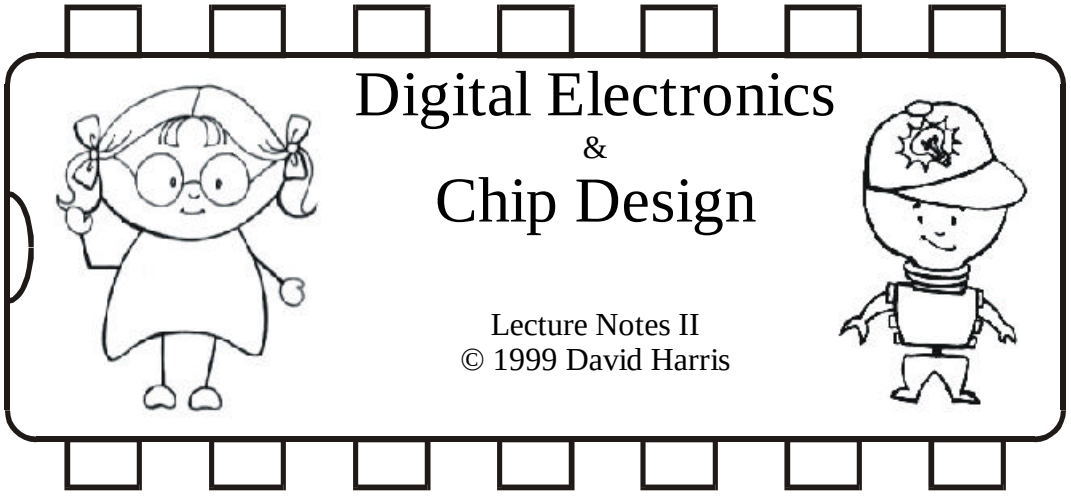


In conventional analog electronics, voltages are continuous; that is, they can be any value at all. In theory, this gives the analog designer tremendous power. Digital electronics, on the other hand, only uses two values of voltage, typically 0 and 5 volts.[1] At first glance, this limitation, called the "Digital Abstraction," would seem to be less powerful. However, digital electronics designers enjoy many advantages that are the envy of analog designers. Design becomes far easier: instead of having to solve differential equations to predict circuit behavior, digital designers can simply look at patterns of 1's and 0's (called Boolean values). With these simpler building blocks to work from, one can construct and understand far more complex circuits. Since the transistor, the building block of digital circuits, is so small and cheap today and readily available, one can construct these complex circuits very cheaply and efficiently. Thus, by limiting our building blocks to very simple 1's and 0's, we actually increase the range of circuits that we are smart enough to build and analyze. This is a powerful principle that appears in many branches of engineering.

Elementary Operations

Let us begin by defining a few terms about Boolean values. A Boolean variable is a variable that is either 1 or 0. We typically interpret this variable as 1 indicating TRUE and 0 indicating FALSE. Or 1 could indicate ON and 0 could indicate OFF. In the setting of circuits for this class, 1 indicates 5 volts and 0 indicates 0 volts.

A Boolean value by itself is boring. Let's do some math with these values. The simplest operation is NOT. NOT turns TRUE into FALSE and FALSE into TRUE. Given a Boolean variable A, we write **NOT A** as $\overline{A}$ and sometimes refer to it as the complement of A. We can write a "Truth Table" to represent this operation:

A	$\overline{A}$
0	1

[1]These are arbitrary values and state of the art chips are beginning to use 0 and 3.3 or 0 and 3 volts, but for all of the work done in this class, we will be using TTL (Transistor-Transistor Logic) gates that operate on 0 and 5 volts.

1	0

The next operation we will consider is OR. OR takes two Boolean values and returns TRUE if either one of them is TRUE. Logicians often write **A OR B** as **A** Ç **B**; in engineering, we more frequently write it as **A + B**. (But remember that this is not the + operation you are accustomed to; 1 + 1 = 1 OR 1 = 1.) Below is the truth table for OR:

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

AND is the third basic Boolean operation. AND takes two Boolean values and returns TRUE only if both are TRUE. Logicians write **A AND B** as **A** È **B**; engineers write **A • B** (or, when they are really lazy, simply **AB**). Below is a truth table for AND:

A	B	A•B
0	0	0
0	1	0
1	0	0
1	1	1

There are a number of other common Boolean operations: NOR, NAND, and XOR. **A NOR B** is defined to be **NOT (A OR B)** and is written $\overline{A+B}$. **A NAND B** is defined to be **NOT (A AND B)** and is written $\overline{A \cdot B}$. XOR (also seen as EOR on occasion) means Exclusive-OR; **A XOR B** is written **A** Å **B** and is TRUE if either A or B but not both are TRUE. Truth tables for these operations appear below

A	B	$\overline{A+B}$	$\overline{A \cdot B}$	A Å B
0	0	1	1	0
0	1	0	1	1
1	0	0	1	1
1	1	0	0	0

From these elementary operations, we can construct more complex Boolean functions. For instance, consider the function **F(A,B,C) = A•B + A•C + B•C**.[2] This function is TRUE if any two of it's three arguments are TRUE. We can write a truth table for it:

A	B	C	A•B	A•C	B•C	F(A,B,C)
0	0	0	0	0	0	0
0	0	1	0	0	0	0

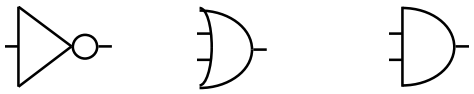
[2]Note that AND is of higher precedence than OR in Boolean expressions; i.e. **A + B • C** means **A + (B • C)**, not **(A + B) • C**.

0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	0	0	0
1	0	1	0	1	0	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

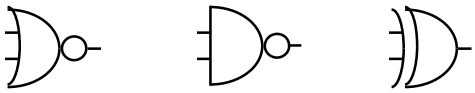
Let us take an aside on conventions of writing truth tables. It is conventional to write your inputs in the order shown here: the rightmost column alternates 0,1; the next alternates 0,0, 1,1; the next 0,0,0,0, 1,1,1,1, and so forth. This is equivalent to counting in the binary system. Next, we list the various terms of the expression: A•B, A•C, and B•C. While we don't care all that much what values they have, it is convenient to write them down because we need these terms to compute the final expression. It's a good idea to write terms when functions get very complex; for simpler functions like this, it is easy enough to read off the answer in your head that you may not bother with the terms after a bit of practice.

Logic Gates

This theory of Boolean algebra is amusing, but how can we build actual practical digital circuits? Fortunately for us, these various Boolean operations map directly to circuit elements that we have at our disposal as digital designers. These elements, called logic gates, are built into chips that you have in your lab kit. For instance, the 7404 chip has 6 NOT gates on it. First, let us look at how we draw circuit schematics that represent the various Boolean operations we have discussed:



NOT OR AND

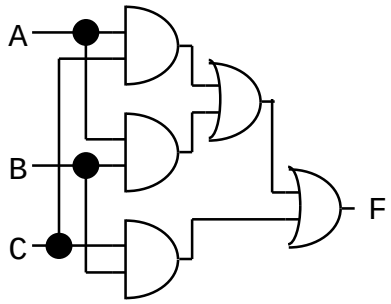


NOR NAND XOR

Note how the circle on a terminal denotes inversion. In addition, we occasionally encounter multiple-input versions of these gates. For instance, the three input AND gate (which returns TRUE if and only if all of its inputs are TRUE) is shown below:

AND3

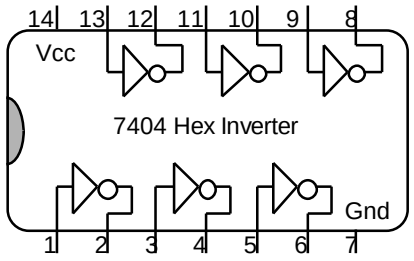
More complex functions may be constructed by connecting gates in sequence. For example, consider the function **F(A,B,C) = A•B + A•C + B•C** that we examined earlier. We can build it from five gates, as shown below:



Constructing Physical Circuits

In the economies of chip fabrication, it costs virtually the same amount of money to construct chips with a single transistor as it does to construct chips with hundreds or even thousands of transistors. Therefore, when you buy logic gates, they generally come on a chip containing several gates.

The TTL Logic Databook will become your close friend over the duration of this course. It contains diagrams of hundreds of types of chips, showing how to connect inputs and outputs to realize various logic functions. Let us examine the 7404 "Hex Inverting Gates."



The 7404 chip contains 6 NOT gates in a 14 pin DIP (Dual Inline Package) chip. At one end of the chip, you notice a notch and a dot beside a pin. Pin 1 is always immediately to the left of the notch, looking at the chip from above. Pins are numbered from there in counterclockwise order. The chip contains connections to power and ground to supply energy for operation and six inputs and outputs to NOT gates. It is conventional to place a chip in a breadboard such that the notch is at the top (if vertical) or left (if horizontal). By consistent placement, one makes fewer mistakes identifying pin numbers and connecting wires.

The first rule to always remember when building circuits is to always connect the power and ground (Vcc[3] and Gnd) pins of a chip. Vcc connects to 5 volts; Gnd connects to 0

[3]Vcc stands for the collector voltage on a bipolar transistor. While we are not concerned with the actual transistors, the voltage terminology crosses over. Occasionally, you will also hear the term Vdd, the drain

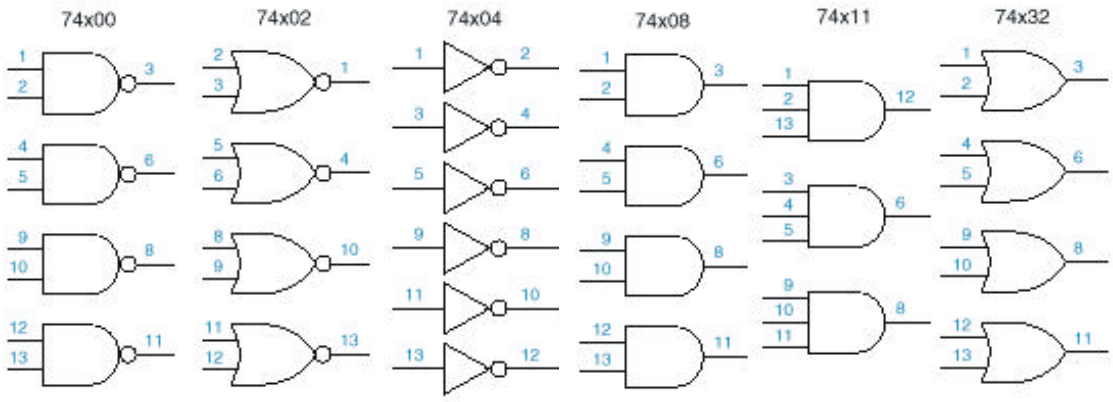
volts (ground). This is perhaps the single most common mistake that beginners have in their designs. On TTL logic gates and most other chips, Vcc is almost always the pin to the right of the notch and Gnd is the pin diagonally opposite. The salient exception to this rule is the 7483 adder chip, a very old design with power and ground in the middle instead of at the corners.

Now, to place a not gate in your circuit, choose one of the six gates. Let's choose, for no particular reason, the bottom right inverter. Hook the input to pin 5. The output can be measured at pin 6. That's all there is to it.

The 7400 series of digital logic chips have almost every type of gate you might wish to use in addition to a large set of complex logic circuits such as counters, adders, and shift registers. These various devices are described in your data book. In particular, you will probably be interested in the following chips, which are provided in your lab kit and which you may wish to use to solve the various problems:

Chip	**Gate**	**Gates/Chip**
7400	NAND	4
7402	NOR	4
7404	NOT	6
7408	AND	4
7411	AND3	3
7432	OR	4

Pinouts of these chips are shown below. Remember that the top right pin is always Vcc and the bottom left pin is always Gnd.



voltage on a MOSFET (Metal-Oxide-Semiconductor Field Effect Transistor). In either case, simply think 5 volts.