Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.111 - Introductory Digital Systems Laboratory

How to Make Your 6.111 Project Work

There are a few "tricks of the trade" which allow an experienced digital designer to fix problems quickly. This note is an attempt to fill you in on some with which you might not be familiar.

Wiring Errors

1. Do you have a diagram of your circuit? Not that piece of napkin from common's last night, I mean a DIAGRAM. If you don't, then think about making one. Making the diagram may in itself lead you to finding the problem. A timing diagram is often helpful for finding errors as well.

2. Now that you have a diagram, it should be easy to have your partner check the wiring. Not you. Your partner. You will make the same mistake that you made in wiring your circuit a second time when you "check" it.

3. Check each IC package. Is there a wire on each of its pins? There probably should be, in most cases. Explain the cases where there is not (unused section of 74LS00, parallel inputs not used on counter . . .). You would be surprised at how many errors are caught with this simple technique. Be careful especially on the power and ground signals.

4. The pin assignments of a 74LS02 are not the same as those of a 74LS00. When you use them, double check the pin numbers.

Care and Feeding of the Power Supply

1. Don't overload the kit power supply. If in doubt, measure the power supply with an OSCILLOSCOPE, not a VOM. Many of the power supply overload problems show themselves as hum or ripple on a supply rather than low voltage output.

2. "Grid" your power supply distribution. This means construct an X-Y pattern of power interconnections for the ground and power. The extra wires are redundant if you believe you are working with superconducting zero inductance wires. You are not. Recall that KCL requires that signals which force current to flow OUT a wire also cause the SAME CURRENT to flow BACK on the ground and power wires.

3. This means that if you are, for example, building a project with multiple kits, you should be quite careful to provide enough grounds between the kits. I'd say a conservative minimum is about 1 ground wire for each 3 signal wires. The ground wires should connect diverse points on the grid of each kit, not all be connected to the power

supply pin. Don't connect the +5 volt power on adjacent kits together. You can check the adequacy of your grounding between kits by measuring the ground potential difference between points in your project with a scope. Worry about anything more than about 0.5 volts.

4. You may have heard from an analog course about "ground loops" and the necessity to avoid them. In digital systems it is completely impractical to avoid ground loops, and the opposite approach is taken, best summarized in a phrase which I will pass on: "Let Ground Abound".

5. If you take care with the ground distribution in your projects, few bypass capacitors will be necessary, but for conservatism you might consider placing a 0.01 to 0.1 ceramic bypass capacitor between the ground and +5 power grid every half a dozen packages or so. Your oscilloscope is your best friend here. Look at the +5 power runs while your project is running. Does it look like a DC power supply? Anything more than about 1 volt p-p noise needs to be fixed. Sometimes adding power supply bypassing can actually make a project not work. The way this happens is that the ground distribution system is high impedance, and the addition of the decoupling capacitors makes the current in the ground lines higher, leading to ground noise. Since the noise margin in the low state is worse than in the high state, this can be a bad tradeoff.

Unused Inputs

They have to be connected to either ground or to "high". Ground is easy to come by. "High" can be the output of a grounded input inverter, or a resistor tied to the positive supply. The resistor value is non-critical, somewhere in the range of 1K. Many inputs can be wires in parallel to this resistor. Making the run extremely long can be bad from the standpoint of finding errors. It is common, and the source of difficult problems, for the high run to have some output connected to it. Since you often pull up sets and clears on widely distributed circuits, this can cause very bizarre behavior. Keeping the number of inputs on the high run to a reasonable number (5-10) will help isolate these problems.

Behavior of Ungrounded Parts

The behavior of parts which do not have a ground pin connected can be quite disconcerting. They seem to almost work. The reason for this is that the input signals can provide a source of negative supply. So, if you have a NAND gate, for example, in a package with no other sections used, then it can behave correctly for the case when the inputs are both low, one or the other high, but incorrectly if both are high. This can be MOST confusing, and is even worse if you are using all four sections of some NAND gate, since you are (almost) certain to have some input low, and the gates will almost work. Looking at your logic signals with a scope will discover this and other problems, since the signal levels will look very poor.

Tri-State Logic Signals

Busses which have tri-state signals driving them are tricky to debug at times. Scoping them can give signals which appear to be "middle" due to the parts not driving the bus at all times. One way to counteract this is to explicitly add pullup resistors to the runs, which will force the bus to be a logic "1" even if it is undriven.

A second problem with tri-state busses can happen when two outputs are "fighting" one another over the bus. Large currents and overheating of the design can occur under these circumstances, and the logic levels on the bus will be "middle".

Thermal debugging of designs is a quite efficient test strategy, since it is very quick and can find some problems with very little work. Check to see that all of your parts are dissipating some power. Are any so hot that you can't touch them? Unless you are using some part with an excuse to be hot, then some output is probably trying to pull down the +5 volt power supply, or fighting with some other part.

Open Collector Signals

Calculate the resistor value carefully. To do this, calculate the total input current of all the loads on the run. Subtract this from the output drive available from the driver part. If more than one type of part is being used, use the lowest output drive. The remaining current is available to pull down the resistor. The resistor value is then given by:

$R = \frac{5}{I}$ohms. This will usually be in the several hundred ohm range.

"PNP" Inputs

Newer TTL parts, those with numbers above 300 for the most part, use a low DC current input circuit using a PNP transistor. These parts have a very low TTL current, which might make you think that you can drive hundreds of them with a single part. This strategy is dangerous if you value the time and signal quality of your design. AC considerations mean that you should limit the number of signals on a run to the 10-15 range regardless of the DC current specification. The same comment applies, with even more force, to the MOS inputs of static memory parts and EPROMS. These parts have high input capacitance which makes driving them with a reasonable number of loads essential.

Handling CMOS Parts

CMOS devices are static sensitive. They have sufficiently high input impedance that the static charge stored on your body capacitance can permanently damage them. Ground yourself to the bench and your project before touching these devices, and they will work when you are finished plugging them in.

Wire Routing

Wires, as we mentioned above in the section on power supplies, are not perfect components. They have a parasitic inductance, resistance, and a mutual inductance with other wires. There are some strategies for minimizing the effect of these imperfections in our ideal

wire model.

For many of the signals in a design, the bad properties of the wires are irrelevant, for the same reason that we can afford to ignore combinational logic hazards–we don't look at the wires until they behave well. This works fine except for the wires used for timing signals in our design. Here are some general guidelines if you want to be careful:

1. Keep wires short. The shorter your wires are, the less chance of noise pickup, coupling, and other undesirable behavior.

2. Drive a wire from one end or the other of the string of places it goes, not from the middle.

3. Wire all the places a signal goes sequentially, one after another, not as some sort of tree-structure. This applies to single electrical runs.

4. If you have trouble with making your signals look good on a run, consider resistor termination of the far end of the run. Your friend here is your oscilloscope. Does the signal look like it has clean edges in the positive and negative going directions? Is there ringing or strong over and undershoot? Your termination will have to be a Thevenin equivalent resistor to about +3 volts with an impedance of about 150 ohms (bet you thought you would never hear about those again, didn't you . . .).

5. Undershoot on LS series logic, particularly, can lead to drastic changes in the guaranteed Tplh and Tphl speeds for parts. If you can't understand why the part which says it should be a 10ns part takes 20ns to switch, check for the 1.5 volt undershoot on its input. Remember that the spec sheet is your contract with the part manufacturer. He doesn't guarantee the behavior if you are feeding garbage into the part.

6. Bundling wires together makes your kit look neat, but it also makes a good transformer. Unfortunately, you probably didn't want a transformer between you logic signals. For clock signals, route the critical wires far away from other wires, particularly wide parallel busses switching at the same time. You might consider twisted pair, which wraps a ground wire around the signal wire. Ground both ends of the twisted pair ground.

Clock Distribution

Clocks, and write pulses for RAM's are the most sensitive logic signals in a typical design. The design methodology which we are teaching in this course allows most signals in a machine to have hazards, ringing, and even be slow, without actually making the circuit you are designing not work. The penalty for sloppiness in most signals is simply that the circuit

will operate slower, rather than not at all.

This is not the case with clock signals. If a clock has ringing on it, or a slow rise time, then your circuit may not work AT ALL. It pays then to take special care in distributing the clock. Loading rules should be strictly obeyed (they should be anyway, of course).

A more insidious problem sometimes occurs, however, due to the possibility of CLOCK SKEW. This means when the clocks for different portions of your logic design have their rising edges occur at slightly different times. This can happen as a result of long wires in the clock circuit, from different numbers of gates in series with different clocks in the machine, from different loading on different clocks, and from lots of other reasons. Whatever the reasons, clock skew is a common source of problems. To see why, consider a pair of type D flip-flop registers, such as LS374 parts, each clocked by a different clock. It may be your intention to load data first into register 1 on the first clock, and then into register 2 from the outputs of register 1 on a second clock. If the clock on register 1 lags the clock on register 2 by more than the sum of the setup and propagation delays of the part, then the incorrect data will be loaded into register 2: namely, register 2 will receive the data NEWLY clocked into register 1. So it is important to keep the location of the clock edge synchronized over the entire diagram. This is hard sometimes.

One technique to use is to build a "tree" of clock distribution: The main clock generator drives a set of four gates, each of whose outputs fans out to four more gates, whose 16 outputs each drive a section of the machine's clock. Try to keep wire lengths the same and to keep the clock runs as short as possible. Loads should be distributed evenly across all of the clocks available. With the logic family you are using in the course, clock skew should not be a major problem. You should be aware of its existence, though.

Gating the Clock

Don't do it. This is one of the most common problems novice digital designers encounter. Assume you are using positive edge triggered logic. Your clock can be thought of then as a "low asserted" signal. If you were to gate if off, you would use a positive OR gate to hold it high. Unfortunately, you need to start holding it high BEFORE it goes LOW. Gating it off after the clock goes low will actually just make a positive transition of the clock occur slightly early.

Clock skews can also arise from being careless in gating clocks. If you absolutely can't contain yourself, and must gate a clock, use a clock from before the final clock fanout to compensate for the gate delay in the gating circuit.

Gating of a clock can almost always be avoided by using registers with a clock enable input. Counter and shift register parts can also be forced into the "do nothing" state easily.

RAM Write Pulses

Write pulses for static memories are likely to be one of the few areas where you do need to gate a clock-like waveform with a logic signal. Since the RAM parts available act as latches, it is important that the signals used to gate the write enable off, and that the address in-

put and clock enable signals arrive early enough during the clock enable cycle not to cause glitches in the RAM functioning. To help you meet the Thold requirement of some parts, it is often helpful if your timing generator terminates the write pulse slightly prior to the rising edge of the main clock, assuring an adequate Thold. You can often make the clock signal of a design have a short enough Tlow such that it can be used as both the clock edge for registers and as the write-enable for RAM parts. Thold for the RAM address and data lines is sometimes a problem with this strategy, however.

Synchronizer Errors

Whenever you clock a signal which is asynchronous with the clocking waveform, there is a probability of producing a meta-stable state in the clocked flip-flop. For low-frequency clocks the probability of this meta-stable behavior is low. For LS flip-flops, a flip-flop output will have either set or not set by about 200 ns after the clock edge with very high probability. If you are designing a system which depends on synchronizing external signals faster than this, consider using a 74AS74 flip-flop as the synchronizer, since it is much faster than the LS parts.

Testing Strategies

You should be able to help yourself considerably in the testing of your project if you include in the design a means of controlling all of the finite-state machines. This means that you should be able to start the machine in a known state, preferably in any state, so that you can check out subroutines individually. You should be able to single step the machine to track down incorrect behavior. Making the machine loop at full speed doing some repetitive task is also a helpful debugging strategy, since you can then use a scope to examine the timing of the device directly. Those of you working with TV monitors will find that the monitor itself is a good debugging tool.

Driving High Current Devices

Those of you working with stepping motors and other high power devices should take care in the power/ground routing of motor power. The motor should be driven with a power supply other than the logic supply, and here, despite what I said above, you should worry about ground loops. Make the ground pin of the driver parts the common point for the motor and logic grounds, and otherwise strictly segregate them.

You should use power diodes as flyback protection across your motor windings. A 1N4001 would do nicely.