UW Math Circle
October 3rd, 2019

Logic Circuits

Logic circuits are built from *gates* connected by *wires*. Each gate has one or more input and output *terminals*, and represents a function from the input terminals to the output terminals. Inputs and outputs are each either 0 (off, or false) or 1 (on, or true).

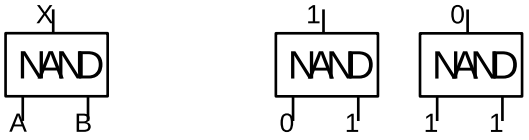
For example, the 2-input, 1-output NAND gate is a function from inputs A and B to output X, and outputs 0 if both inputs are 1, 1 otherwise. NAND is an abbreviation for *not and*.

Since NAND is a function, it must specify whether the output X is 0 or 1 for every one of the four possible combinations of inputs A and B.

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

We will call such a table of output values the function's *logic table*, and each row an *entry*.

A circuit consisting of a single NAND gate is drawn below on the left, with names A, B next to the input terminals and X next to the output terminal: convention is that inputs are at the bottom and the circuit flows upwards with outputs at the top.



The copies on the right of the diagram show how particular input values produce particular output values, as determined by the second and fourth entries in the logic table.

The NAND gate may seem arbitrary, but we will see that it is extremely powerful when combined into circuits. Many circuits in modern electronic computers are built entirely from NANDs.

Constructing new gates

To construct larger circuits, the input of a gate may be wired to another gate's output. The *negation* function (NEG) of one input inverts it, switching a 0 to 1 and a 1 to a 0:

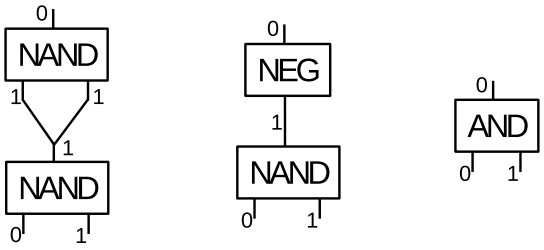
A	X
0	1
1	0

This logic table can be implemented by a circuit of a single NAND gate where both the gate's inputs are connected to a common wire, so that the NAND gate receives either 00 or 11 as input. Having built this circuit, we can abbreviate it as a new kind of gate, NEG:

Now we have two kinds of gates! Another useful function is *AND*, which is 1 only if *all* inputs are 1:

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A circuit for AND can be built by inverting the output of a NAND gate:

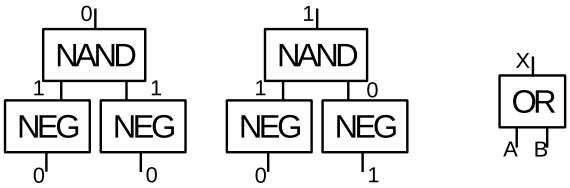


On the left of the figure above is a circuit for AND built from two NAND gates. In the middle the circuit is simplified by using the NEG gate. On the right the circuit is abbreviated as a new gate, AND. Each diagram shows the inputs 0 and 1 producing a 0 output, as per the second entry in the logic table.

The final basic logical function is *OR*, which is 1 if *any* input is 1:

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

A bit of experimentation shows that OR can be implemented using a circuit that inverts both inputs to a [NAND]:

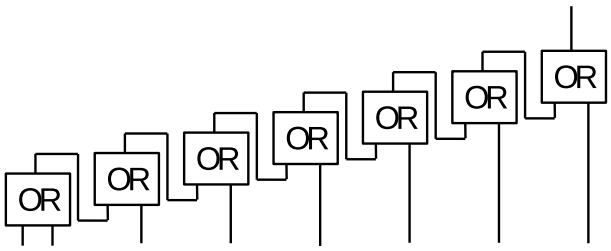


The leftmost two figures show two different inputs, 00 and 01, flowing through the circuit. On the right the circuit is abbreviated as a new gate, [OR].

Multi-input gates and circuit depth

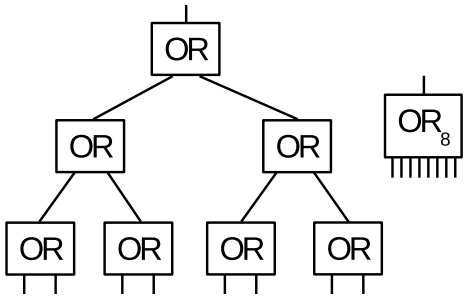
The functions AND and OR can be defined for any number of inputs: AND is on if *all* its inputs are on, and OR is on if *any* of inputs are on.

Here is one way to build [OR_8], an 8-input OR circuit, from seven 2-input [OR] gates:



The leftmost [OR] combines the first two inputs, and then the next [OR] combines the previous output with the next input.

Another way to implement [OR_8] is using a *tree* structure:



Again seven 2-input OR gates are used, but now the longest chain of OR gates between input and output is 3, not 7. This is called the circuit *depth*. Minimizing circuit depth is important, since in real computers each gate takes time to compute its outputs from its inputs.

If each OR is implemented using NAND gates, the depth of the circuit is doubled, because each OR is implemented as a sequence of two NAND gates.

The same structures work for building multi-input AND from 2-input AND. *To think about: What is special about these functions that enables them to be built from 2-input versions? Does this property have a name?*

Universal circuits

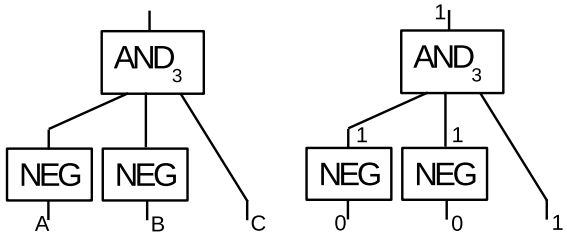
Is there any limit to what can be built using NAND? So far we've seen how to implement NEG, AND and OR.

Using circuits built from NEG, AND and OR, any function can be computed! We say therefore that NEG, AND and OR form a *universal family* of gates. Since these can all be built from NAND, NAND on its own is a universal family.

How do we know any function can be computed using NEG, AND and OR? The idea is to start from the logic table mapping inputs to outputs. Consider the following 3-input, 1-output function:

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

For each entry in the logic table that has a 1 output, we'll construct a sub-circuit that outputs 1 only if the inputs exactly match that entry. Then we'll OR together these sub-circuits using a multi-input OR. In our example function, the second, fifth, seventh and eighth entries have 1 outputs, so the final OR will have 4 inputs. Each sub-circuit will be an AND that verifies each of the three inputs matches its entry. For example, the sub-circuit for entry two must check that A is 0, B is 0 and C is 1, implemented as AND_3(NEG(A), NEG(B), C):



The figure on the right shows how the output is computed when the input matches the entry.

This process works for any number of inputs, but the number of entries grows rapidly: if there are n circuit inputs, then there are 2^n entries, and each entry that has a 1 output takes up to n NEG gates plus an AND_n circuit to implement. Therefore functions of many inputs can not practically be implemented this way, and computer engineers must rely on the special properties of the function to come up with cleverer circuits.

Because NAND is universal, any circuit family that can implement NAND is also universal, such as NEG and AND. This raises the question: what circuit families are *not* universal? For example, do AND and OR constitute a universal family - that is, can use just AND and OR to construct a circuit for every function?

Multi-output circuits

For functions with multiple outputs, one can construct a separate circuit for each output, but sometimes it helps to share sub-circuits, by connecting several wires to a single sub-circuit output.