

Physics 201 Electronics Lab 7:

Digital Logic

1. Prelude

Up to now, all of your work in the electronics lab has involved *analog* circuits. Such circuits use a continuous range of voltages. If you have an analog circuit with an input voltage of 2.0 V, say, and you change the input to 2.1 V, the circuit will act differently. Even if you change it to 2.001 V, the circuit will act a little differently (or maybe a lot differently, depending on what circuit it is.)

In contrast, *digital* circuits use only a discrete set of voltage levels (example: 0.0 V, 0.1 V, 0.2 V, etc., but no voltages in between these values). Digital logic circuits are an extreme case: they only use two voltage levels, typically 0 V and +5 V. Every signal you put into a digital logic circuit, and every signal you get out, will be at one of these two voltages.

Here are some names for the two levels:

- “Low” and “High”
- “Off” and “On”
- “False” and “True”
- “0V” and “+5V”
- “0” and “1”

We will use the names “0” and “1.”

Computers are made of digital logic circuits. Everything that is stored in a computer is stored as a series of 0’s and 1’s. To pick a random example, the letter “q” is usually stored in a computer as this series of digits: 01110001. Because they can only use the digits 0 and 1, computers must do all computations as binary arithmetic.

You will be working with several types of digital circuits today. You should think of each circuit as taking one or two input signals, each of which is 0 or 1; doing something with those signals; and producing an output, which is also 0 or 1.

2. Inverter

Let's start by looking at the input and output voltages for the simplest sort of logical circuit, an inverter. An inverter takes an input and gives you the opposite. Here are the symbol for an inverter and a "truth table" which describes its operation:



Get a 7404 "hex inverter" chip. It is in a 14-pin DIP package. Install it on your breadboard, with pin 1 in the upper left corner. Install all your chips today with this orientation. Pin 1 is indicated either by a notch in the end of the chip or a dot right next to the pin, in one corner of the chip. Beware that many chips also have a dot centered on one end of the chip; this dot does *not* indicate pin 1. It is easy to misjudge which pin is pin 1. When in doubt, ask.

A datasheet for this chip is at the end of this writeup¹. The pinout diagram on the datasheet shows how to power the chip. Connect pin 7 (labeled GND in the pinout diagram) to ground and connect pin 14 (labeled V_{CC}) to +5 V. Notice that the symbol for the inverter appears six times in the pinout diagram. The chip has six inverters, each operating independently of the others. That's why it is called a *hex* inverter chip.

Connect the input of one of the inverters to +0 V. You should think of this as setting the input to 0. Measure the output of the inverter with a voltmeter. It should be around +5 V, possibly somewhat lower. You should think of this output as 1. The circuit has "inverted" the 0 and given you 1.

Now connect the input to +5 V, i.e., an input of 1. The output should now be 0 V. Is it? Draw a (very simple) diagram of this circuit in your notebook and note the results you are measuring.

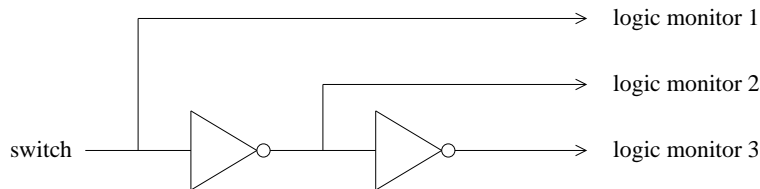
¹This circuit is called HD74LS04 on the datasheet. "HD" refers to the manufacturer. "LS" refers to the particular technology used for the components in the chip. Different technologies have different voltages, different power consumption, different speeds, different output impedances, different manufacturing costs, etc. However, the input and output connections and basic operation will be the same for all 7404 chips.

Your Cadet II board makes it easy to provide inputs to your circuits and to look at their outputs. In the lower left section of the Cadet II board there is a set of eight “logic switches. The logic switches control voltages in the little breadboard block immediately above the switches. Each switch either puts out +0 V (if set to 0) or +5 V (if set to 1) in one column of that breadboard. (The switch labeled “+5” and “+V” should be set to “+5.”)

To look at the output of a circuit, use the logic monitors on the upper right section of the Cadet II board. (The switches in this section should be set to “+5” and “TTL.”)

Connect one of the logic switches to one of the logic monitors via a long wire. Check that the logic monitor lights up correctly as you flip the switch back and forth. Leave the wire in place and run a second wire from that same switch to the input of the inverter, and another wire from the output of the inverter to a second logic monitor. Flip the switch back and forth. What do you see? Does it make sense?

This is still pretty basic stuff. Inverters are not too complicated. Let’s do one more circuit using them. Leave all the wires that are already in place. Add a wire from the output of your inverter to the input of another inverter on the same 7404 chip. Hook up the output of that inverter to another one of the logic monitors. A circuit diagram looks like this (you should draw every circuit in your notebook, using logic symbols):

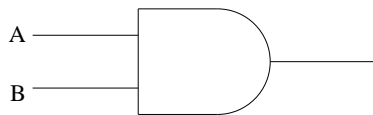


What do you see when you flip the switch back and forth? Does it make sense?

Leave the inverter chip plugged into your breadboard with its power connections intact. You’ll need to use it again. You can remove the wires to the inputs and outputs.

3. AND gate

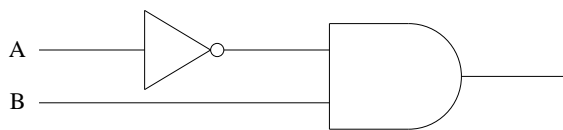
A device like an inverter is called a “gate.” The inverter is a gate with one input. Many gates have two inputs. Get a 7408 “quad 4-input AND gate.” It has four independently operating AND gates. Each gate has two inputs, A and B. If both inputs, A and B, are 1, then the output is 1. Otherwise the output is zero. Here are the symbol and truth table for an AND gate:



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

The pinout for the 7408 is given on its datasheet. Connect GND to ground and V_{CC} to +5V. You need to do this for every chip. Then connect two switches to two inputs of one of the AND gates and connect the output to a logic monitor. Try the different combinations of switch settings. Does the output look like you expect?

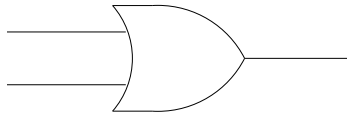
Gates can be combined with one another. Wire up an inverter and an AND gate in the following combination. Try the different switch settings and fill in the truth table (in your notebook). Do the output values make sense?



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

4. OR gate

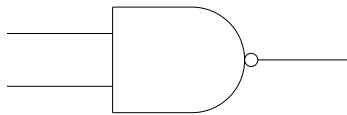
An OR gate takes two inputs. Its output is 1 if either A or B or both is 1; otherwise it is 0. The 7432 chip contains four OR circuits. Get a 7432 chip and verify that it behaves in the expected way.



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

5. NAND gate

Get a 7400 quad NAND gate. Wire it up and figure out what its truth table is. Why do you suppose it is called NAND? Here's the symbol:



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

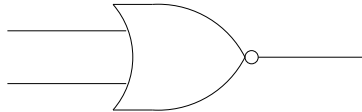
NAND has some nice properties. Compare the circuit diagram on the NAND datasheet to the circuit diagrams on the AND and OR datasheets. The NAND requires the fewest components to build. That is important if you are a circuit manufacturer trying to squeeze as many gates as possible onto a single chip. (Computer chips have millions of gates.)

Another nice thing about NAND gates is that you can build any digital circuit out of NAND gates. Figure out how to build each of the following out of NAND gates. In each case, wire up the circuit that you have designed and verify that it works.

1. An inverter. This can be done with a single NAND gate. How do you wire it up?
2. An AND gate. You can combine two NAND gates to a circuit equivalent to an AND gate. How? (Hint: think about your previous circuit.)
3. An OR gate. This is trickier. You can combine three NAND gates to make a circuit equivalent to an OR gate. How?

6. NOR gate

For completeness, we mention the 7402 quad NOR gate. It complements the OR gate in the same way that NAND complements the NOR. Here's the symbol and truth table. You don't need to wire this one up.

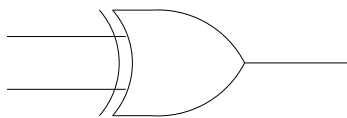


Input A	Input B	Output
0	0	1
0	1	0
1	0	0
1	1	0

7. XOR

7.1. Making an XOR

Exclusive OR (XOR) is a two-input logic operation whose output is 1 if either A or B is 1, but not if both are 1. The symbol and truth table look like this:



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

How can you construct an exclusive OR out of other gates (not necessarily NAND gates)? There are various ways to do it using three gates. This is a puzzle—it may take a while for you to figure it out. Once you have a design, wire it up and verify that it works.

7.2. What its good for

One of the things that exclusive OR is good for is binary addition. Let's pause and talk about binary numbers for a minute. All numbers in digital logic are binary numbers. How do they work? Recall that, in base ten, a number like "8325" means (reading the digits right to left) 5 ones plus 2 tens plus 3 hundreds plus 8 thousands. The values of the columns (ones, tens, hundreds, thousands) are powers of ten. Each digit is a number from 0 to 9.

$$\underline{8325} = \underline{8} \times 10^3 + \underline{3} \times 10^2 + \underline{2} \times 10^1 + \underline{5} \times 10^0$$

In binary, there are only two digits, 0 and 1. The columns are powers of two (ones, twos, fours, eights, sixteens, etc.), so that "1101" would be, reading right to left, 1 one, 0 twos, 1 four, and 1 eight, for a total of 13. Thus 1101 in binary is 13 in decimal.

$$\underline{1101} = \underline{1} \times 2^3 + \underline{1} \times 2^2 + \underline{0} \times 2^1 + \underline{1} \times 2^0$$

Here are the first few binary numbers:

decimal	binary	decimal	binary	decimal	binary	decimal	binary
0	00000	8	01000	16	10000	24	11000
1	00001	9	01001	17	10001	25	11001
2	00010	10	01010	18	10010	26	11010
3	00011	11	01011	19	10011	27	11011
4	00100	12	01100	20	10100	28	11100
5	00101	13	01101	21	10101	29	11101
6	00110	14	01110	22	10110	30	11110
7	00111	15	01111	23	10111	31	11111

Binary addition works like decimal addition. Line up the two numbers being added. Add the rightmost column first. If the sum is greater than one, you have to carry, just like in decimal arithmetic when the sum of a column is greater than nine. In binary, you carry 2, so subtract 2 from the sum of the column and carry it (as a 1) to the next column.

Here are examples of binary sums:

$$\begin{array}{r}
 001 \\
 +010 \\
 \hline
 011
 \end{array}
 \qquad
 \begin{array}{r}
 101 \\
 +001 \\
 \hline
 110
 \end{array}
 \qquad
 \begin{array}{r}
 101 \\
 +011 \\
 \hline
 1000
 \end{array}$$

The above three sums are the binary equivalents of $1+2=3$, $5+1=6$, and $5+3=8$.

One of the uses of an XOR gate is as the one's digit of a two-bit adder. A two-bit adder is a circuit which adds two one-digit binary numbers, each of which is 0 or 1. It has two outputs, which are the two digits of the sum of the numbers, expressed in binary. Here's a truth table for the two outputs:

Input A	Input B	A+B in Decimal	A+B in Binary	Output 2's Column	Output 1's Column
0	0	0	00	0	0
0	1	1	01	0	1
1	0	1	01	0	1
1	1	2	10	1	0

Notice that the output 1's column is the XOR of inputs A and B. What circuit would give you the output 2's column? If you build these two circuits and put the output displays for the two digits next to each other, you have built a calculator. It isn't a very fancy calculator—and it can only add 0 and 1—but you have to start somewhere.

8. *Build your own logic monitor*

(If you prefer, skip this section and jump to the next one.)

If your breadboard didn't have built in "logic monitors," you could build your own. How can you get a red LED to light up when a voltage is high and a green LED when a voltage

is low? Work out a design for the circuit, and then get the components you need and build one on your breadboard. (Hint: any resistor(s) you use should be $1\text{ k}\Omega$ or greater. Logic circuits aren't good at driving low impedances.)

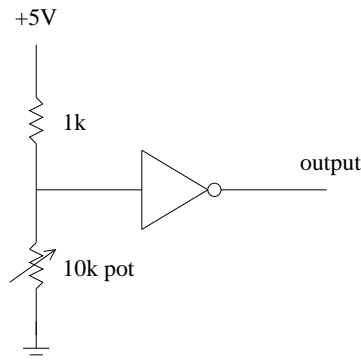
9. A challenge

Can you make a 3-bit adder, i.e., a circuit which takes three numbers and sums them in binary? You need to design a circuit both for the ones bit and the twos bit. You might find it useful to assume that you have a chip with XOR built in. In fact, we have a chip like that, the 7486.

10. Low and high

What range of voltages really represents 0 and what range represents 1 in the circuits you have been using? It depends on the chip. Look at one of the datasheets. They give the maximum voltages allowed for low level signals (0) and minimum voltages required for high level signals (1). The voltage ranges depend on the specific chip technology being used, and your chip may be different than those described in the datasheet.

Try measuring the voltage levels for your inverters by sending a range of voltages from 0 to 4.5 V into the input and seeing what output voltages result. To do this, hook up the 5 V supply to a $1\text{ k}\Omega$ resistor and a 10k pot. Use this as a variable power supply from 0V to around 4.5 V. (How does this work?) connect this into the input of the inverter (7404) and measure the output using a voltmeter.



What range of input voltages cause the output to be high (around 4 or 5 V)? What range cause it to be low?

11. Postlude

Today's experiments used the most basic logic chips. That's always a good place to start. The 7400 series has circuits which do somewhat more sophisticated things than just simple logic operations—for example, there are chips which add to binary numbers together, and which act as “memory cells” to store information and move it around. Much more complicated chips are available, up to and including the central processing units of computers. These are just scaled-up versions of what you have been using today. Finally, there are *field programmable gate arrays*. A single FPGA chip can have millions of gates, and the connections between those gates (on the chip) can be configured and re-configured by the user. No more wires!