Digital Logic
Logic Gate Primer

Authored by xor

evilzone.org

DIGITAL LOGIC STATES

In digital logic, there are only two recognized states for a signal.

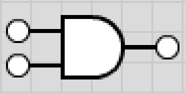
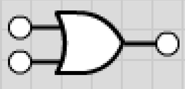
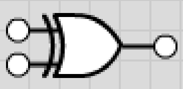
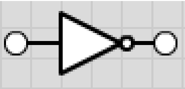
1 = HIGH / ON
0 = LOW / OFF

BASIC LOGIC GATES

A logic gate has two or more INPUTS and one or more OUTPUTS. INPUTS are generally shown on the left hand side of a gate, and the OUTPUTS on the right.

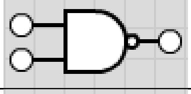
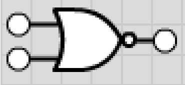
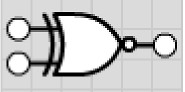
A logic gate defines a set of rules that apply to the inputs to affect the outputs, this can be represented visually and with Truth Tables.

Common Logic Gates are shown in the table below with their associated symbol and purpose.

Name	Symbol	Purpose
AND		If both inputs are HIGH, output is HIGH. Any other input, output is LOW.
OR		If one or more inputs are HIGH, output is HIGH. If both inputs LOW, output is LOW.
XOR		If one input or the other is HIGH, output is HIGH. If both inputs are HIGH, or LOW, output is LOW.
NOT		Inverts a gate OUTPUT, or INPUT signal. HIGH becomes LOW LOW becomes HIGH

COMBINED LOGIC GATES

Combining NOT with the other gates will invert their OUTPUT. You will notice the only difference between the symbols of the inverted gate is the circle just before the OUTPUT. These gates are shown below.

Name	Symbol	Purpose
NAND		If both inputs are HIGH, output is LOW. Any other input, output is HIGH.
NOR		If one or more inputs are HIGH, output is LOW. If both inputs are LOW, output is HIGH.
XNOR		If one input or the other is HIGH, output is LOW. If both inputs are HIGH, or LOW, output is HIGH.

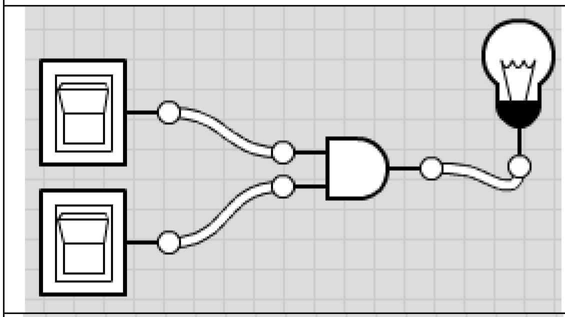
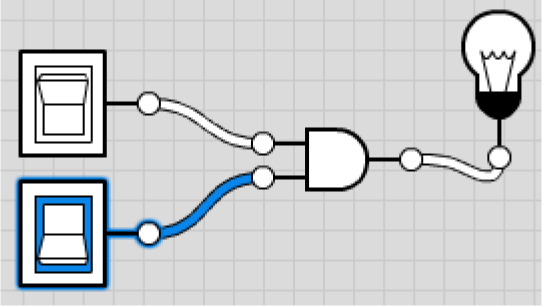
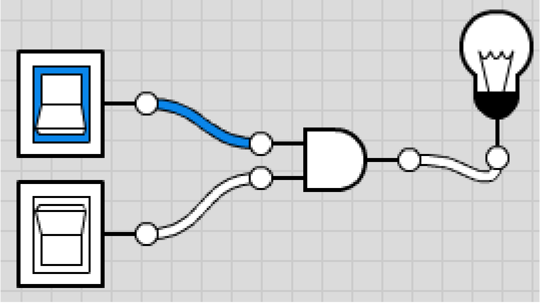
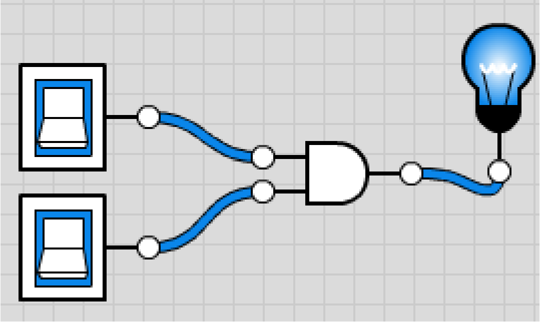
BASIC LOGIC GATES CONTINUED – WITH TRUTH TABLES

When designing digital logic, truth tables are often used to show all the possible combinations of inputs, and the expected output for each input combination. Examples of these tables are shown below, with a simple demonstration of each state using switches as INPUTS and light bulbs as OUTPUTS.

AND GATE

If both inputs are HIGH, output is HIGH.
Any other input, output is LOW.

Input A	Input B	Output
0	0	0
0	1	0
1	0	0
1	1	1

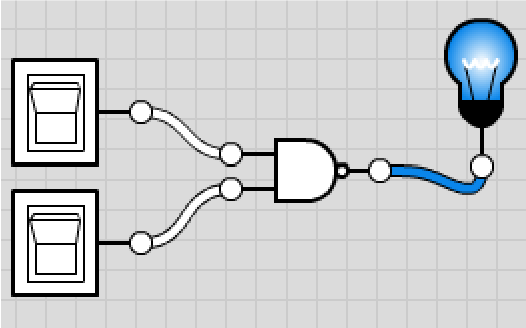
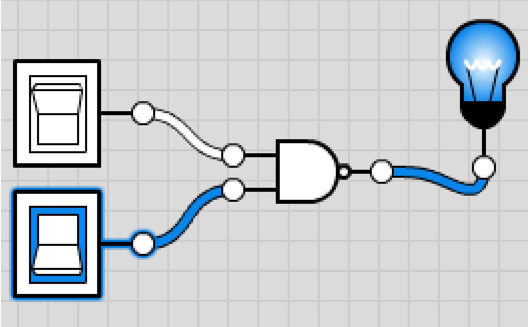
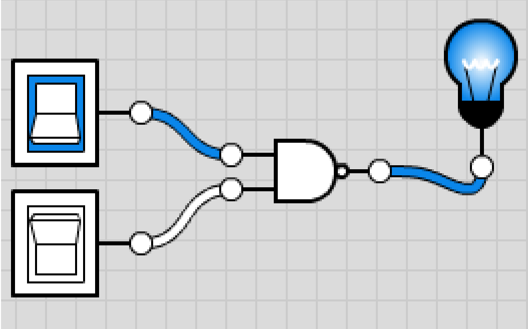
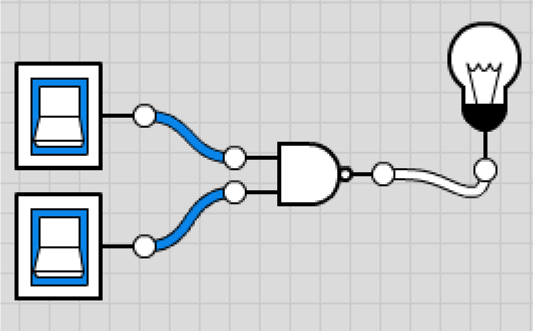
Schematic	Input A	Input B	Output
	0 LOW OFF	0 LOW OFF	0 LOW OFF
	0 LOW OFF	1 HIGH ON	0 LOW OFF
	1 HIGH ON	0 LOW OFF	0 LOW OFF
	1 HIGH ON	1 HIGH ON	1 HIGH ON

NAND GATE

By Adding a NOT gate we INVERT the signal of the AND GATE.

If both inputs are HIGH, output is LOW.
Any other input, output is HIGH.

Input A	Input B	Output
0	0	1
0	1	1
1	0	1
1	1	0

Schematic	Input A	Input B	Output
	0 LOW OFF	0 LOW OFF	1 HIGH ON
	0 LOW OFF	1 HIGH ON	1 HIGH ON
	1 HIGH ON	0 LOW OFF	1 HIGH ON
	1 HIGH ON	1 HIGH ON	0 LOW OFF

Further Learning – Self Testing

To play around with the other gates yourselves, head on over to http://logic.ly/demo/ and see if you can create the correct input output combinations for each of the other gates explained above.

OR

Input A	Input B	Output
0	0	
0	1	
1	0	
1	1	

NOR

Input A	Input B	Output
0	0	
0	1	
1	0	
1	1	

XOR

Input A	Input B	Output
0	0	
0	1	
1	0	
1	1	

XNOR

Input A	Input B	Output
0	0	
0	1	
1	0	
1	1	

Summary Notes

This is a precursory lesson to more advanced tutorials showing how digital logic gates are used within computers.

The next lesson will demonstrate how some of these gates can be combined to create more useful logic circuits such as a 1-Bit Adder.

If you have any questions, or would like to post me your results of the further learning page, please feel free to PM me your results on the forums.

Happy learning!

-- xor