EE283 Debugging Guide for Digital Circuits

John Gilmer
Wilkes University
October 31, 2007

Introduction:

The course EE283 Electrical Measurements Lab features two projects that involve construction and demonstrations of digital circuits. The first of these is a combinational circuit that converts a four bit Grey Code value into binary, which is displayed using a seven segment decoder. The circuit is constructed on a solderless breadboard, using 74LS TTL integrated circuits. The second circuit is a digital counter / timer, also using the same technology.

The problem with both projects is that the circuits are constructed and tested outside of scheduled class hours. While there are open lab periods where students can seek help, there is a tendency to delay work on the project too long to take advantage of this opportunity. In addition, the graduate students in the lab at those times do not themselves have extensive experience with debugging. Diagnosis and debugging is very seldom addressed in classes or texts, leaving students without any good principles to follow in debugging their projects. It is usually a skill that must be learned by doing. Thus, there is a need for additional material on this subject, which this document is intended to address.

The examples in this guide are drawn primarily from the EE283 Grey Code Decoder Project. If opportunity permits, this document will later be expanded to include more specific examples from the counter project.

Basics:

When confronted by a circuit that does not work properly, there are several things to check first, regardless of the specifics of the particular circuit:

1) Power and Ground:

   Connect a Voltmeter with the common terminal at the power supply Ground, and check all of the Power (usually pin 14; pin 16 on 16 pin devices) pins. You should see 5 Volts on all of them. Be sure to put the Voltmeter probe on the pin of the IC itself, not the breadboard strip. This is important, because occasionally an IC pin will not go into the breadboard properly, as shown in Figure 1. If this happens to be the power pin, the whole IC won't work.

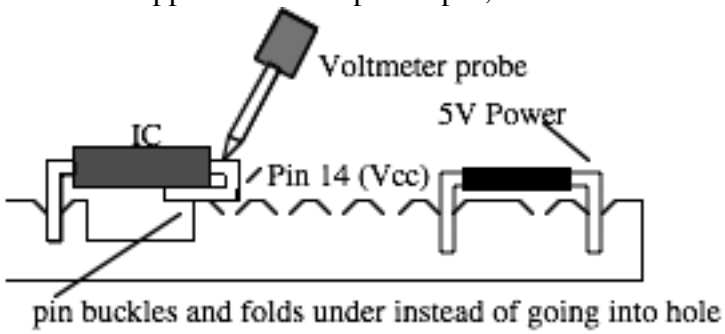

Figure 1 Testing of pins with a Voltmeter

After testing all of the power pins, check the Ground pins. Here, be alert for any slightly larger Voltage than what you measure on the adjacent Ground power bus strip. A buckled Ground pin may still have a Voltage close to zero, but usually it is high enough to raise suspicions. Often an IC may even appear to work properly without a good ground, but in fact it will fail under at least some conditions.

2) Check signal pins for odd Voltages

It isn't a bad idea to check the signal pins too. A signal output should have a Voltage either below .5 Volts (a good zero) or above about 3 Volts (a good one). If you see a Voltage in between, there's a good chance that this output is somehow shorted to another output, and they are pulling in opposite directions (a signal "conflict"). If you see this, pull out the wires connected to this output and see if the signal becomes a good one or zero. If so, plug in the connecting wires one at a time, checking the Voltage after each is connected. When you plug in one and get a bad Voltage, follow the wire and look for an error: it is probably connected to another output. Then go back to your schematic to find the problem. There should never be two outputs connected together. (As an exception, with "open collector" and "three state" devices, outputs often are connected together. But we do not use these in EE283. They do show up in EE241.)

When you check input signals, you might see a Voltage of about 1.7 Volts rather than a good one or zero. An input pin (or pins) that are not connected to any output are said to be "floating". TTL ICs like we are using register a floating input as a "one", but your circuit should not have any floating inputs. If an input is floating, a connection is incorrect or missing. Try to figure out which. One possibility is a buckled pin. Occasionally a gate, say a 3 input AND, is left with only two inputs, and the third input will be floating, unconnected to anything. In such cases connect the unused input to either one of the other inputs to the gate or through a 1K resistor to 5V power. It should not be left floating. (Floating inputs in sequential circuits, such as counters, are very dangerous. They pick up switching transients and cause the circuit to behave unpredictably.)

3) Check your inputs:

In the EE283 Grey Code decoder project, the inputs are taken from DIP switches. Often these are wired incorrectly. Put your Voltmeter probe on the output signal wire, being careful to touch exposed uninsulated wire, as shown in Figure 2. You should see 5 Volts or 0 Volts, depending on which way the switch is set. Often the DIP switch isn't pressed in firmly, and does not make good contact. Sometimes resistor leads are clipped to short, and do not make good contact. Often the grounds are missing. Or the signal is connected to the wrong side of the DIP switch. Notice the principle involved: the resistor is pulling the signal up (weakly, since there is a 1K Ohm series resistor) but when the switch closes (is "On") the input is shorted to Ground. You can actually get by with using a much larger resistor most of the time, if power drain (5mA with the switch closed to give a "0") is an issue.

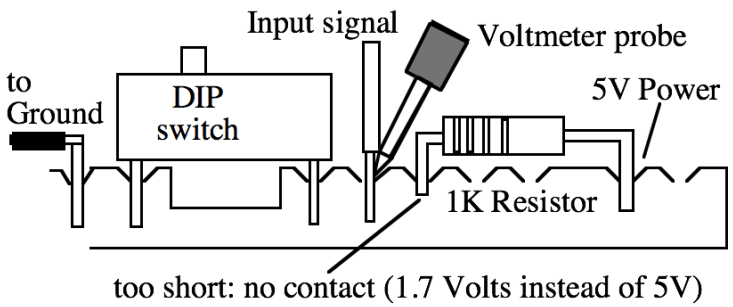

Figure 2 Input testing

4) Check your Outputs

Use either a DIP switch or a probe with a 5V 1K Ohm pull-up (or an equivalent to the DIP switch circuit of Figure 2) to test the outputs of your circuit. You should disconnect the circuit from the outputs, then try and make sure that the output devices work for all of the combinations of signals they will see.

For the EE283 Grey Code project, the output circuit includes a 7 segment display and a 74LS47 seven segment decoder device, as shown schematically in Figure 3 below. The 74LS47 device has some convenient diagnostic features. The "BR" (Ripple Blanking) and "LT" (Lamp Test) inputs should normally be pulled high (using a 1K resistor). You can check the output segments by pulling "LT" low by running a wire from Ground to this pin. (If you use the same resistor to pull up both "LT" and "RB", disconnect the resistor from "LT" first.) All of the segments of the 7 Segment LED display should come on. If they do not, check the power to the display, and the various resistors and wires connecting the 74LS47 to the display. Use a Voltmeter at each end of each connection, from the 74LS47 pin to the wire connecting the display. (Do NOT test segments with a wire to ground; it will blow out the segment since the common anode for the display LEDs are connected directly to power. You can check the LEDs with a 1K resistor probe to ground.) When RB is pulled down, all segments should go off. This should adequately test the output circuit itself.

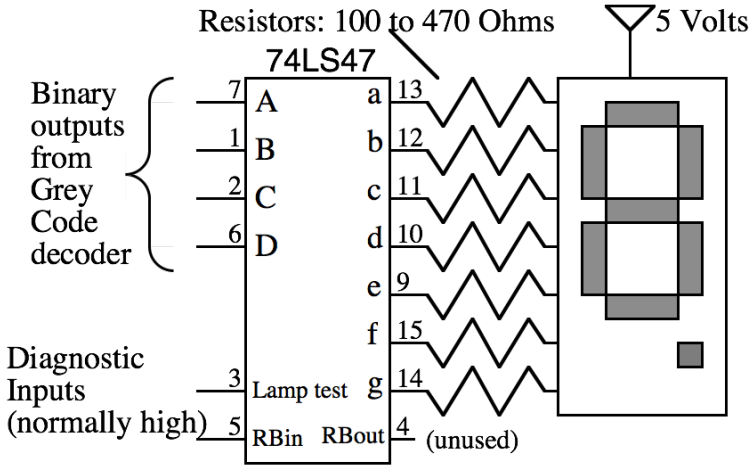

Figure 3 Seven Segment Decoder Output

You can then walk through all 16 combinations of 1's and 0's for 4 bit binary values DCBA and ensure that you get 0 to 9 and then several weird characters, and blank for 1111. (The 74LS47 is designed for and is expecting only decimal numbers. What it does with 1010 to 1110 is arbitrary but predictable.) Once you have done this, you know that the output circuitry is working correctly.

The Example:

The debugging process will be illustrated using the variable "C" circuit given in the lab instructions, shown in schematic form in Figure 4 below. One thing that is important for debugging is to have a good, properly annotated schematic. That includes labeling the ICs (U1, U2, etc.) and the pins, as shown in the figure. That way you can find the signals you want to check. The schematic should not try to portray the physical arrangement. (However, the simpler the physical arrangement, the easier debugging is. Note that most gates go from the "left" end of the IC toward the right. So it makes sense to put the 7404 inverter to the left side, the 7408 AND gates (and 7411's) in the middle, and the 7432 Ors toward the right. That means fewer crossed wires, and easier debugging. Color coding helps a lot too.) Figure 5 shows a possible physical arrangement of this circuit.

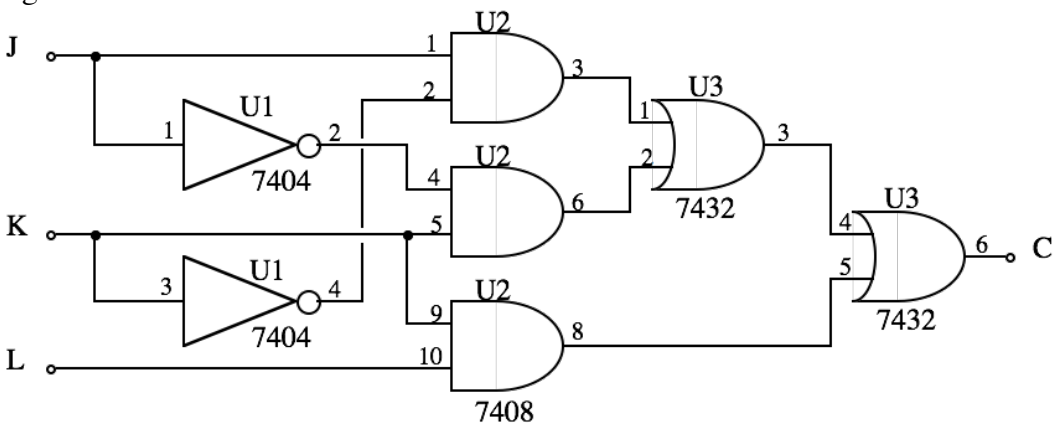

Figure 4 Example Combinational Logic Circuit Schematic

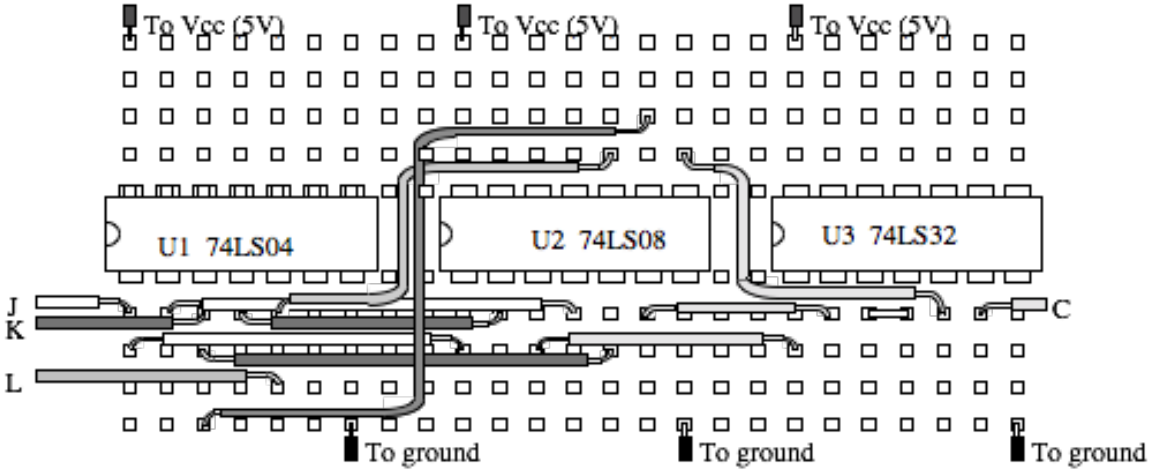

Figure 5 Circuit as constructed

The first step is to identify a "fault" condition. This is a set of values on the inputs that produces the wrong output. For this circuit, let us suppose that when JKLM = 0000 we get C=1. We know that this is incorrect. With this input, we should get DCBA=0000. (We might observe erroneous output for other outputs and conditions too, but we concentrate on these one at a time.) So, looking at the schematic, and finding that C=1 where it should be 0, we identify the IC and pin from which C comes. It is U3, pin 6. We will want to probe this node. You can use a meter. But if you want to simultaneously probe several nodes, you need to use something else. The simple probe shown in Figure 6 is usually effective. This probe will indicate when you have a "1" on a signal. It won't indicate that you have a bad "0" or floating value. For that, use the probe shown in Figure 7.

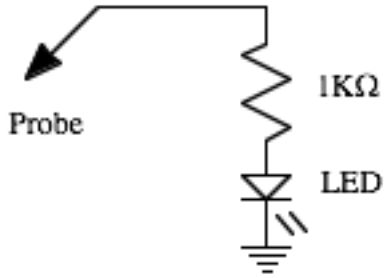

Figure 6 Simple Probe

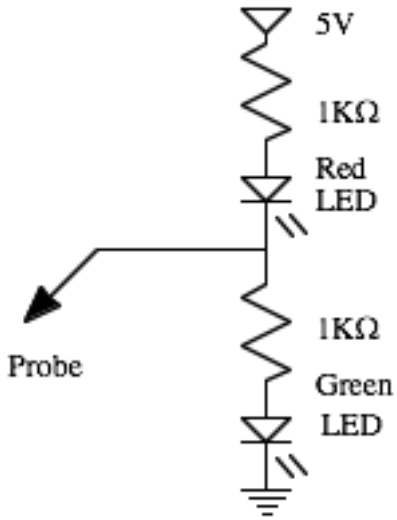

Figure 7 A Better Probe

With the two LED probe, the Green LED on shows a good "1", the Red LED on shows a good "0" and both on dimly shows that the probe is either not connected to a good output or it is a floating node. (If neither is on, check power!)

So, now probe pin 6 of U3. We see a "1", the expected wrong value. Now it's time to apply logic. U3 is an "OR" gate. The only way we can get a "1" out (if it is working correctly) is if one or the other inputs are "1". Since we

should be getting out a “0”, then both inputs should be “0”. One of them must be, erroneously, a “1” instead. So, we check them. Putting our probe first on pin 5, we find a “0”. That’s good; it’s what we expected. So, now we check pin 4. It is a “1”! It should also be a “0”, but we expected this. Since we had a fault on “C,” the fault had to come from somewhere. This brings us to the state shown in Figure 8. The correct values are shown, but for “C” and the signal going into pin 4 of U3 we see “0->1” indicating that the fault (wherever it is) has changed the 0 to a 1.

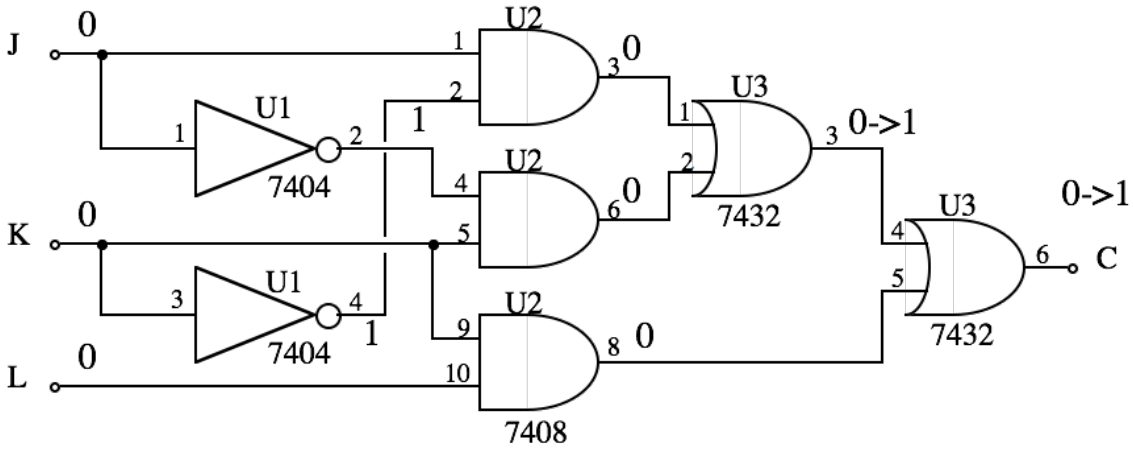

Figure 7 Illustrating the Debugging Process

Now, we just apply the same principle over again. The signal at U3 pin 4 comes from where? Pin 3 of the same device, adjacent. So we probe pin 4 to see if the fault is present there as well. It is. Pin 3 is the output of another OR gate, so one of the inputs, pin 2 or pin 1, must be wrong. Probing tham, we find that pin 2 is the guilty party. It is a “1” and should be a “0”, while pin 1 is a correctly a “0”. That means the fault may lie with U2, gate 4-5-6, the gate that feeds U3 pin 2. So, we probe U2, pin 6. Yes, sure enough, we see a “1”. It should be a “0”.

U2 is an AND gate device. So for the output to be a “1”, both inputs must be “1”. We see that with JKLM=0000, one of the inputs, pin 4, should be a “1” since it comes from an inverter connected to “J”. So the fault must lie with pin 5. (It wouldn’t hurt to check 6 too.) So, we probe pin 5. We see 1.7 Volts! (If we are using the probe, we see the LED off, and think this is a good “0”, but we’d have missed the symptom.) It should be a good “0” since this pin is connected directly to input “K”. We go back and check the input “K” at the DIP switch. It’s 0.11 Volts, a good zero, as expected. How can this be? Looking at the physical arrangement, we see that the wire to pin 5 of U2 actually goes to pin 3 of U1. We check U1, pin 3. It’s OK! So the problem must be in the wire.

Upon closer inspection, we find that the end of the wire is actually broken: The place where the insulation was stripped off was nicked, and the wire just appeared to be intact because a couple of other wires held it down, hiding the break. Figure 8 illustrates. To fix this, we pull out the wire, and use a sharp

Exacto knife or equivalent to wedge out the broken off fragment (so we don't lose the use of that hole on our breadboard). We'll need to be more careful stripping wire to avoid such nicks! Now we replace the wire. Does C work correctly for JKLM=0000? Yes! So, now we try all the other combinations. If we fin d another fault, we do the same procedure. Sooner or later, we will find them all, and our circuit will work.

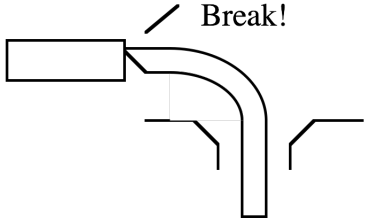

Figure 8: Wire break at the breadboard

Sequential Circuits:

For sequential circuits, those that include state storing components such as counters, latches, and memory, there are other things that can go wrong. The combinational subcircuits can be debugged separately using the procedures above. But the sequence may still be wrong, usually because of one or more of the following problems.

1) Failure to "bypass" the power supply adequately.

This is what tantalum capacitors are for. There should be one close to each sequential IC. Figure 9 shows one way of doing this. The Tantalum capacitor could be placed on the bus strip instead. Beware! Tantalum capacitors are "polarized". One side needs to be positive with respect to the other. There should be a small "+" on the positive (5V) side, and the positive lead is usually longer (if the leads have not been clipped). Somewhere, you should also have a large aluminum electrolytic capacitor (about 1000 μF is what we usually use) across the power supply (5V to Ground). That handles the lower frequency noise; the tantalums handle the higher frequencies.

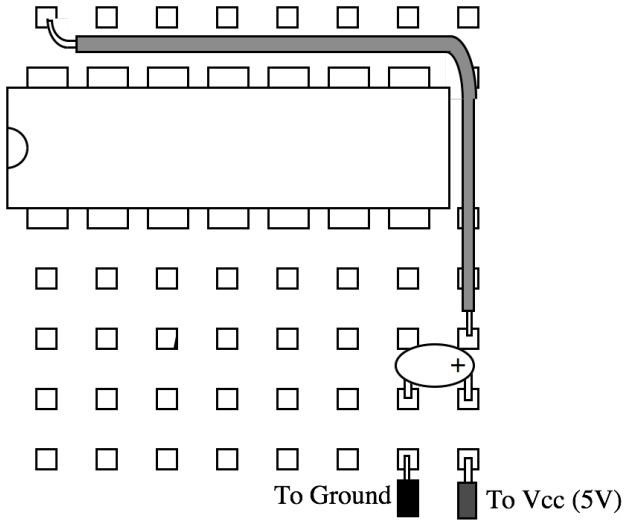

Figure 9 Tantalum Bypass Capacitor, Example of Use

2) Be sure that all unused signals, especially clocks and enables, are pulled securely high or low. Usually unused signals need to be pulled high. You can pull up more than one with a single 1K Ohm resistor. It's best to pull them up to the same side of the chip where the power supply is bypassed.
3) When you check power and ground, use an oscilloscope if you are having problems. Any power or ground on which you see some sort of signal has a problem. Look for bad connections or broken pins.
4) Try substituting a slow "clock" for your normal clock, so that transitions happen slowly enough for you to see, and you can probe different points using your meter or an LED probe. (However, probing sensitive points can sometimes create a glitch just from the stray capacitance.) A useful slow pushbutton clock can be formed from two NAND gates (or AND + Invert) as shown in Figure 10. If you do not have a SPDT pushbutton, ask for one. If none is available, you can use two SPST pushbuttons that you push alternately.

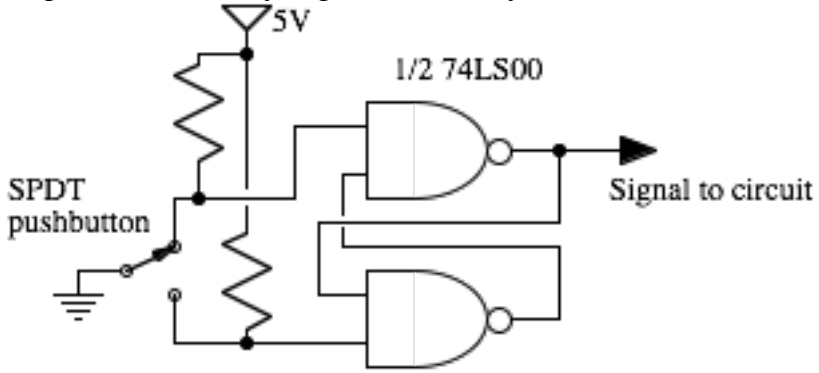

Figure 10 "Debounced" pushbutton circuit that you can use as a slow clock

Conclusion:

Debugging is at least as much art as science. There's no substitute for creating a circuit with faults, and then having to exercise logic and your reasoning ability to find and fix the bugs. You learn more by doing this than by building a fault free circuit. Ultimately, you will find that these principles that apply so straightforwardly to these digital circuits apply equally well to all sorts of problems, in all sorts of domains.