Massachusetts Institute of Technology

Department of Electrical Engineering and Computer Science

6.111 - Introductory Digital Systems Laboratory

Laboratory 1

Logic Analyzers, Digital Oscilloscopes, and PALs

Issued – September 4, 2002

Due: September 16, 2002

Introduction

This initial lab assignment introduces you to the single most important measuring equipment that you will use to test and verify your digital designs. This instrument is a multi-channel logic analyzer with an integrated two-channel digital oscilloscope. In conjunction with learning about the use of this instrument, you are also required to interact with the laboratory computer systems to program a PAL and then to use the logic analyzer and digital oscilloscope to verify the correct operation of the digital circuitry implemented by this PAL. You are also required to implement the logic specified by the VHDL file with SSI (Small Scale Integration) and to verify what it does.

Before you begin, please read the handouts you have been given on the lab equipment and rules. These should answer many of your questions. If you have any other questions or problems – ask!

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 entire lab as many of these circuits will be reused in subsequent parts of this lab, perhaps even in later labs.

1. Read and understand the whole assignment.

2. Design everything - on paper as well as in your head.

3. Build, test, debug, and fix each section in turn. You do NOT have to get checked-off on a section before proceeding to the next section.

4. Correct the documentation.

5. Prepare your report which should include:

   - your name, recitation time, and T.A.'s name
   - logic diagrams for the PAL (you need not reproduce logic diagrams shown in this assignment)

- logic diagrams for the SSI circuitry used to implement the logic specified in the VHDL code. Do not forget to include pin numbers and location information.
- test results
- waveform sketches
- answers to specific questions posed in the exercises.

Exercise I: TTL Static Electrical Characteristics

The Boolean values of '1' and '0' are represented by voltage levels in electronic logic. The voltage levels and other electrical characteristics are not standardized from one logic family to another. The primary family of logic to be used in 6.111 is TTL (Transistor–Transistor Logic).

The first and most elementary gate developed in TTL is the NAND gate, the circuit of which is shown in Figure 1. Other gates in the TTL family performing various other logical functions will have different circuit topologies, but will be electrically compatible with each other.

Because of variations in the manufacturing process, it is not reasonable to expect that each gate will possess identical electrical characteristics. Rather, slight differences will be found from one batch of gates to another. The manufacturer, however, specifies an acceptable range of gate input and output voltage levels over which proper operation can be guaranteed. The manufacturer-guaranteed voltage ranges for inputs and outputs are shown in Figure 2 (for Texas Instruments' TTL integrated circuits). **THESE LEVELS ARE IMPORTANT: REMEMBER THEM!**

It is sometimes necessary to create fixed logic levels not derived from a TTL gate to properly terminate an unused input on a logic gate. To force a gate input to logic '0', the input may be directly grounded (zero volts). To force a gate input to logic '1', the input may be tied to +5 V as shown in Figure 3.

> NOTE: You will find that an unconnected input will often behave as if it were at LOGIC 1. However, sometimes it will seem like a LOGIC 0. It is **bad practice** to leave inputs unconnected, as they are then vulnerable to noise.

Wire two NAND gates from a 74LS00 chip as shown in Figure 4. Do not forget to wire power and ground! These connections are (most) often omitted from logic diagrams when the lower left pin is GND and the upper right pin is +5 volts. Often the top of the chip has a small circular cutout as shown in Figure 5. Amazingly, some ICs "almost work" when the ground wire is omitted.

Ground the input of the inverter and measure the output voltage. (Use an oscilloscope for voltage measurements; be sure it is calibrated.)

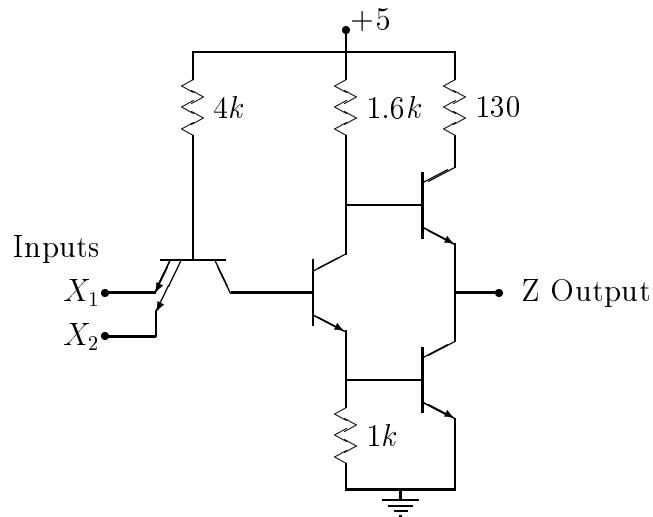Connect the input to logic '1' and repeat the measurement.
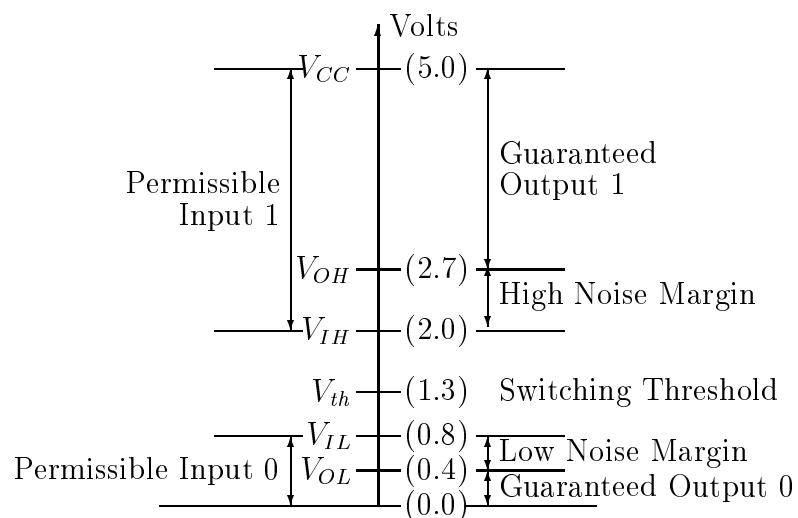
Figure 1: Basic TTL NAND Circuit

Figure 2: LSI TTL Voltage Ranges

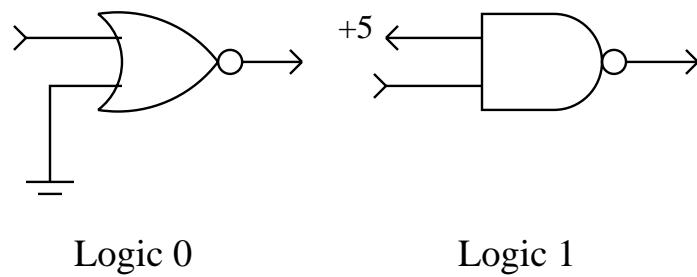Logic 0   Logic 1
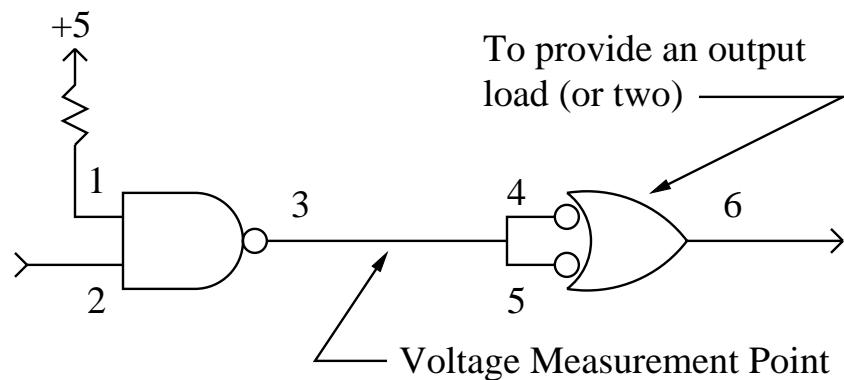
Figure 3: Termination of Unused Inputs.

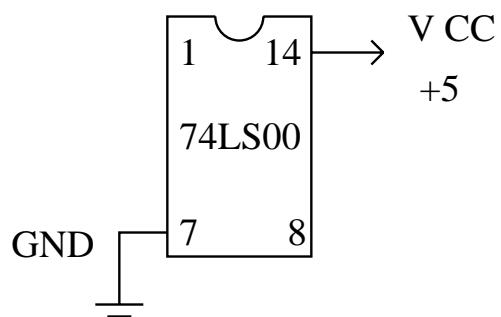Figure 4: Logic Level Measurements

Figure 5: Power Supply Wiring for 74LS00

Exercise II: TTL Dynamic Electrical Characteristics

Three 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 the electronic circuitry within the gate to change state. The rise and fall times measure how fast a binary signal can change state. These two dynamic factors are illustrated in Figure 6. V_{IL} and V_{IH} are thresholds which say when an input is valid. V_{OL} and V_{OH} are thresholds which say when outputs are valid.

The effect of propagation delay time can be demonstrated by the circuit of Figure 7.

Propagation Delay Time and Rise Time

Construct a ring oscillator using five TTL inverters (74LS04) and as little wire as is reasonably possible. From this circuit, determine the average propagation delay of a TTL INVERTER by measuring the period of oscillation.

HINT: Think about the number of gates a signal must travel through to complete a full period of oscillation.

Transmission Line Effects

Insert a long piece of wire (2-3 feet) in the ring. Observe how this extra length of circuit affects the signal.

MORAL: Minimize the wire length whenever possible.

One Gate

What happens when you let $n = 1$ and wire the output of an inverting gate to its input? (If it oscillates, add a capacitor from output to ground.)

Exercise III: Flip Flop Operation Times

Clock

A *pulse* may be defined (in the digital sense) as a signal which is in one state most of the time and transits to the opposite state only for periods of short (usually fixed) duration. Positive pulses are normally in the LOW state, changing to HIGH for short periods of time. The time in the HIGH state is thus the pulse width. A good practice to follow in designing digital systems is to NEVER route pulses through mechanical switches. If the switch were to be operated while the pulse was active, then the possible contact bounce might cause the output of the switch to appear as several pulses rather than only one. Capacitance and

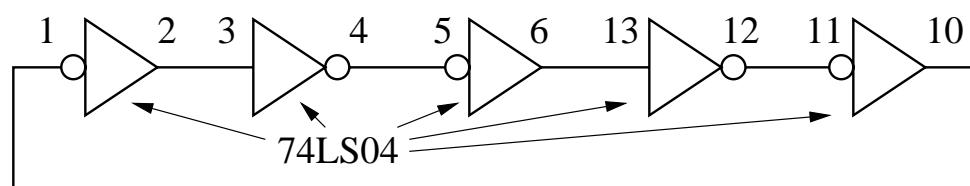Figure 6: Dynamic Characteristics Of A Gate
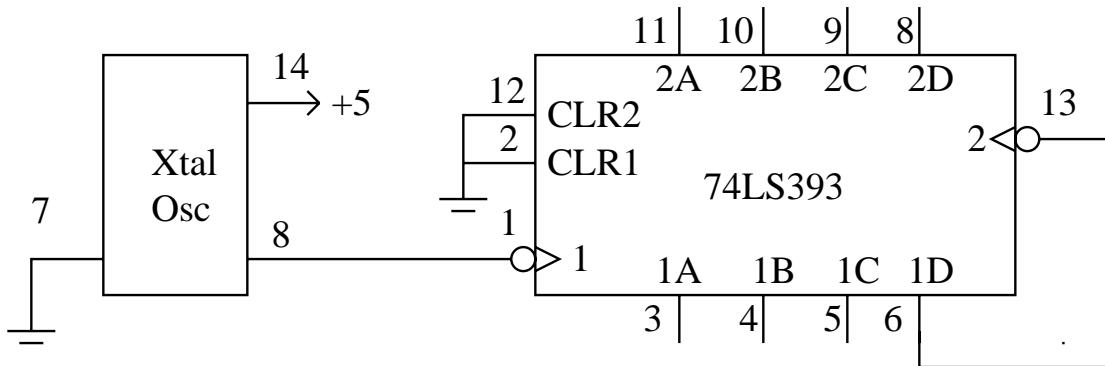
Figure 7: Ring Oscillator

Figure 8: Clock and Ripple Counter

inductance associated with the switch might degrade the rise and fall times or, worse yet, allow the pulse to "leak through" even though the switch was open.

A *clock* in a digital system is a signal which produces a (usually continuous) series of pulses spaced at regular intervals. We often use crystal oscillators as sources for clock waveforms. The *period* of the clock waveform is the cycle time or the length of time from the beginning of one pulse to the next occurrence of that pulse. The *duty cycle* of a clock waveform is the percentage of time that it is active.

Ripple Counter

Wire up a crystal oscillator output to an 8-bit ripple counter as shown in Figure 8 and verify its operation. Do not forget to wire power and ground! *Please* take care to put the crystal oscillator in RIGHT SIDE UP! They are destroyed when powered while upside down. Save this circuit for later use in this Lab.

Measure the delay from the falling edge of the crystal clock to a change in the most significant bit (MSB) of the 8-bit counter.

HINT: Use the MSB of the 8-bit counter as your scope sync.

What is the average operation time for a flip flop in the ripple counter?

HP Logic Analyzer/Digital Oscilloscope

Logic analyzers and digital oscilloscopes are complex instruments. They are also extremely powerful in aiding one to debug complex digital systems. You will find that time spent now in learning to use a logic analyzer will benefit you greatly and save you much time and effort in debugging later Labs and your final project.

Read the handout, "A Brief Introduction to the HP Logic Analyzer". Then come into

the lab and experiment with the logic analyzer. Additional documentation can be found in the pocket on the top of each logic analyzer, and several copies of the reference manual are placed in the lab.

Check out an HP Training Kit from the instrument room and go through all of those exercises to learn about the logic analyzer and oscilloscope.

DO NOT remove the HP documentation from the lab.

Exercise IV: Timing Analyzer Operation

Set up `Analyzer` 1 as a timing analyzer with a name of "Lab_1T". Assign `Pod 1` to analyzer 1. Set analyzer 2 to `Off`.

Go to the `FORMAT` menu and enter labels for CLK, Q1B, Q2D, RING, and /GLIT and assign these to bits 0 through 4 respectively. The `TTL` threshold should be selected. Positive polarity, `+`, should be selected. Set the `Acquisition mode` to `Transitional`.

Go to the `Trigger` menu. Change the `Base` to `Binary` for all signals. Set the trigger (`TRIGGER on ''Edge1'' 1 time`) to specify a falling edge trigger on Q2D which is bit 2.

Wire up the CLK, Q1B, and Q2D signals from your kit. Refer to Figure 8. Press the `RUN` button and manipulate the `X` and `O` markers to read the delay time from the falling edge of CLK to the falling edge of Q2D. To get at the `X` and `O` markers, you have to set `Markers` to `Time`. Press `RUN` several times to repeat the measurement. You can automate repeated runs by selecting the `Run` button (by holding down the mouse button or by using the arrow keys) and choosing `Repetitive` from the menu.

Change the `Trigger mode` back to `Single` and explore changing the `Time/Div`, `Delay`, and the `X` and `O` markers.

Now wire the RING bit to an output of your ring oscillator and measure the period. Change the logic analyzer to trigger on either edge of the RING bit.

Modify the wiring of your 74LS00 so as to conform to the circuit shown in Figure 9. Wire the /GLIT bit to the (purposely) glitchy output on pin 8 of the 74LS00.

Measure the width of the glitch.

Now change the `Acquisition mode` (select the `Format` menu) to `Glitch` and trigger the logic analyzer on any glitch of /GLIT. Make sure you can see a few full periods of Q2D.

Press `RUN` several times and sketch one of the more interesting traces of /GLIT.
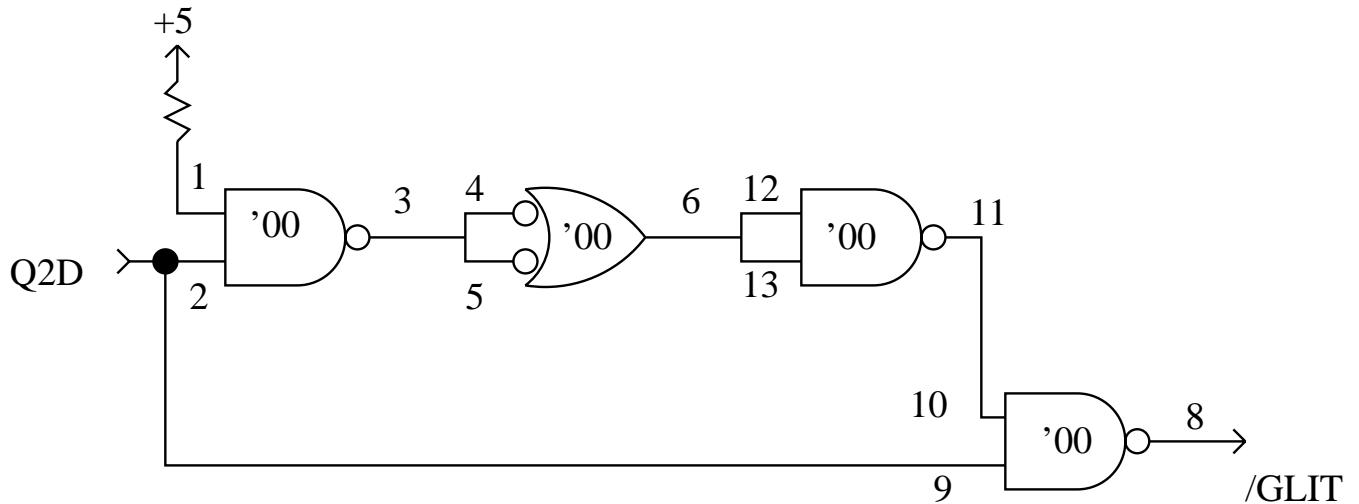
Figure 9: Glitch Measurement

Exercise V: Counter

One of the most useful of the TTL MSI ICs is a 4-bit counter such as the 74LS161/3. These two ICs are identical except that the clear input on the 161 is asynchronous (i.e., takes effect immediately when asserted) while on the 163 the counter only changes state as a result of the rising edge of the clock.

Wire two 74LS163s so that they always count, connecting an output of your ripple counter to its clock input. Trigger your scope on the most significant bit (MSB) and verify its operation. Save this circuit as you will need it for the rest of this Lab.

- About how long does it take, after the rising edge of the clock, for one of the flip flops to change state? This may be easier to measure if you use a fast clock or trigger the scope on the LSB of the counter. This delay is short and thus hard to measure.

- Can you observe any glitches on the carry output? Look at *both* carry outputs. Look carefully around the 0111 to 1000 transition. Glitches like these are hard to see as they are very short. The carry output does not *always* have glitches, just sometimes for some particular chips!

  You may or may not be able to detect a glitch on the carry output with a logic analyzer configured as a timing analyzer and triggered in glitch mode. The logic analyzer will catch glitches that are 5 ns or greater and may or may not catch shorter glitches.

Exercise VI: Programming a 20v8

Brief Description of a 20V8:

The 20V8 is electrically erasable. Mistakes are easy to correct by simply reprogramming the device with a corrected jedec file. The logic in the chip is programmable so that an

Figure 10: Output Architecture Configuration of 20V8

output can be combinational or from a D flip-flop. In either case, an output can be inverted or not. An output pin could be used as an input or the output could be tri-state. If any output is configured as a flip-flop, then pin 1 is the clock and pin 13 is the output enable. For this lab, some outputs will be a flip-flop, and all possible outputs (eight of them) will be used. Also, none of the outputs will be tri-stated. You will (should) ground pin 13 to always assert the output enable for the flip-flop outputs. Two appropriate configurations are shown in Figure 10.

The programming information for the configuration fuses (as well as for the logic terms) is contained in the jedec file. Each line in Figure 10 labeled ``FROM AND ARRAY`` represents a product term which can contain any input or output signal except pins 1 and 13 when any output is a flip-flop.

Creating your source file:

Actually, we have created the source file for you, so all you have to do is copy it to where you can compile it. You should do this on one of the lab computers or on a Project Athena Sun workstation. If you are using an Athena Sun workstation, remember to execute ``setup 6.111``. Don't use "add 6.111". Make a copy of the initial specification file by executing the commands:

```
cd
mkdir lab1
cd lab1
cp /mit/6.111/vhdl/lab1/lab1.vhd  lab1.vhd
chmod 600 lab1.vhd
```

Creating your jedec file:

Run galaxy to create your jedec file. You may wish to look at the handout, **A Beginner's Guide to WARP**. Set the device to be C20V8 and the package to be PALCE20V8-7PC. You may want to look at the report file generated when you created your jedec file.

Programming the PAL:

The 20v8 PALs are CMOS. Please ground yourself before touching them as you may otherwise burn them out with a high voltage resulting from static charge. Use the conductive foam which has been provided for storing and transporting PALs.

Program your PAL on one of the universal programmers. See the "PAL Programming" handout in this packet.

Exercise VII: Analyze and Test the Latch

- Draw a logic diagram using AND gates and NOR gates for the latch realized by the last two concurrent statements in the VHDL source file on page 15. You may use inverters as required.

- Implement an SSI version of a latch with 74LS00 NAND gates.

- Wire debounced switches to the input pins corresponding to S, R, LATCHCLK, and DATA for both the PAL and SSI versions. Wire LEDs to the outputs QH and QL.

- With LATCHCLK = low, test that both latches work as expected when you operate the switches connected to the S and R inputs.

- With S = R = low, test that the latch works as expected when you pulse the LATCHCLK input for the two possible values of DATA.

- What happens if you wire DATA to the complement of pin 22 (qh) and pulse the LATCHCLK input?

Dynamic Testing of Combinational Logic

Wire five counter outputs to pins 2 through 6 of your PAL as shown in Figure 11. Set up the scope so that it syncs on the falling edge of the MSB of the counter. Adjust the time base and position the trace so that one complete period of the MSB occupies 8 scale divisions. This makes it reasonably convenient to use a single scope probe to display a combinational logic waveform for all 32 possible input combinations. Demonstrate your scope waveforms to a T.A.

Exercise VIII: Analyze and Test Your Combinational Logic

- Provide sketches of the outputs X2, X1, and X0.

- Are there any static hazards in your realizations?

- Can you observe any glitches? Why or why not?

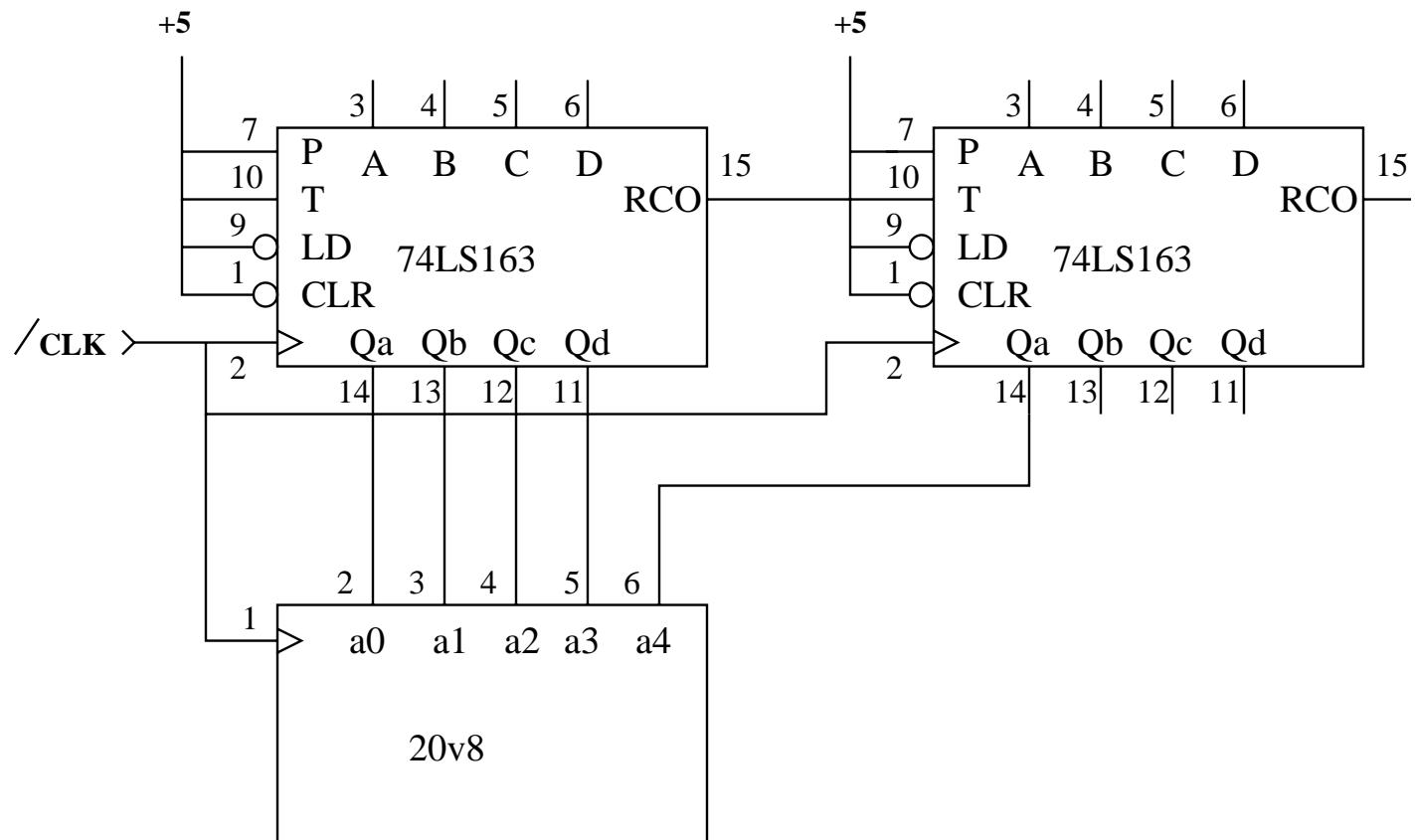- What is the delay through your combinational logic?

Figure 11: Counter To PAL Wiring

Exercise IX: Analyze and Test the T, D, and JK Flip Flops

- Implement these three FFs with 74LS74 and 74LS00 as required.

- What is the delay or operating time of a PAL flip flop? That is, what is the time from the rising edge of /FFCLK to a change in state of the flip flop? Measure this for a 74LS74.

- What do you think the setup and hold times are for both the PAL and SSI versions of the T flip-flop?

Exercise X: Logic Analyzer as a State Analyzer

Configure `Analyzer 1` as a `State` analyzer and assign `Pod 1` to it. Wire bits 0 through 3 to the label, ADDR. Wire bits 4, 5 and 6 to the label X and bits 7, 8, and 9 to the labels JK, T, and D respectively.

Wire the CLK signal (pin 1 of your PAL) to the CLK channel and enable the `J Clock` for a rising edge.

Go to the `TRIGGER` menu and set the trigger so that the analyzer will trigger on the second occurrence of the 4-bit address = 0. You should then be able to display a truth table of X2, X1, and X0 as a function of the address.

Change `Analyzer 1` to a `Timing` analyzer and display timing waveforms similar to those previously displayed on your oscilloscope trace. Also use the logic analyzer to verify the operation of the JK, T, and D flip flops.

Demo your proficiency with the logic analyzer to a T.A. who will initial and date your PAL listing (which you should include in your report).

The following is a listing of the vhdl input file:

```
-- This comment is before the library and use clauses.
library ieee;
use ieee.std_logic_1164.all;
-- here is the entity
entity lab1 is
  port (ffclk, a0, a1, a2, a3, a4, latchclk, data, s, r : in std_logic;
        x0, x1, x2 : out std_logic;
        dff, tff, jkff, ql, qh: buffer std_logic);
attribute pin_numbers of lab1:entity is
-- This attribute is implemented by Cypress (and not by Altera).
  " ffclk:1 a0:2 a1:3 a2:4 a3:5 a4:6 latchclk:7 data:8 s:9 r:10" &
  " x0:15 x1:16 x2:17 " &
  " dff:18 tff:19 jkff:20 ql:21 qh:22 ";
  -- don't forget to ground pin 13
-- having these two attributes makes the equations easier to understand
-- it also ensures symmetry
attribute synthesis_off of qh:signal is true;
attribute synthesis_off of ql:signal is true;
end lab1;
-- here is the architecture
architecture behavioral of lab1 is
begin
  -- x2, x1, x0 is (supposed to be) the integer part of
  -- the square root of a4, a3, a2, a1, a0
  -- concurent statements implementing x2, x1, and x0
  x2 <= a4;
  x1 <= (not a4) and (a3 or a2);
--   x0 <= not (((not a4) or a3)(a3 or (not a2)) and (a3 or a1 or a0);
  x0 <= (a3 and a1) or (a3 and a0) or (a3 and a2) or
        ((not a4) and (not a2) and a0) or ((not a4) and (not a2) and a1);
  -- a process implementing the three flip-flops
  process (ffclk) begin
    if rising_edge(ffclk) then
      dff <= a0;
      tff <= a1 xor tff;
      jkff <= (a2 and (not jkff)) or ((not a3) and jkff);
      -- note that extra parentheses are ok
    end if;
  end process;
  -- concurrent statements implementing qh and ql
  ql <= r or (not qh) or (latchclk and (not data));
  qh <= s or (not ql) or (latchclk and data);
end behavioral;
```

– This is an intentional blank page –