Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.111 - Introductory Digital Systems Laboratory (Fall 2005)

Laboratory 1 - Logic Gates

Issued: September 9, 2005
Checkoff and Report Due: September 23, 2005

Introduction

This lab assignment introduces you to important tools and devices that we will be using throughout the term. You will be introduced to the following:

- Tektronix oscilloscope and logic analyzer
- 74LS TTL series chips, including the ‘00, ‘04, ‘163, ‘393
- 74HC00, a CMOS NAND gate
- 1.8432 MHz Crystal Oscillator
- TTL and CMOS voltage levels
- Karnaugh Maps and Boolean Algebra
- Simple Verilog
- Simple Sequential and asynchronous circuits
- 7-segment Display

The following are relevant handouts that you should be using in conjunction with this lab:

- Lectures 1-5
- Safety Memo

Procedure

This lab is divided into several exercises to guide you through the design, construction, and debugging process. You will be asked to wire circuits for many of the exercises. Save all of these circuits until you have completed the lab as many of these circuits might be reused in subsequent parts of this lab.

1. Read and understand the whole assignment.
2. Design, build, test, debug, and fix each exercise in turn. Be sure to answer all the questions in the report template before you get checked off. You do NOT have to get checked-off on a exercise before proceeding to the next one. For each exercise, be sure to have the appropriate figures ready. Turn in the completed report template to your TA after your checkoff.

Exercise 1: TTL/CMOS Static Electrical Characteristics

The logic values of 1 and 0 are represented by voltage levels in the hardware logic implementation. The voltage levels and other electrical characteristics are not standardized from one logic family to another. 6.111 will use both TTL (Transistor-Transistor Logic) and CMOS (Complementary Metal-Oxide Semiconductor) logic. The voltage ranges for the two logic families are not compatible.

In this exercise, you will first measure the electrical characteristics of a TTL and CMOS gate using the circuit in Figure 1. Wire up this circuit using a 74LS00 part. Do not forget to wire power and ground! These connections are usually omitted from logic diagrams, as the power and ground of the 74LS series are generally the top-right and bottom-left pin, respectively. Typically, the top of the chip has a small semi-circular cutout, or a white dot next to pin 1.

Ground the input of the inverter (the first NAND) and measure the output voltage using the oscilloscope. Be sure to be using the voltage markers on the oscilloscope.

Connect the input to a logic ‘1’ and repeat the measurement.

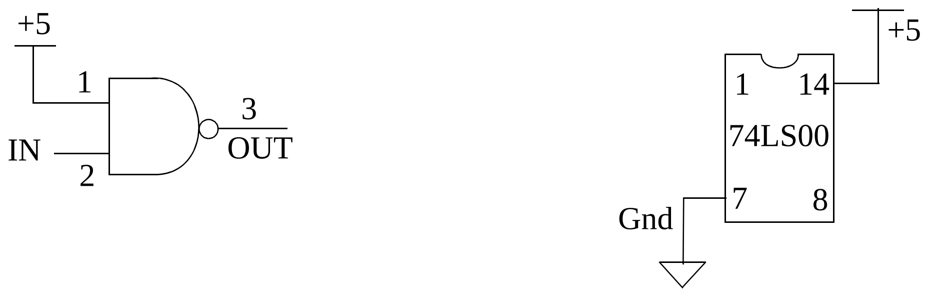

Figure 1: (a) Logic Level Measurement (Measure voltage at OUT node).
(b) Power Supply Wiring.

Next, use a 74HC00 and wire up the same circuit and hook V_{CC} to a +5V supply. Perform the same measurements and record your results.

Look up the valid input and output voltage ranges using the datasheet for a 74LS00 and a 74HC00. For each experiment, do your output values satisfy the range specified in the datasheets?

Consider interfacing a TTL inverter to a CMOS inverter and vice versa. Look at the datasheet titled “HCMOS Family Characteristics”. Based on the +5V supply you used, find out the recommended input voltage for HCMOS inputs. Discuss potential issues when interfacing TTL and CMOS components? Run the two experiments (interface TTL to CMOS and vice versa) and make voltage measurements.

Exercise 2: Build Your Own Ring Oscillator

Important timing parameters associated with the speed of digital logic gates are the propagation delay time t_{PD}, and the output signal rise and fall times, t_R and t_F. Propagation delay is a measure of how much time is required for a signal to change state. It is measured as the time from the 50% point of the input to the 50% point of the output (Figure 2). It is often cited as the average of the high-to-low and low-to-high delays (corresponding to the two transitions). The rise and fall times represent the amount of time for a signal to change state. To measure rise and fall times, you should be using the 10% to the 90% point, or vice versa.

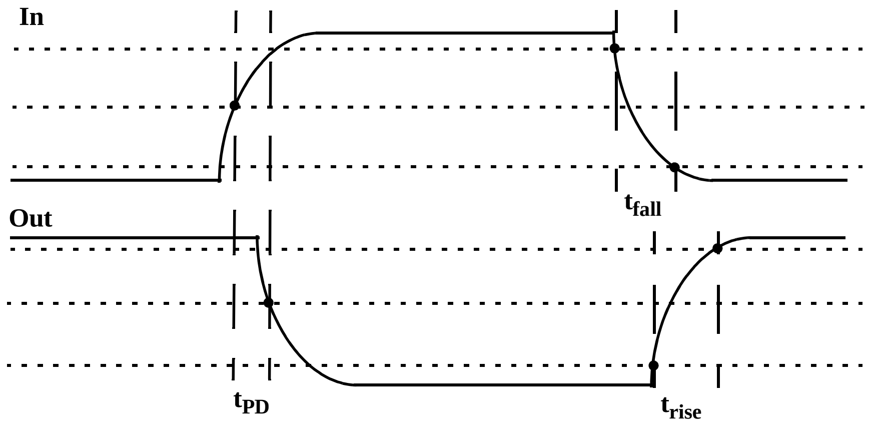

Figure 2: Timing Characteristics (10%, 50%, 90% marked).

Construct the ring oscillator shown in Figure 3 using a 74LS04 with as little wire as possible. From this circuit, determine the average propagation delay of a TTL inverter by measuring the period of oscillation by using the time markers on the oscilloscope. You can determine this by determining the number of gates a signal must travel through to complete a full period of oscillation.

What should the period of oscillation be with 3 inverters in the ring? Rewire the circuit and measure the period. Comment on the new result.

Insert a long piece of wire (about 3 feet) into the ring of 3 inverters. Observe how this extra length of circuit affects the signal. Can you explain the change?

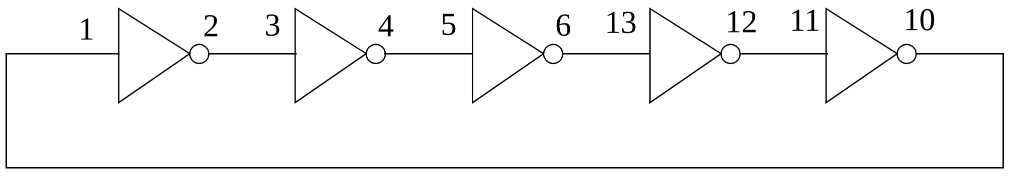

Figure 3: Ring Oscillator (using a 74LS04).

Finally, take a single inverter, and wire the output to the input. The voltage should be stable. If it isn't, connect a capacitor (0.01 μF) between this single node and ground to stabilize the voltage. Measure the voltage, and explain the significance of this voltage.

Exercise 3: Glitches

Wire up the circuit in Figure 4 using a 1.8432 MHz crystal oscillator and a 74LS00. Be sure to wire the crystal oscillator right side up. Pin 1 should be marked with a dot. The output GLIT is a purposely glitchy output caused by the circuit. Using the oscilloscope, measure the width of the glitch.

Next, add CLK and GLIT as signals to the logic analyzer and measure the width of the glitch. Is there a significant difference between your two different types of measurements? Why does this glitch occur, and what is the lesson learned from this exercise? Under what conditions is it a bad idea to use a glitchy signal as an input?

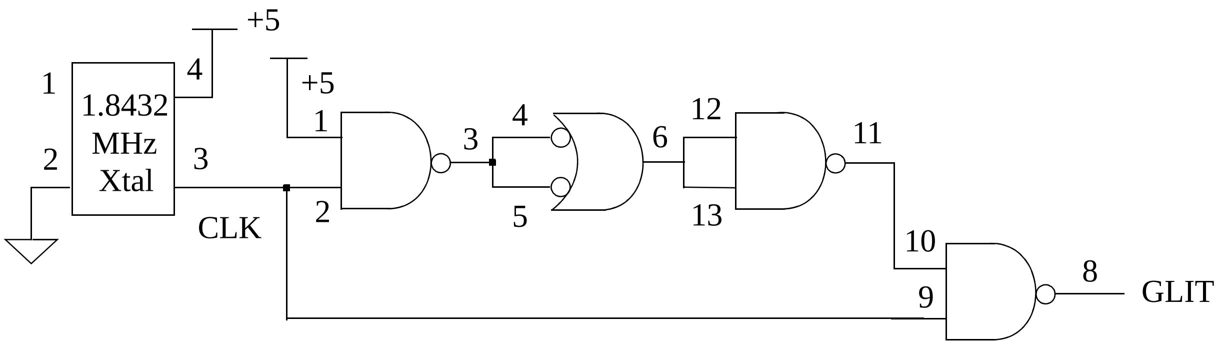

Figure 4: Glitch Measurement Circuit (74LS00).

Exercise 4: Asynchronous Counters

Wire up the 1.8432 MHz clock from the previous exercise to an 8-bit ripple counter as shown in Figure 5. You do not need to disconnect the previous circuit. In this exercise, we are concerned with how each output bit changes with respect to other bits. Since this counter increments its count every falling edge of the clock, each output bit must have a period twice that of the next significant bit. Because of this characteristic, counters make good clock dividers; if you want a slower frequency clock, it may be helpful to use a counter or a series of counters. Verify the counter's operation using the oscilloscope.

Trigger on either a rising or falling edge of the MSB, and measure the time between the falling-edge of CLK to edge of the MSB. Estimate the clk-to-q delay for each flip-flop in this chip.

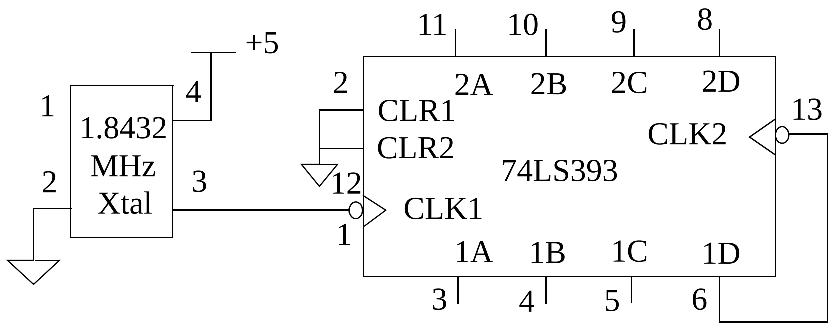

Figure 5: Clock and Ripple Counter.

Exercise 5: Synchronous Counter

One of the most useful of the 74LS series is the 74LS163. This is a 4-bit loadable synchronous counter with a synchronous reset. Wire two 74LS163s so that they count continuously as shown in Figure 6. The RCO output of the low-order counter is connected to the ENT input of the high-order counter. Explain when that signal is a 1 and how it controls the operation of the high-order counter.

Using the oscillosope, measure the delay from the rising edge of the clock to when the A output changes state. Do you get a noticeably different measurement for the B output when it changes state? Explain.

Use the logic analyzer in internal timing mode ("timing acquisition") to capture the A, B, C, D and RCO outputs of the low-order counter. Display the values of A, B, C and D as a 4-bit hex number and trigger the analyzer so that it starts its display when the count is 0. Note that as the count changes from one value to the next, eg, from 0001 to 0010, it may momentarily read as zero depending on when one bit turns off and the other bit turns on. How can you avoid these transitional values when triggering the logic analyzer?

It's possible for the RCO output to have momentary glitches after the rising edge of the clock. Explain the origin of these glitches. Use the "Glitch Trigger" feature of the logic analyzer to see if you can locate any RCO glitches.

Be prepared to comment on the difference between the 74LS163 and the 74LS393 in terms of design and performance; in particular, the speed and area of the device. By area, we are asking you to consider how complicated it is to implement the counter. How many flip-flops and how much logic is required to implement each counter?

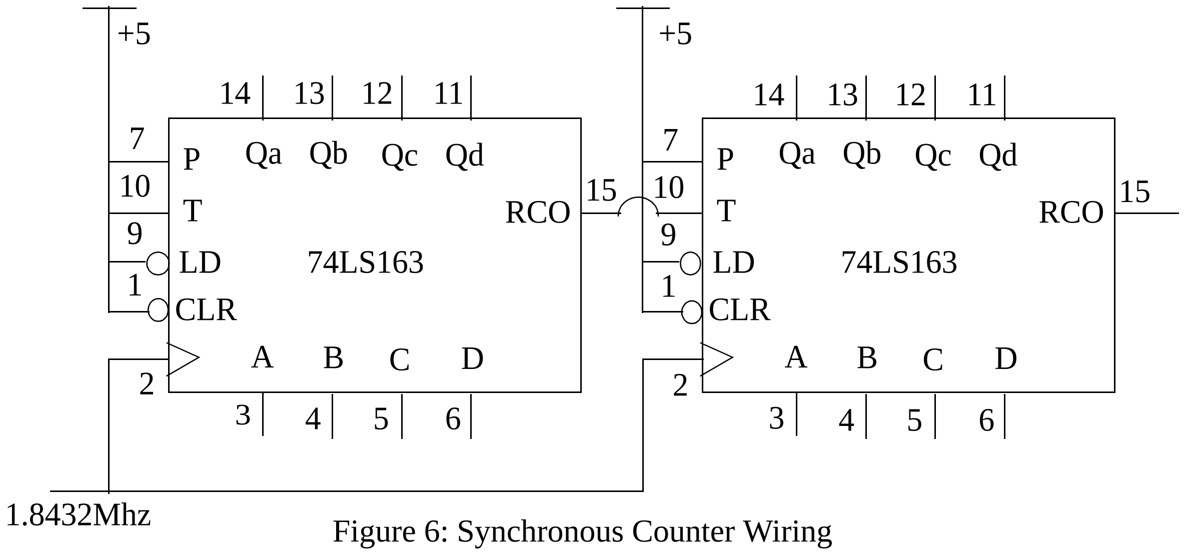

Figure 6: Synchronous Counter Wiring

Exercise 6: Designing Combinational Logic

In this exercise, you will be designing combinational logic that takes as input a 2-bit binary number and generates control signals that will properly illuminate a 7-segment display. Figure 7 shows the 4 possible digits.

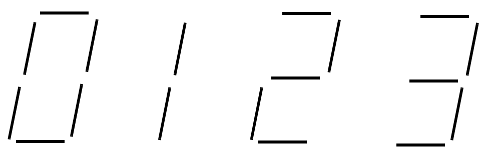

Figure 7: The first four digits on the seven-segment display

Using Karnaugh Maps, design the combinational logic that converts the 2-bit input into control signals for each of the 7 segments of the display. Fill out each Karnaugh Map, circle the terms in the Minimal Sum of Products and write the logic equation for each segment.

Draw a schematic diagram using only inverters and 2-input NAND gates that implements your logic equations. Construct the circuit on your protoboard using 74LS00s and 74LS04s (hint: you'll only need one of each IC) using a dipswitch to supply the 2-bit input as show in Figure 8. What's the propagation delay of your circuit?

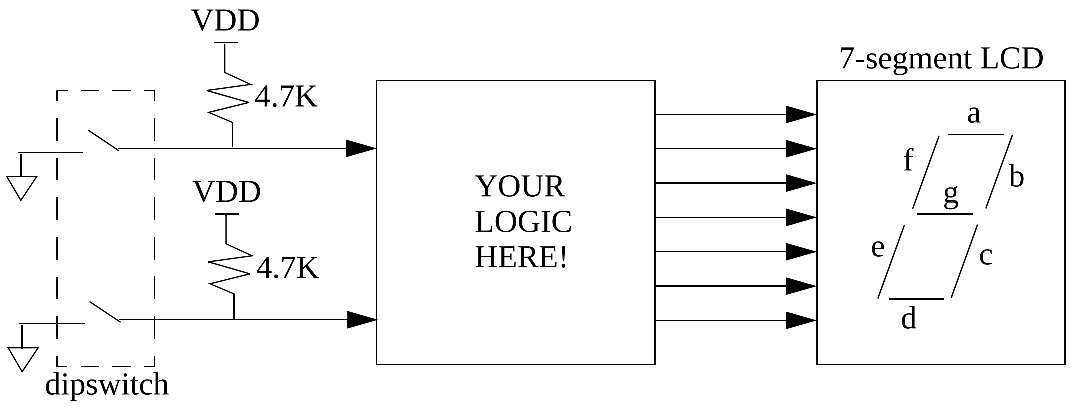

Figure 8: Block diagram for testing your decoding logic

Exercise 7: Writing Verilog code

In this exercise you'll design a Verilog module that implements a 74LS163. Here are the steps:

1. Log into one of the workstations in the Digital Lab. Your username is your Athena login name and your initial password is "changeme".
2. Download lab1.v from the Handouts page of the course website. Fire up a browser, navigate to this semester's website, click on Handouts, right-click on the lab1.v link and select Save Target As (or Save Link As), specify your home directory (U:) as the destination.
3. Start Modelsim by double-clicking its icon on your desktop. If it complains about a missing license file, go to Start -> Programs -> Modelsim -> Licensing Wizard. Click the Continue button and specify 27000@mtlcad.mit.edu as the location of the licensing file. If it offers to add the appropriate environment variable, let it! The wizard will complete, click OK when it's done. Now restart Modelsim and everything should be happy.
4. At the bottom of the Modelsim window there's a frame labeled "Transcript" where you can type in commands and see various messages from the simulator. Type in "`cd u:`" to change to your home directory.
5. Type "`vlib work`" to set up the simulator's working library
6. Type "`vlog lab1.v`" to compile the verilog file you downloaded above. Output from the compiler is displayed in the transcript window.
7. Type "`vsim test`" to start simulating the test module found in lab1.v.
8. Type "`run 2000ns`" to run the simulation for 2000ns. You should see the following print-out in the Transcript window

```
# Starting test of LS163...
# clr_bar was asserted, but counter didn't clear
# out = xxxx, expected 0000
# Break at lab1.v line 48
```

These messages were generated by code inside the test module as it runs through various tests of the LS163 module. The LS163 module supplied in lab1.v is empty which is why the test failed. Your job is to fill in the body for the LS163 module, implementing the correct functionality. Refer to the 74LS163 datasheet to see what functionality your code needs to implement.

You can use the editor of your choice to edit lab1.v appropriately; Modelsim has a simple built-in editor which should be displaying lab1.v after you completed step 8. As you edit lab1.v, repeat steps 6 through 8 above to test your code. When you're successful you'll see

```
# Starting test of LS163...
# Finished test of LS163...
```