Logic Gates

Abyan Majid

July 8, 2023

1 Logic gates

Logic gates are used to perform logical operations on boolean inputs (ie. 0s and 1s, where 0 represents "False" and 1 represents "True") to produce boolean outputs. Such use make logic gates a fundamental building block for designing digital circuits.

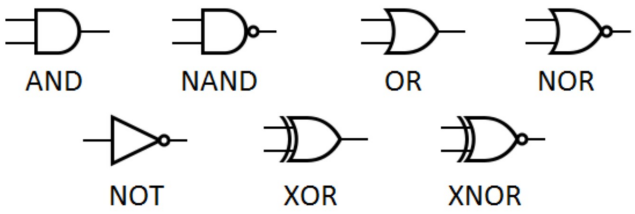


Figure 1: Symbols of logic gates

There are 7 elementary logic gates as shown below.

AND (Conjunction)	**OR** (Disjunction)	**NOT** (Negation)
$a \wedge b$	$a \vee b$	$\neg a$
Output is 1 if ALL inputs are 1	Output is 1 if ANY input is 1	Output is the OPPOSITE of input

NAND (Not AND)	**NOR** (Not OR)
$\neg(a \wedge b)$	$\neg(a \vee b)$
Output is 0 if ALL inputs are 1	Output is 0 if ANY input is 1

XOR (Exclusive OR)	**XNOR** (Not Eclusive OR)
$a \veebar b$	$\neg(a \veebar b)$
Output is 1 if the no. of inputs of value 1 is ODD	Output is 1 if the no. of inputs of value 1 is EVEN

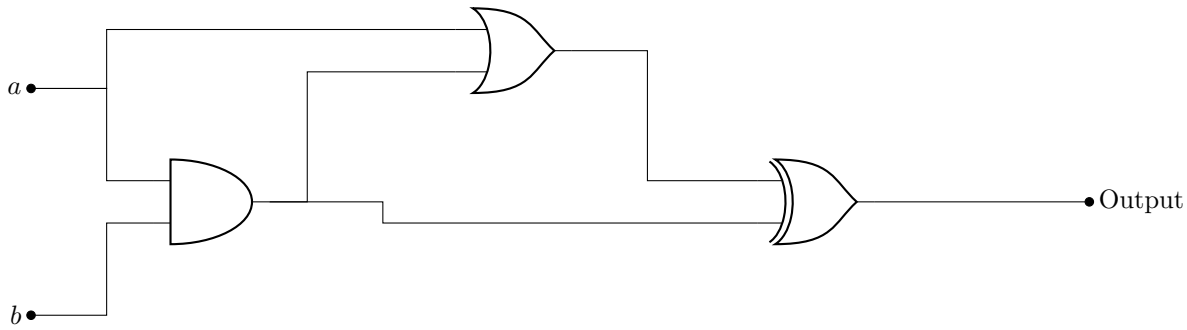
1.1 Alternate logical notation

The following is an alternate logical notation, if you don't feel like using $\wedge, \vee, \neg$, and $\veebar$ (honestly, i discourage using the alternatives below because they can be confused for arithmetic operators):

- AND: $a * b$
- OR: $a + b$
- NOT: a'
- NAND: $(a * b)'$
- NOR: $(a + b)'$
- XOR: $a \oplus b$
- XNOR: $(a \oplus b)'$

2 Digital circuits with logic gates

You can use a combination of logic gates to create a digital circuit, like the example below.



Suppose that we have inputs $a = 1$ and $b = 0$. We can follow through the logic gates to find the output of the digital circuit.

1. AND gate: Since $b = 0$, the AND gate outputs 0.
2. OR gate: The OR gate here takes the output of the AND gate as one of the inputs alongside a. Since $a = 1$, the OR gate outputs 1.
3. XOR Gate: The XOR gate here takes the outputs of the AND gate and the OR gate. Since the output of the AND gate is 0, and the output of the OR gate is 1, we have an odd number of inputs valued 1. Therefore, the XOR gate outputs 1.

So when $a = 1$ and $b = 0$, our digital circuit above will output 1!

2.1 Deriving truth table from a digital circuit

I hope you realize that if we try every possible combination of input values, we will end up with a truth table for our digital circuit. For example, here's a truth table derivable from the digital circuit above.

a	b	**Result**
0	0	0
0	1	1
1	0	1
1	1	0

2.2 More compact boolean expressions make for more compact circuits!

Recall that for every boolean expression E_1, there is two other equivalent expressions E_2 (sum of products) and E_3 (product of sums) that all result in the same truth table. In any case, **YOU SHOULD ALWAYS DESIGN DIGITAL CIRCUITS USING THE MOST COMPACT BOOLEAN EXPRESSION!** And to do that, you avoid basing your circuit on a "sum of products" or "product of sums" expressions! Instead, you design your circuit based on E_1, the most compact equivalent expression from which the "sum of products" and "product of sums" expressions are derived!