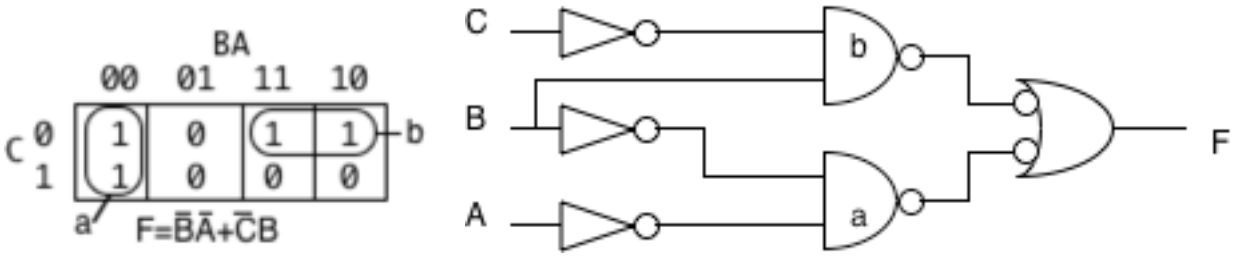
EE241 Digital Design
Lab #6 Timing Analysis
Report due March 23, 2018

Objective: Gain and understanding of static hazards and propagation delay in combinational logic. Also, to become familiar with the principles and use of a logic analyzer. This instrument is a very useful tool for observing the operation of digital circuits, especially computers.

About Hazards and Glitches: preliminary work

A hazard exists in a combinational function when, with only one variable changing, the output should stay unchanged but it in fact changes from 1 to 0 and back to 1, or from 0 to 1 and back to 0. The reason is that the signal follows two paths through the logic that have different speeds, and sometimes these are different enough to cause a "glitch", a very short temporary change in an output. Some glitches may be more complex, such as a signal that starts at 1, goes to 0, returns to 1, then finally reaches its final state at 0. See the text, section 4.4, for further background. Sometimes glitches are undesirable, especially in any signal such as a clock that synchronizes other signals, or is counted. Often they do not matter, and this is really true in our case. The human eye is not responsive enough to notice a glitch. But we will use this circuit to study them anyway.

First, look in your K-maps for an output function that has a hazard, and may have a glitch. This occurs where there is a "crack" in a Karnough map. For example, suppose the function F that drives segment f has a K-map as follows:



Suppose the inputs CBA are 010. Gate "b" is responsible for making sure the output, F, is driven high. (If neither gate a nor b is "on" (low, since they are NAND gates in NAND-NAND logic) then F will be 0. When the input changes to 000, gate "b" turns off and gate "a" turns on. If these happened at the same instant, we would see no glitch. In this case, gate "a" receives B', which must go through an inverter, and hence may well be delayed enough so that gate "b" turns F off before gate "a" can turn it back on again. In the opposite direction, 000 to 010, we would be unlikely to see a glitch, because "b" will probably turn on before "a" turns off. We are not guaranteed to see a glitch, because the switching may be too fast or near simultaneous. But with LSTTL, we probably will get a glitch of about 10 nsec.

So, look for such a hazard in your own K-maps, and predict where a glitch will occur. That is the signal you will want to watch. Set up your three inputs to switch back and forth both ways. In this case, we would put 000 / 010 / 000 on the three sets of 3 DIP switched to expose the hazard. (Remember, we only want one input to change. Simultaneous changes in two inputs can also cause glitches, but we are not looking for that kind. If, by some quirk of circumstance

you have no hazards, that is what you will have to do. You ought to identify several glitches if possible, so that if you do not find one, you can look for a different one.

Instructions for using the Logic Analyzer (NI Virtual Instrument version):

1. We will be using logic analyzers which are built into the National Instruments "Virtual Instrument" oscilloscope unit. (There are some other analyzers available which do not depend on the NI units, but the NI units are both convenient and seem to be effective.) These logic analyzers can sample up to 24 signals. A demonstration will be performed to show how they work (March 2 or 16, depending on how March 2 goes).

You should do this exercise together with another student. Select a working multiplexed 7 segment display project of one of the students to use as the subject of this lab. Later, do the same exercise with the other one. (Do not apply power yet.)

2. Determine a segment and setting that is expected generate a glitch. Examine the K-maps for the decoder circuit (from the formal lab report) and find a segment that has a static hazard: there are two adjacent cells which are not covered by a gate. Note which segment this is, and the two three bit binary values that should generate a glitch on that signal. Set the DIP switches so that the codes go from one of the codes to the other and then back. For example, if there is a hazard on segment "a" between binary cells for 6 and 7, the DIP switches would be set to 110 - 111 – 110 so that you cross the boundary in both directions. (You could try it as 111-110-111 as well, since that may give different results due to timing features in the counter and multiplexor.)

3. Connect the Logic Analyzer timing probe to various parts of the circuit. There is a big multi-pin socket on the front of the logic analyzer (60 pins?). Find the connector for it (in the overhead bin if in SLC224). Plug in the connector. Each of the pins is led out to a wire that has a socket on the end which is the right size to receive a piece of stripped 22AWG hookup wire. The leads with black insulation are grounds. Connect them all to Ground, and it is best to connect them all to ground on your breadboard (using black hookup wire) near your clock and counter. (Try to keep the connecting wires as short as possible without being too inconvenient.).

The signal inputs are in groups of eight. Signals D0 to D7 are the red wires from the connector, on the top row with D0 leftmost facing the NI VI. To distinguish which signal is which, use a piece of black hookup wire for D0, Brown for D1, Red for D2, Yellow for D4 and so forth, following the color code if possible, within each group of 8. Signals D8 to D15 have blue wires from the connector, and so forth. There are LOTS of signals so you can watch everything you want to! The following is a suggested set of signals (based on the limitation of nine for old logic analyzers. You may need to vary from this depending on your design.)

black: signal D0	clock output (oscillator)
brown: signal D1	first counter Q (one of two signals to the sequencing logic)
red: signal D2	second counter Q (the other signal to the sequencing logic)
orange: signal D3	control signal for digit 0 (or an input to your multiplexer)
yellow: signal D4	control signal for digit 1 (or an input to your multiplexer)
green: signal D5	control signal for digit 2 (or an input to your multiplexer)
blue: signal D6	bit 0 on bus (or one that is changing to excite the hazard)
violet: signal D7	bit 1 on bus (or one that is changing to excite the hazard)
grey: signal D8	segment a output or some other segment where hazard is expected.

4. Bring up the NI Virtual Instrument by turning it on, and clicking on the icon to start it running that appears on your computer's window. (Actually, you can do this before step 2.) This start up the VI application. You will see the window and have some options to select. (You CAN simultaneously use the oscilloscope to watch two signals in analog mode if you wish. This might be most appropriate and interesting for the bus signals. If you do so, be sure to use real oscilloscope probes in "x10" mode.)

5. Apply power to your circuit. (Do not do this before powering up the Logic Analyzer!)

6. Set up the logic analyzer. Under the channel 1 and channel 2 (red, yellow) controls there is a "Digital" (blue) control button. Click it to enable the logic analyzer. To the right of the blue button a "..." button appears when you move the cursor over it. Click on it. This brings up a dialog box where you can adjust the acquisition rate (the default of 1Gsample/sec is fine), the amount of memory to use, and which channels to collect. You should click on D0 through D8 (or more if you are using others). (By default, it starts with only D0 to D3.) Traces for those signals will be seen on the screen. You can rearrange the order of the traces by clicking and dragging in the display, should you want to. For the "trigger" you can designate a signal and ask for a "rising" or "falling" edge, just as you do for an oscilloscope. You might want to trigger on the clock at first, but later switch the trigger to the signal on which you want to find a glitch.

7. Now we are ready to acquire data. Put the oscilloscope in "Auto" mode, and collect some data, then "Stop". It really works very much like an oscilloscope, but you just see 0's and 1's (instead of a Voltage) and there are lots of traces. The analyzer will acquire the data and display it. You can manipulate which signal you trigger on and the time base to look for a glitch.

8. Once you find a glitch or something else interesting, you need to measure the timing relationships. (It can't hurt to do a screen capture, too. I suggest "White background" mode be used for captured images; they will print better, and not use so much ink or toner.) Now you can measure timing delays through the circuit using the cursors. The cursors are accessed with a button under the display. You want time cursors. You can drag the cursors back and forth and read the time distance between them. This is how to make time measurements.

9. Make observations on the timing in your circuit. Look for glitches. You should find one where you predicted it, on either the 0 to 1 transition or the 0 to 1. If you do not, you could try to find a different glitch. If you have a long clock, you may find that shifting to a higher frequency will make it easier to find glitches, which occur when you go from one digit to the next. If you are using an LM555, you could shift to the 4 MHz clock. (Be sure to bypass it!)

10. Shut down your circuit & disconnect probes before turning off the Virtual Instrument.

Analysis:

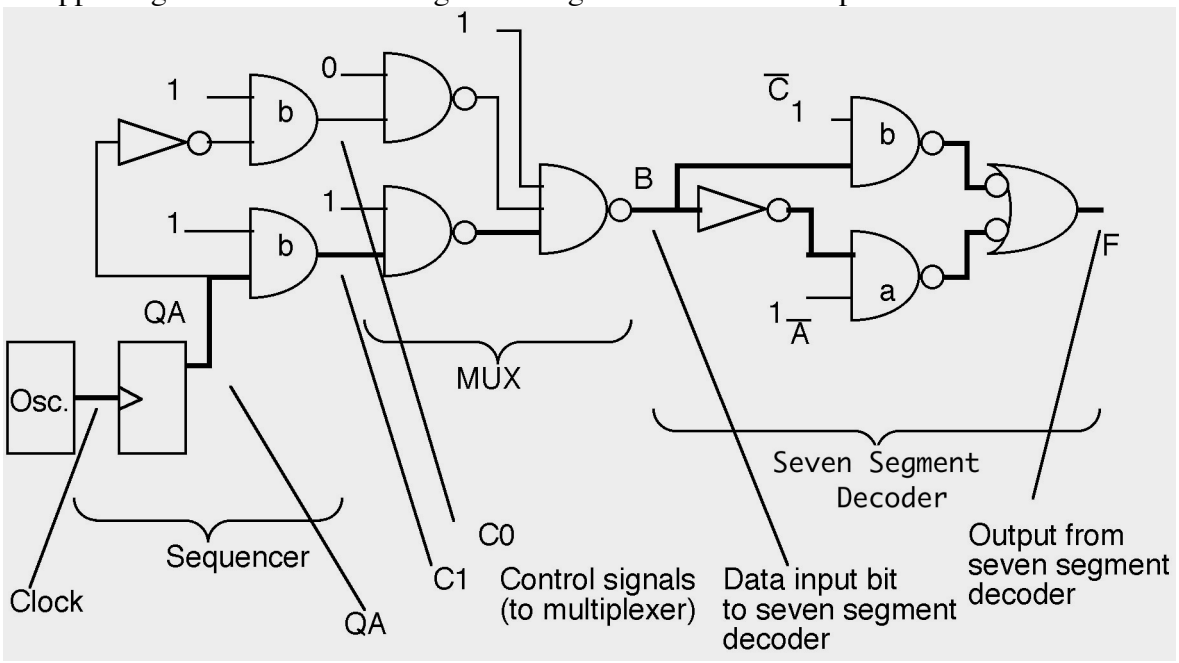
You are to account for why you see the glitches you do (at least the one of them that you expected) in your circuit. The informal report for this lab exercise is due at the beginning of the following lab session (March 23). You will have to analyze the timing in your decoder circuit. You may find Logicworks or ORCAD/PSpice helpful for your report. You can also move the

nine timing probes around while collecting data, or use more than nine, to get more data on particular parts of your circuit, to follow a particular glitch as it is formed.

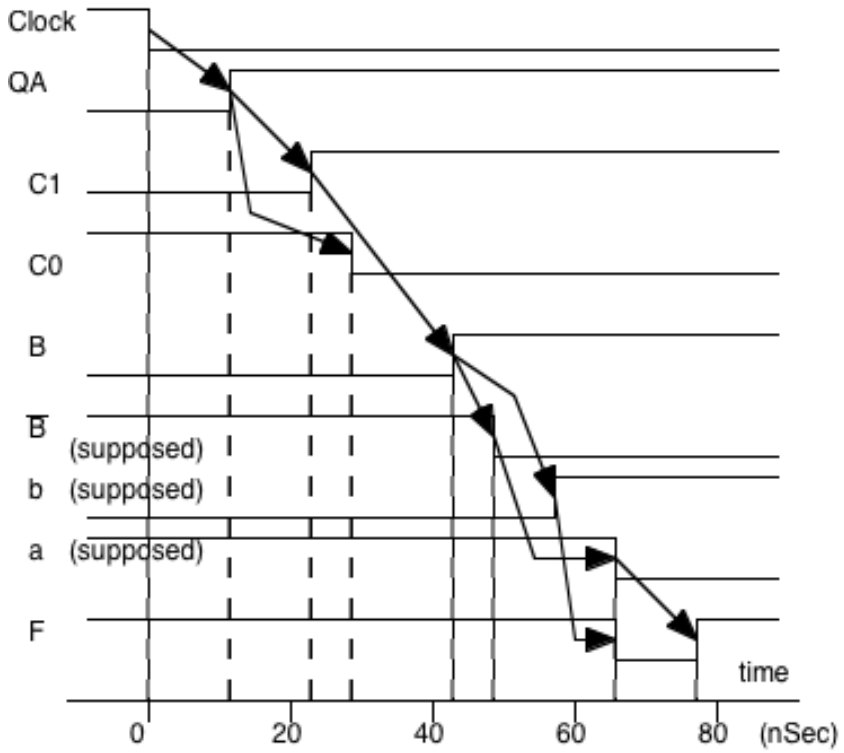
Diagram the critical path in your circuit. The diagram below is representative for the example given above, assuming that the multiplexer is constructed of NAND gates (Yours will be different). You should be able to explain how the glitch occurs in reference to both the critical paths here and the timing diagram derived from the Logic Analyzer. You should use a diagram similar to that below to explain how the glitch occurs. In all cases, there will be two different paths through the logic that need to be explained. It may be necessary to suppose certain timing relationships that have not been observed, such as those for signal B', and gates "a" and "b" internal to the seven-segment decoder. (Better, since you have lots of inputs to the logic analyzer, to capture all signals along the critical signal paths.) The expected timing delays through different types of gates, for example, a typical gate delay of about 9 nSec for a 74LS00 gate, may be important, and should be referenced. If you use MSI logic, such as the 74LS153 or 74LS244, you will need to look up information on those devices. If you use open collector logic (74LS03) you may need to estimate rise time given the resistance you use and capacitance driven, which will be fairly large with our breadboards (maybe as much as 50pF). You can (and should!) observe the rise time on the oscilloscope at the same time as you collect the data.

Report:

Turn in an informal report showing your relevant K-map and logic, the logic analyzer results (these may be neatly sketched, but must be well annotated), and your critical path analysis with supporting discussion describing how the glitch occurs. The report is due March 23.



Critical path Example



Timing Analysis Diagram Example

Note: This is NOT just a copy of your logic analyzer output. It is annotated and interpreted. It shows causation, not just arbitrary timing differences. That is, each arrow shows how a transition on one signal causes a later transition on another signal. If you have a glitch, it is because at some point there are two separate signal paths, shown by two separate chains of arrows, that both arrive at the final signal (a decoder output) at different times.