Physics 335
Lab 1 – Intro to Digital Logic

We'll be introducing you to digital logic this quarter. Some things will be easier for you than analog, some things more difficult. Digital is an all together different animal than analog and you'll rarely be concerning yourself with exact voltage values this quarter. Rather you'll mostly be concerned if a signal is one of two values "Hi" or "Low" (Frequently "True" and "False").

We'll be using mostly 5 volt logic this quarter. Hi will be "close" to 5 volts, Low "close" to ground. (What we mean by "close" will vary slightly, depending on the logic family – More on this in a few.) Because we only care about a signal being high or low, a range of voltages are acceptable for each logic family.

Advice:

- Use the fixed 5 volt supplies for your power in digital circuits, rather than the variable outputs.
- Contrary to the advice given for analog circuits, set up the power busses along the sides of the breadboard, using one for +5 and one for 0 volts. Ground the 0 volt side. Put bypass caps, one 15μF tantalum cap and one 0.1μF Mylar or ceramic cap across the breadboard power supply strips.
- Connect all +5 circuit points with red wires and all ground points with black wires. This helps when you try to trouble shoot.
- Power supply connections in digital circuits are almost never shown in the circuit diagram. As a way to remember that they are necessary, always make the power connections *first*, before you wire up the signal paths.

1-1 Logic Probes

We'll be using logic probes for much of the quarter. They're a simple device, but useful and easy to understand.

You'll use them mostly to determine 3 states:

- Hi – Close to 5 Volts
- Low – Close to Ground
- Floating – Not connected to a voltage source

The third state is an important one—and an advantage of the logic probe at times over just using a scope probe.

Study the laminated sheet which describes the operation of the logic probe. Connect the power clips to the power supply, with +5V to red and 0V to black.

There are two switches on most of the logic probes: pulse/memory and TTL/CMOS. Set them to PULSE and CMOS respectively for the moment.

Note that at first, no indicator LED's are lit. Touching the probe to Ground (Low) lights up the low indicator LED – Similarly for the +5 V (Hi) side. Note that with the probe you can distinguish both HI and LO from NOT CONNECTED. This feature makes it especially useful.

The SYNC output from the function generator is also specially designed for digital logic. This output is a square wave of 5V amplitude. Look at the output on the scope, DC coupled, and confirm the amplitude and that LO is at 0 volts. Also, look at the SYNC output with the logic probe and confirm it behaves the way you would expect it to. Run the SYNC signal to a place on your breadboard, and touch the tip of the probe to this signal. Use the PULSE setting and try a very low frequency setting (< 0.3Hz). Watch how the probe responds in correspondence with the signal seen on the scope. Turn up the frequency (slowly, to $>$ 1kHz) and note what the probe does.

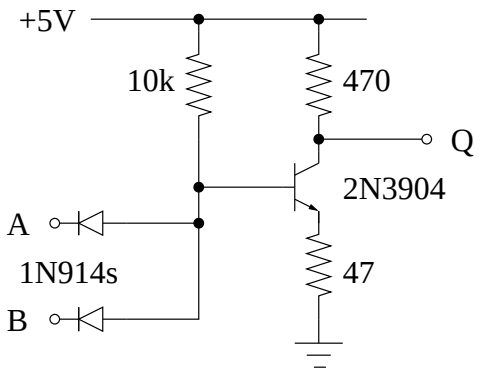
Try the MEM setting on the probe with very low frequency from the generator (≈ 0.1Hz) and compare what you see with the behavior you get with the PULSE setting.

You'll notice that in the PULSE position when you give the logic probe a HI signal, the "pulse" light will flash. This can be "captured" by switching the pulse/memory switch to "memory". This is useful for detecting a brief glitch on a logic line.

When using the probe, depending on the particular logic chip you are testing, select for TTL (all chips with an "LS" in their chip name) or CMOS (All chips with a C, HC, HCT or CD in their chip name).

1-2 A Simple Transistor Gate

Wire up the simplified discrete logic gate shown below.



Apply +5 or 0 volts to each of the inputs, and make a table showing the resulting voltages that all combinations of inputs produce at the output. From this table, make a truth table (using 1's and 0's, or H's and L's) and deduce what sort of a gate this is: AND, OR, NAND, NOR, or XOR.

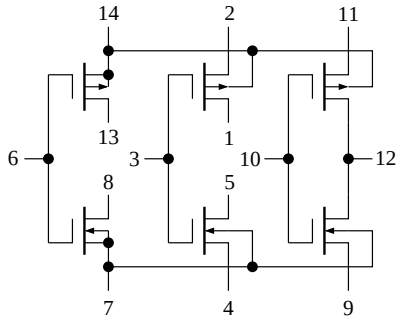
Describe in your report how the circuit works. That is, explain what the transistor does in order to generate the kind of output you see, how much current flows in the base and collector legs of the transistor, and what you expect the threshold voltage to be. The threshold voltage is the voltage applied at the input which is necessary to cause the gate to go from its low to its high state (or vice versa).

Try to measure the threshold voltage by applying a variable input voltage to the gate. You can use either a slow triangle wave or one of the variable outputs on the power supply. Keep the maximum input voltage $\leq$ 5V.

What happens if both inputs are left unconnected (open)?

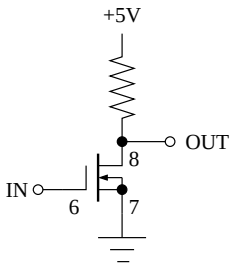
1-3 CMOS NOT Gate

We will use the CD4007 chip. The pinout is shown below



(a) Passive pullup

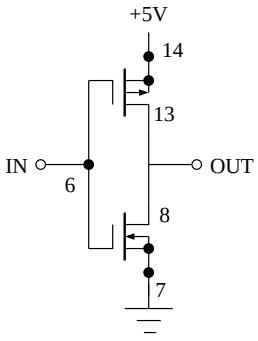
Build the circuit below using one of the N-channel MOSFETs in the CD4007 package. Be sure to tie the "body" pins appropriately: Pin 7 to ground, pin 14 to +5V. (In many cases, the body is tied internally, so you don't have to worry about it.)



Drive the inverter with the SYNC square wave. Look at the output on the oscilloscope and draw the waveforms that you see when the input is about **1kHz** and when you increase the frequency to **100 kHz**. Make an estimate of the capacitance at the output from the time constant $\tau = RC$. (Do this quickly by measuring the half-life of the RC decay.) Notice this is just an inverting amplifier. You could have made the same thing, of course, with a bipolar transistor. (You practically did, actually, back when you made a common emitter amplifier (which had some additional circuitry for transistor bias, etc. But things are remarkably similar.)

(b) Active pullup

Now replace the 10k resistor with one of the P-channel MOSFETs to build the following inverter:

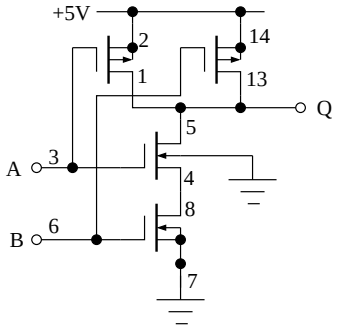


Look at the output as you did with the passive pullup version at low and high frequencies. Sketch the output waveforms.

The improved waveform at high frequencies is why active pullup is used in almost all logic families. The CMOS version is especially simple in conception, and the very low ON resistance of the MOSFET makes it possible to have the output stay close to the supply rails, either HI or LO. We'll compare with TTL a bit later.

1-4 CMOS NAND

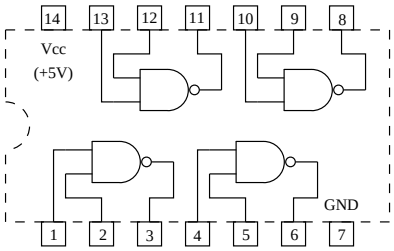
Wire up the circuit below, and confirm that it does indeed perform the NAND function (assuming HI = 1).



Make a truth table showing the values of the input voltages and the corresponding output voltages.

1-5 Input and Output Characteristics of CMOS vs. TTL

Now that you can "make" logic gates out of pieces, let's switch to more standard packages. The pinout below shows a quad 2 input NAND gate, the 74XX00: in TTL it is 74LS00, in CMOS it is 74HC00.



Different logic families have different logic threshold levels. Let's briefly look at them.

Consider two 74XX00 chips: 74HC00 and 74LS00. Both perform the same logic function (NAND) indicated by the 00 in their part number. They are not, however, completely interchangeable.

Plug both chips into the breadboard, and wire up the power connections. Then, for the CMOS part (74HC00) connect all *UNUSED* inputs to ground or +5V. It does not matter which, but the inputs on CMOS cannot be left floating (we'll see why in a moment). Do NOT connect the OUTPUTS of the gates to anything. For the TTL part, you can leave the unused inputs alone (although it is a good idea to tie them high in a circuit you make for real.)

(a) Outputs

Make a table like the one below, and make measurements to fill it in. Only one "logic out" column is needed, because both CMOS and TTL should agree in terms of HI and LO.

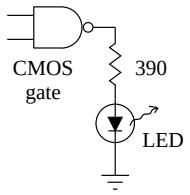
INPUT		OUTPUT		
		Logic	Voltages	
Logic in		(0 or 1)	TTL	CMOS
0	0			
0	1			
1	0			
1	1			

You can just plug the wires into ground or +5 volts for the input. Use the logic probe for logic levels and the voltmeter for voltages.

(b) Floating Inputs

TTL – *Disconnect both inputs* (i.e., let them float) to a TTL NAND and note the output *logic level*. What does the TTL gate apparently see on its inputs when they are not connected, a HI or a LO?

CMOS– Wire up an LED and 390Ω resistor to one of the NAND outputs on the CMOS version. This will allow you to see whether the output is HI or LO while your hands are doing other things, as they will be shortly.



Now tie one input on that same NAND to +5V, and for the other input, tie it to about 6 inches or so of wire, but leave the other end *unconnected* and dangling in air. Now watch the LED as wave your hand near the wire. Weird, huh?

Try touching one hand to +5 V on the breadboard and to ground as you wave your other hand near the loose wire. You should be convinced by this that floating CMOS inputs are to be avoided, if you want sensibly behaving logic!

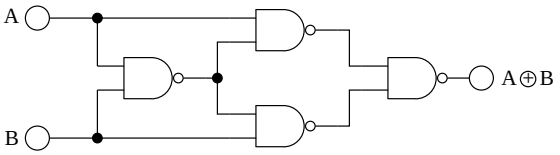
NOTE: Uncertain inputs on CMOS chips also cause them to consume considerably more power than they should. Also, if you leave the power terminals on CMOS chips unconnected (by accident!), you can get some very strange behavior.

1-6 Making What You Have Into What You Want

Clever use of logic will allow you to make one function out of another. Enter DeMorgan's Theorem.

Use NAND gates (either CMOS or TTL) to make the following functions, under the assumption that HI = Logic 1. Sketch the circuits that you build.

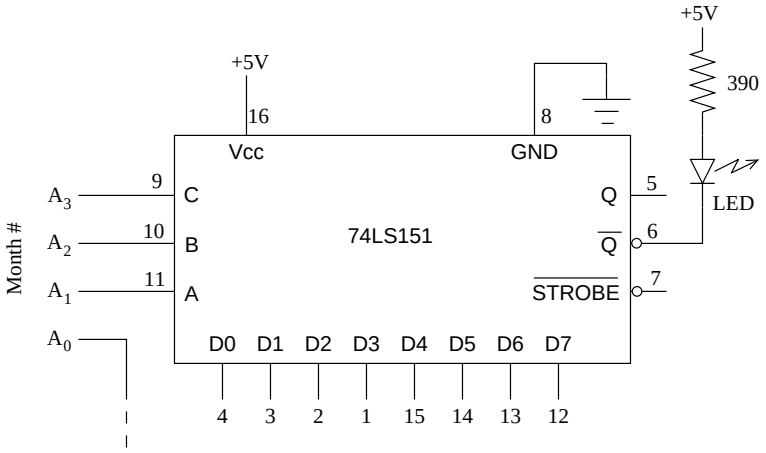
- Make an AND gate.
- Make an OR gate.
- Make an XOR gate. Note one version below. You may come up with your own.



1-7 Larger Functional Blocks :A Multiplexer Circuit.

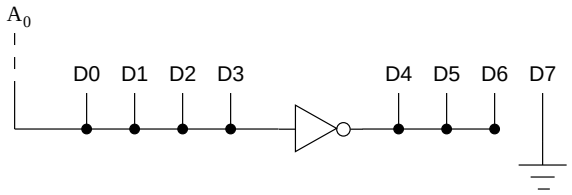
This exercise is optional. Try it if you have time, or just study it if you don't.

Wire up the 74LS151 8-input multiplexer as shown below. Connect an LED with current-limiting 390Ω resistor to the $\overline{Q}$ ("Q-bar") output as shown. Since LOW TTL outputs sink current better than HIGH outputs source current, we use $\overline{Q}$ to run the LED via a pullup resistor.



Set up a binary number (address) on the A, B, C inputs and ground the $\overline{STROBE}$ (i.e., *ENABLE*) input. Momentarily bring each one of the D_0 to D_7 to ground until you see the LED turn on and off. The input which causes this response is the input selected by the address. Repeat this procedure for a few different A, B, C addresses until you understand the chip's function. Note: the LED lights up when $\overline{Q}$ is LOW which means that Q is HIGH.

Add the inverter circuit shown below, and make the connections to the data input lines as indicated. Use one of the 74LS00 NAND gates wired as an inverter. Don't forget the power supply connections! The addition of the gate and wiring turns the multiplexer into a "31-day machine": the LED lights whenever a month number (Jan. = 0001, Feb. = 0010, Mar. = 0011, ...) is applied to the address lines $A_0 \ldots A_3$.



Try the circuit out and draw a truth table (i.e, the states of $A_0 \ldots A_3$ vs. Q). In the table, group the the months in pairs according to the patterns in the 3 most significant bits, and note how for each pair the "31-ness" of the month depends on the least bit A_0. Compare this with the wiring of the inverter circuit and discuss how the circuit does indeed indicate which month has 31 days.

Prepared by D. B. Pengra and J. Alferness
Parts adapted from *The Student Manual for the Art of Electronics*
by Thomas C, Hayes and Paul Horowitz (Cambridge University, 1989)
`Logic_Lab1_14.tex -- Updated 4 April 2014`