Combinational Logic

The logic gate is the basic building block from which most digital circuits are constructed. Most gates have a number of inputs, and produce a single output. The signals at the input and output terminals are either at a HIGH voltage level or at a LOW voltage level, representing the logical '1' or '0' conditions. Gates will produce one output level according to the combinations of input levels and the type of gate, and the opposite output when any other combination is present at the inputs. In this chapter, you will be introduced to the various types of gates and the ways they can be interconnected to perform different logical functions. The types of gate we will be using are the AND, OR, NOT, NAND, NOR, XOR and XNOR. We will be using the International standard symbols for the gates, as per Logic Gate Simulator, and we will assume positive logic, i.e. a higher voltage is a logical 1 and a lower voltage is a logical 0.

Truth tables list all the possible combinations of the inputs applied to a gate. As we are dealing with 2-state devices, each input can only be at the logic 1 level or the logic 0 level. The output of a logic gate defines its function, so it is normally labelled as 'F'. The number of combinations is always 2^n, where n is the number of inputs. They may be listed in the binary counting order, that is from 0000 to 1111, alternately they may be listed in the reverse order. This is a matter of personal preference, but you should make sure that whichever scheme you chose you are consistent in its use. From a completed truth table a *Boolean* expression can be created to represent the function of a logic circuit.

Combinational logic circuits are those whose output(s) are determined by the logical states of their existing input(s). This is in contrast to *sequential logic* circuits, whose output is set by both the present input *and* the previous output state of the circuit.

Basic Gates

The AND logical function.

The AND function can be demonstrated by the circuit in Figure 1. This shows a lamp connected in series with two switches S1 and S2 and a D.C. voltage supply. For a current to flow through the circuit and light the lamp, both switches must be closed. The operation of the circuit can be described by its truth table that is given in Table 1, where a 0 indicates the switch is open (or off) and a 1 indicates that the switch is closed (or on).

Figure 1

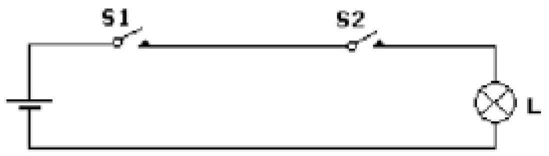


Table 1

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

The Boolean expression for the output F of a 2 input AND gate is given by the equation

$$F = A \bullet B$$

The dot • is the Boolean symbol for the AND logical function but it is often omitted, hence this could have been written

$$F = AB$$

Table 2

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

The output F of an AND gate with 3 inputs A, B and C is

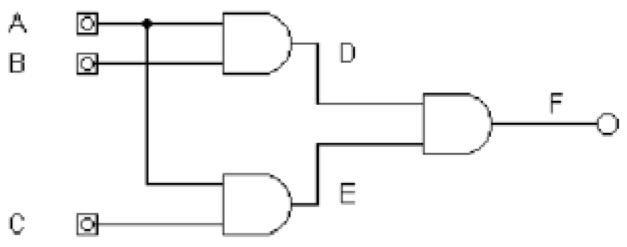
$F = A \bullet B \bullet C$

The truth table of a 3-input AND gate is given in Table 2 and shows that the output is at logical 1 only when A *and* B *and* C are at 1. Always remember that the number of combinations of the input variables is 2^n, where *n* is the number of variables. So a 4 input gate requires 2^4 (16) rows in the truth table.

Activity 1

Figure 2 shows a digital circuit constructed using AND gates. Write down the truth table for the circuit and use this to simplify the circuit.

Fig. 2



The truth table for the circuit shown is given in Table 3. The final column for this truth table (**F = D • E = A • B • C**) is the same as the final column in Table 3.2 for a three input AND gate. Thus, the circuit shown in Figure 2 can be replaced by a single 3-input AND gate. This is the first indication that it is often possible to simplify a logic circuit.

Table 3

A	B	D=A•B	C	E=A•C	F=D·E=A•B•C
0	0		0		
0	0		1		
0	1		0		
0	1		1		
1	0		0		
1	0		1		
1	1		0		
1	1		1		

The OR logical function

Current will flow in the circuit shown in Figure 3 if either switch S1 OR switch S2 OR both are closed, or ON. The only time current will not flow is when BOTH switches are open, or OFF.

An OR gate has two or more inputs and a single output. The output is LOW only when all the inputs are LOW. The truth table for a two-input OR gate is show in Table 4.

Fig. 3

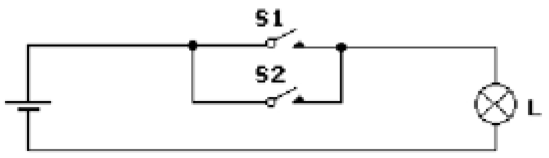


Table 4

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

The Boolean expression for the output F of a 2 input OR gate is given by the equation

$$F = A + B$$

The plus sign + is the Boolean symbol for the OR logical function.

Table 5

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

The output F of an OR gate with 3 inputs A, B and C is

$$F = A + B + C$$

The truth table of a 3-input OR gate is given in Table 5 and shows that the output is at logical 1 only when A *or* B *or* C are at 1.

Activity 2

Write down the truth table for the circuit given in Figure 4 and from this show that the circuit can be obtained in a simpler manner.

Fig. 4

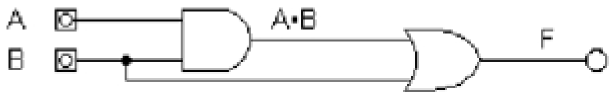


Table 6 shows the truth table for this circuit.

Table 6

A	B	D=A•B	F=D+B=A•B+B
0	0		
0	1		
1	0		
1	1		

The output F of the circuit is always the same as the input B, therefore no gates are required since input B may be directly connected to output F.

Activity 3

Determine the logic circuit corresponding to the truth table given in Table 7

Table 7

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

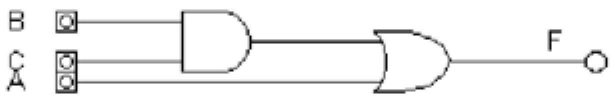
The output F is at logic 1 either if A is 1 OR if B AND C are both 1. This means that inputs B and C are connected to a 2-input AND gate, and the output from this, together with input A, are connected to a **0** 2-input OR gate. The circuit is shown in Figure 5.

$$F = A + (B \bullet C)$$

Note that the AND has a higher priority than the OR so this expression could also be written:

$$F = A + BC$$

Fig. 5



Activity 4

Write down the Boolean equations that describe the logic circuits given in figures 6 and 7. Write down the truth table for each circuit and compare them, and comment on the results.

Fig 6

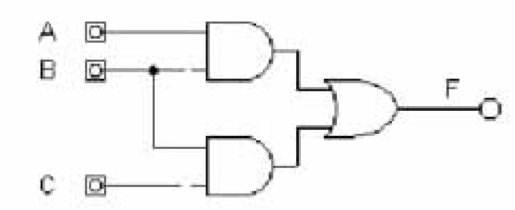


Fig 7

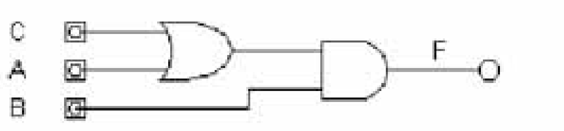


Table 8

A	B	A•B	C	B•C	F
0	0		0		
0	0		1		
0	1		0		
0	1		1		
1	0		0		
1	0		1		
1	1		0		
1	1		1		

$F = A \bullet B + B \bullet C$

Table 9

A	B	C	A+C	F
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

$F = (A + C) \bullet B$

Comparing the columns for F in the two truth tables, it can be seen that these two circuits give identical outputs. Thus, different combinations of gates can produce the same output. In this example circuit 7 uses one less gate than circuit 6.

NOT logical function

A NOT gate has a single input and a single output that is always the *inverse* of the input. If the input is at logic 1, the output will be at logic 0 and *vice versa*. The truth table of the NOT gate is given in table 10. The NOT gate is often known as an inverter and can be said to *complement* a digital signal. The complement of an 8-bit binary number is shown in Table 11.

Table 10

A	F
0	1
1	0

Table 11

Number	01001101
Compliment	10110010

We denote the logical inverse of a value by placing a bar over the value. Hence *A* is the logical inverse of *A*, and is pronounced as "A bar". As you will no doubt be aware the normal

keyboard does not allow for the overbar and you have to use Word's Equation Editor to obtain this feature. Because of this, you will often find people placing a star after the name, as in *A**, to indicate the inverse. Both the overbar and the trailing bar are acceptable.

Activity 5

Write down the truth table of the circuit shown in Figure 8, then write down the Boolean equation representing the circuit.

Fig 8

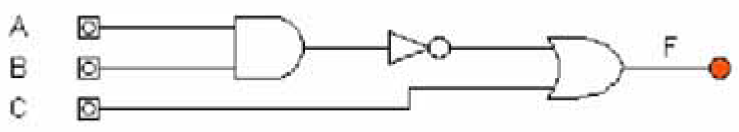


Table 12

A	B	A•B	$\overline{A \bullet B}$	C	F
0	0			0	
0	0			1	
0	1			0	
0	1			1	
1	0			0	
1	0			1	
1	1			0	
1	1			1	

The truth table for the circuit is given in Table 12. The output F is HIGH when the inputs A and B are both LOW OR if C is HIGH.

Hence the Boolean expression for the circuit is:

$$F = \overline{A \bullet B} + C$$

This could also be written:

$$F = (A \bullet B)^* + C$$

The NAND logical function

The AND gate and the NOT gate in the previous example may be replaced by a single NAND gate, which performs the same logical function. That is, it performs the inverse of the AND function, so its output is LOW only when all its inputs are HIGH. The truth table for a 2- input NAND gate is given in Table 13 and that for a 3-input NAND gate in Table 14.

Table 13

A	B	A•B	$\overline{\text{A•B}}$
0	0		
0	1		
1	0		
1	1		

Table 14

A	B	C	A•B•C	$\overline{\text{A•B•C}}$
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

The NAND gate can be manufactured more cheaply than other logic gates, as it requires fewer transistors in its construction. It is also possible to construct all other gates by using only NAND gates. In the real world, therefore, many integrated circuits are constructed using only NAND gates.

For instance, a NOT gate may be constructed by using a 2-input NAND gate with the 2 inputs connected together. Try this with Logic Gate Simulator, and show that it is correct.

The NOR logical function

The output of a NOR gate is the same as that of an OR gate whose output has been inverted. The NOR logical function can be produced by an OR gate followed by a NOT gate, but more usually a NOR gate would be used. The truth tables for 2-input and 3-input NOR gates are given in Tables 15 and 16 respectively

Table 15

A	B	A+B	$\overline{\text{A+B}}$
0	0		
0	1		
1	0		
1	1		

Table 16

A	B	C	A+B+C	$\overline{\text{A+B+C}}$
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

The Exclusive-OR logical function

The exclusive-OR (or XOR, or occasionally EOR) gate has only two inputs. Its output is HIGH when one or other of its inputs is HIGH, but not when they are both HIGH or both LOW. It performs the logical function:

$$F = A \bullet B + A \bullet B = A \oplus B$$

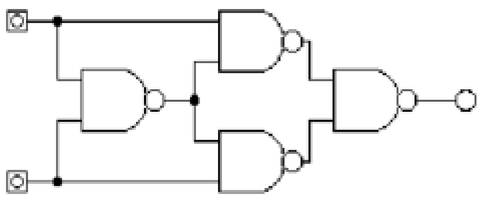
Its truth table is shown in **Table 17**.

Table 17

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

The XOR gate can be made by combining various other types of gate, and one such combination of gates that gives the XOR logical behaviour is shown in Figure 9.

Fig 9



The XOR is frequently used to test if two inputs are different.

Using Logic Gate Simulator, you will construct and test two different logic circuits listed below. The purpose is to demonstrate that both circuits perform the same **XOR** function. **XOR** requires two source operands and returns a single output as indicated by the truth table shown previously in Table 17.

You will use the **AND**, the **OR** and the **NOT** functions (called gates) to construct the circuit. Please note that the **NOT** gate is often also called an Inverter. These gates can represent by use of standard symbols listed below standard symbols, as shown in Figure 10. In all cases the symbols are drawn with the inputs from the left and the output towards the right.

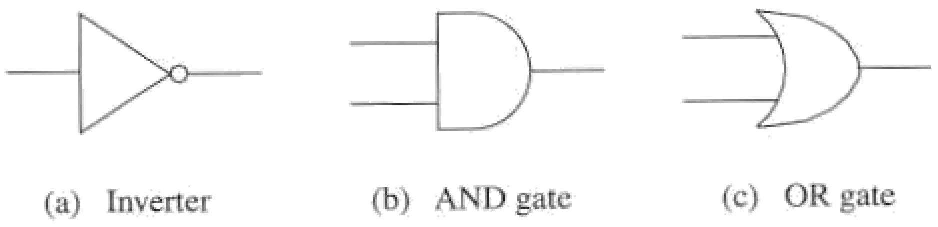


Figure 10: **AND**, **OR**, and **NOT** gates

The **XOR** function can be implemented with two Inverters, two **AND** gates, and an **OR** gate, by connecting them as shown in Figure 11.

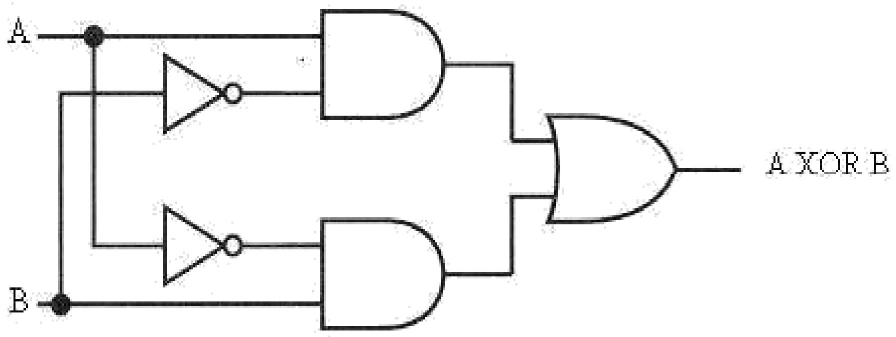


Figure 11 – **XOR** Circuit

Another way of implementing the **XOR** function is by connecting two inverters, two **OR** gates, and an **AND** gate, as shown in Figure 12.

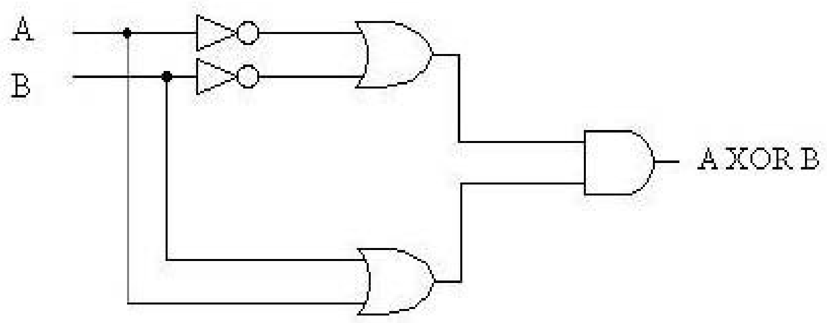


Figure 12 – **XOR** Circuit

You will demonstrate that both circuits perform the same function.

Start by constructing the circuit shown Figure 11. by planting two **AND** gates, an **OR** gate, and two inverters on the work area. Before continuing, save the circuit. The next step is to wire up the gates to create a circuit. Your circuit should now look like as shown in Figure 13.

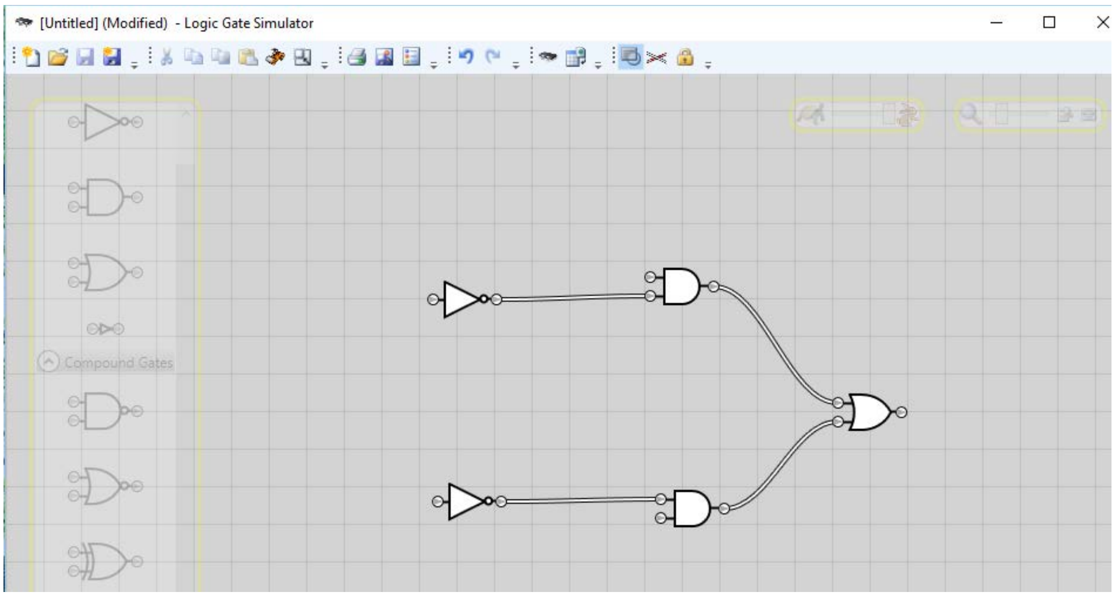


Figure 13: Wiring gates together

Logic Gate Simulator permits a wire to be connected only between two legal connections. In Figure 13 the inputs to the two inverters and the output of the **OR** gate aren't connected anywhere. In order to wire up the inputs and output we need points you can connect the wire to. In this case, you are going to use the interactive input device to provide an input signal from a push button and the **LED** to show the state of the output. You will also record the output of the two **AND** gates, so you will have to connect their outputs to **LED**s as well. You should add inputs and **LED**s to the circuit and complete the rest of the wiring, as shown in Figure 14.

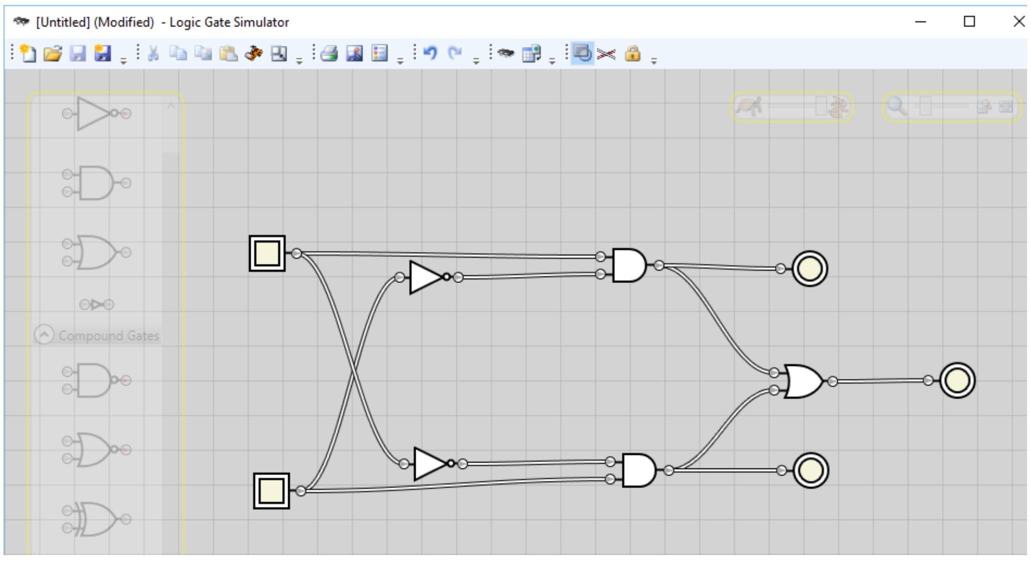


Figure 14: Completing the circuit

At this stage, you could test the circuit if we wanted. However, you will use the comment tool (indicated by the cloud symbol on the toolbar) to give the circuit a title. Click on the cloud and then click on the place at which you wish to add the text to open the text window. This brings in a text box. Enter the text "**XOR circuit**" and click **Ok** to place it on the screen. We also wish to label the circuit's inputs and outputs. To give a name to an input or output, first select it and then right click to bring down a menu. You enter the name into the text box and click **Ok**. Figure 15 shows the circuit where the inputs and outputs have been labelled with names.

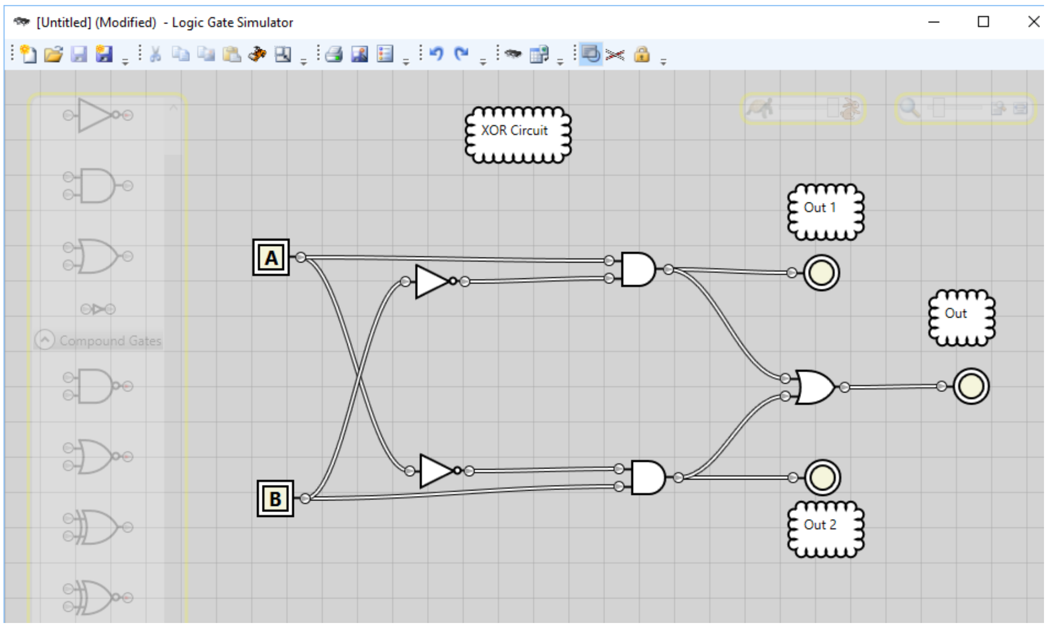


Figure 15: Labelling the circuit and inputs and outputs

You can similarly construct the circuit shown in Figure 12. This circuit requires two inverters, two **OR** gates, and one **AND** gate. Save this circuit in a file named LastnameF_L02. Since you will not record the outputs of the two **OR** gates you do not need to connect them to **LED**s. You still need to add the two inputs, and connect the output of the **AND** gate to an **LED**. Do not forget all the proper labelling just as you did in the previous diagram.

Complete the following table for each of the circuits. Validate that they produced the same results.

Inputs		Outputs		
A	B	Out	Out 2	Out 3
0	0			
0	1			
1	0			
1	1			

The exclusive-NOR logical function

The coincidence, or XNOR gate, has two inputs and one output. It produces a logical 1 only when both its inputs are the same, either HIGH or LOW. If the inputs are different, it produces a logical 0. The Boolean equation that describes an XNOR gate is therefore:

$$F = (A \bullet B) + (\overline{A} \bullet \overline{B})$$

This is the complement of the operation of the XOR gate.

Gates with inverted inputs

All of the previously described logical functions can be performed by using different gates with the inputs inverted. For instance, if the inputs to an AND gate are inverted, it will perform the function of a NOR gate. Similarly, if the inputs to a NAND gate are inverted, it performs the function of an OR gate.