EE 231L Lab 3

Design and Implementation of Sequential Circuits

In this lab we are going to investigate and build several sequential circuits. The behavior of sequential systems depend not only on the current values of the input variables, but also on the sequence of input values that occurred in the past. Such systems have some kind of storage or memory elements. In this lab we are going to construct a debounced switch. We are also going to design an up-down counter using various methods. Finally we will construct some of the sequential components of the final digital computer.

Part 1. Debounced Switch

A switch is a mechanical device and as such is much slower than an electronic circuit. When a switch is opened or closed the mechanical contacts do not break or make a connection instantaneously, but can "bounce" between open and closed, thus making several transitions. If you were to use a mechanical switch to increment a counter (to count, say , people going through a turnstile), a single closure of the switch could increment the counter many times.
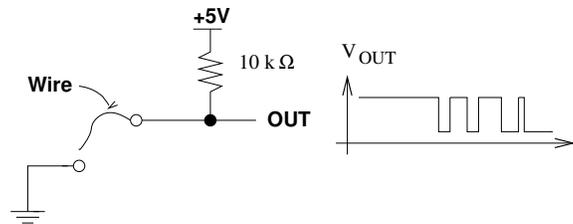
Figure 1: A simple switch

1. Build the switch of Figure 1. For now, just use a wire as the switch. Plug the wire into GND to bring OUT (the switch output) low, alternatively, unplug it to bring OUT high.

2. Test the circuit with a logic probe and make sure it works as described above.

3. Test the circuit with the logic analyzer. Connect the output of the switch to channel 0 of the logic analyzer. Start with your switch closed (the wire connected to GND). Setting up the Logic Analyzer:

   (a) In the upper right-hand corner of the screen is drop down box with a clock speed. Select a 5 MHz internal clock.

   (b) Right-click in the leftmost column of the viewing area and select Delete All Labels.

   (c) Right-click in the leftmost column of the viewing area and select Add Label. Give it a name , and choose channel number 0. (To view more than one signal at a time, just add additional labels and select different channels.)

   (d) Click on the button with the single running man and pull the wire out. The logic analyzer will sample the logic level on Channel 0.

   (e) Observe the Waveform window. You can use the Magnifying Glass icons on the menu bar to zoom in and out. See if you can observe switch bounce.

4. You couldn't see the switch bounce in the above part because the logic analyzer is much faster than you are. By the time you pulled out the wire, the analyzer had already finished sampling, so you were not able to observe the low-to high transition of the switch. In order to observe this transition you need to have the logic analyzer stop shortly after the signal goes high. In other words, you need to trigger the analyzer to stop after observing the desired transition. To trigger the analyzer on the low-to high transition of the switch do the following:

   (a) Click on the "Trig" icon. Click on the X over channel 0, Click on the up-arrow. This will cause the logic analyzer to capture the waveform when the input changes from low to high. After the line goes high it will sample for a bit longer, then show the logic trace both before and after the trigger condition.

   (b) Click the OK button and you are ready to sample.

   (c) Now click on the single running man, then pull the wire out of GND. The logic analyzer should stop and display the waveform. Did you observe switch bounce?

5. Capture several waveforms. How many bounces do you typically get? How long are the bounces? (Ask your TA if you do not know how to measure the length of bounces).

6. Repeat with the switch initially in the open position.

Debouncing your switch with an SR latch

Build the circuit shown in section 10.3.4 (pp. 718-719) of Brown. Use 10k Ω resistors. Use a wire to simulate your switch. Connect R, S, Q, and Q' to the logic analyzer. Observe the outputs when you toggle your switch. Does this circuit eliminate the bouncing?

Building a real debounced switch

1. The switch you will use is a single-pole double-throw switch. This is found in your digital lab kit. When your switch is oriented with the hinge on top and pins facing down (into board):

   • The common leads will be the two on top (connect to VCC).

   • The normally closed lead is the one on the right, and the normally open lead is on the left.

   • Build a debounced switch by substituting your single-pole double-throw switch for the wire.

2. Again, observe your circuit using the logic analyzer.

Leave the debounced switch on your board. You will be using it in the next section.

Part 2. Up-Down Counter with Enable

Here you will design a 2-bit up-down counter. The input, UP, determines the count direction. If UP = 1, the circuit counts up with the sequence ...00, 01, 10, 11, 00, ... If UP = 0, the circuit counts down with the sequence ... 00, 11, 10, 01, 00, ...

Design this counter using 4 different methods:

1. Use discrete components and D flip-flops.

2. Create as a graphical design file in Altera (see Getting Started With the Graphic Design Editor for instructions on how to enter a graphical design file). You can use your design from part 1 if you wish. Simulate this design to show that it works.

   NOTE: The CLRN and PRN of the D flipflop are clear and preset inputs. CLRN resets the flipflop to 0, while PRN presets it to 1. These are active low inputs, so to disable these inputs you should connect them to ___.

3. Create using a state transition table in Altera. Simulate to show that it works. (Discussion of how to implement sequential circuits in AHDL will be posted by the time that you need it.)

4. Create using D flip-flops and If-Then statements in Altera. Simulate and show that this design works.

Choose one of the 4 designs above and implement it (build the circuit). Connect the outputs to LED's to view the count sequence. Use 1 K Ω resistors to connect the LEDs to ground. For the clock input, use your debounced switch. Have your TA or lab instructor verify your circuit counts correctly.

What would happen if you used a switch that was not debounced for a clock input? Try it.

Part 3. Computer Registers

You should now be able to implement the registers for the final computer in AHDL. (HINT: You will want to use `DFFE`'s instead of `DFF`'s. Look them up in Altera's Help.) There are five 8-bit registers in the final computer: `PC` (program counter), `MAR` (Memory Addressing Register), `OUT` (output), `ACCA` (Accumulator A), and `INST` (Instruction Register). There is also a single one-bit register, `C` (What simple circuit element is `C`?).

`MAR`, `OUT`, `ACCA`, and `INST` are all 8 bit registers with synchronous parallel load. These registers all have a clock input, an 8 bit data input, and an active low load/enable input, as well as an 8 bit output.

For example, when `MAR_L` (Memory Addressing Register Load) is VCC the tt MAR register is not enabled. When `MAR_L` goes low the tt MAR register is enabled and on the next clock pulse the 8 bit data on the input line is loaded into the tt MAR register.

The `PC` is an 8-bit register with synchronous parallel load capability, synchronous count, and an asynchronous reset. The `PC` has a clock input, an 8 bit data input, and 3 additional inputs: `PC_L`, `PC_I`, and `RESET`. These three inputs are all active low. `PC_L` loads the program counter, `PC_I` increments the program counter by 1, and `RESET` resets the program counter to 0.

Implement these registers (synchronous load and synchronous load/count) as Altera functions. Include each one in a higher-level design file. Verify that both functions work.