. Also note that a [truth table](#) with 'n' inputs has 2^n rows. You can compare the outputs of different gates.

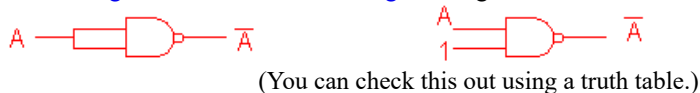
Table 2: Logic gates representation using the Truth table

		INPUTS		OUTPUTS					
		A	B	AND	NAND	OR	NOR	EXOR	EXNOR
NOT gate	A	0	0	0	1	0	1	0	1
	\overline{A}	0	1	0	1	1	0	1	0
	0	1	0	0	1	1	0	1	0
	1	1	1	1	0	1	0	0	1



Example

A [NAND gate](#) can be used as a [NOT gate](#) using either of the following wiring configurations.



Problem

Draw the circuit diagrams like the ones in the [example](#) above to show how a [NOR gate](#) can be made into a [NOT gate](#).

Click [here](#) for answers.



Multiple Input Gates

There are also [multiple input gates](#) if you want to know more about them then click on the link below.

[Multiple Input Gates](#)



Tutorials with LabVIEW simulations

Here are some [tutorials using LabVIEW simulations](#) to show the gate functions and some of the different ways that gates can be configured.


[Tutorials and Simulations](#)



Gates and Functions Quiz

There is a [quiz](#) available to test what you have learned so far. [quiz](#)



To submit your questions and queries please click here: 

Composed by Wale Sangosanya 1997

Updated by David Belton - April 98

Updated by Richard Bigwood 2005