BEE 271 Digital circuits and systems
Summer 2016
Lab 4: Keypad debounce

1 Objectives

In this lab, you will modify the keypad scanner you built in lab 3 to add debounce logic and modify the display to scroll the characters typed.

1. When a key is pressed, the new digit should be added to the right end of the display and the previous digits shifted left.
2. Debounce logic should be included ensure that if you press a key only once, you get only one digit.
3. One of the pushbutton switches should provide a reset function, blanking all the digits.

By the end of the lab, you should be comfortable designing, building and debugging a useful clocked sequential circuit in Verilog and you will have learned how a classic problem of working with mechanical switches is solved.

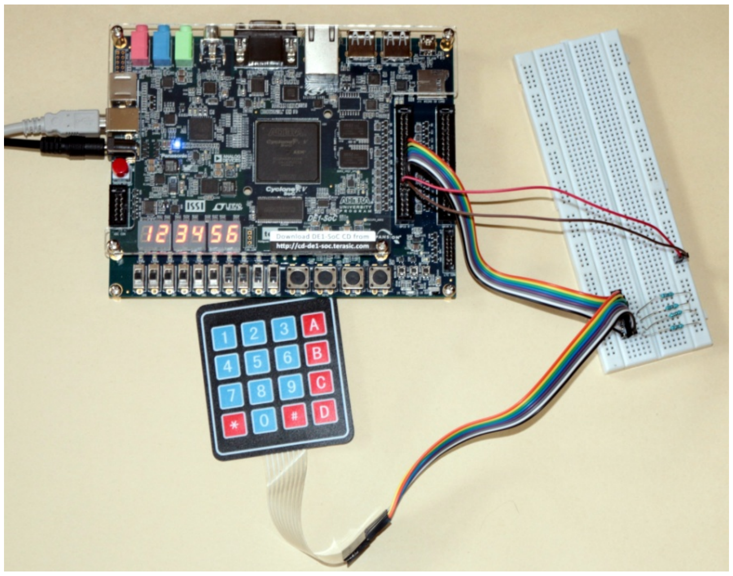
Figure 1. Keypad scanner.

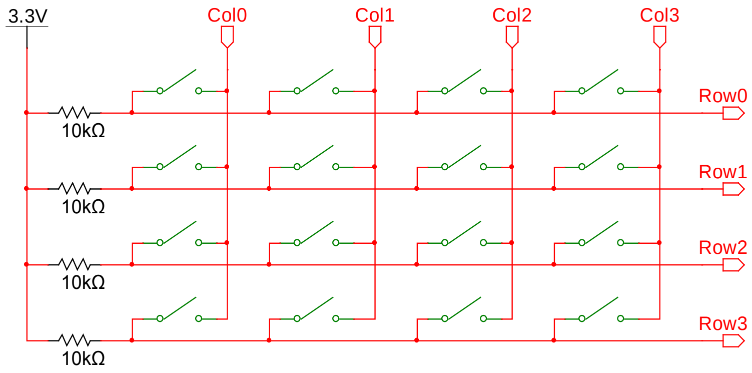

Figure 2. Keypad with pull-up resistors. Columns are the inputs and the Rows

2 Work product

At the end of this lab, you must demo your design and submit your code as a .v file. That is all you need to submit. You will not be writing reports.

3 Switch debounce

When a mechanical switch is closed, the contacts will bounce like a ball for perhaps a millisecond or so, rapidly opening and closing. This happens much too fast for any human to notice, but to digital logic, it looks like the key is being pressed multiple times.

Without additional logic to *debounce* the signal, pressing a key once can generate not just one character, it can generate a whole slew of them racing across the display.

The standard solution is a circuit that exhibits *hysteresis*, meaning that its behavior is sticky. As illustrated conceptually in figure 3, once it in a given state, it's hard to get it out.

Hysteresis can be accomplished by integrating the input over time either digitally or with an analog circuit (e.g., with a device called a Schmitt trigger), switching the output state only after the integral has crossed an appropriate threshold.

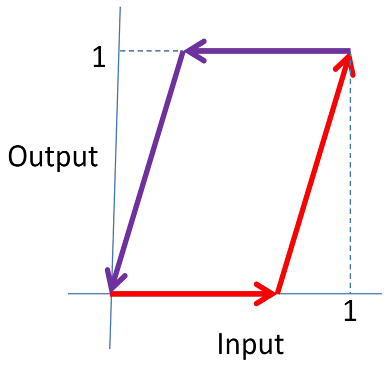

Figure 3. Hysteresis.

This is a class about digital design so course we will do it digitally.

4 Procedure

In this lab, you'll add logic to your keypad scanner to debounce the output to ensure that if you press a key once, you get exactly one key depression.

Here are the design steps you'll follow.

1. Create a new module to debounce the output of your scan module.
2. Add additional logic to shift each a new keystroke into the rightmost 7-segment display.
3. Debug your final design, demo and submit your .v file.

4.1 Stage 2: Debounce each key

Create a module with the following inputs and outputs to debounce the raw output of your scanner. When the input changes, it should wait for it settle before changing its output.

```
module Debounce( input CLOCK_50, input [ 3:0 ] rawKey, input rawValid,
      output reg [ 3:0 ] debouncedKey, output reg debouncedValid );
```

The way to decide if the input has settled is to use a counter to count down until enough time has elapsed and the signal has been steady enough that you trust it. You will also need some reg variables to remember the last state of the raw input.

How long you wait for settling is not very critical. Any reasonable value from about 1 ms = 5,000 clocks to about 10 ms = 50,000 clocks seemed to work fine for me. The shorter the settle time, the more responsive the keyboard but anything in the few millisecond range is fast enough to seem instantaneous to humans.

Create an always block entered on the positive edge of the clock.

1. Each time you see the same valid raw key as last time, decrement the counter.

2. Each time the input isn't valid, increment the counter.
3. If you see a *different* key, reset the counter to the maximum, representing the desired settle time, and set the debounced output as invalid.
4. If the counter hits 0, it means the key has settled. Stop decrementing, set the debounced output to the new value and indicate that it's valid.
5. If the counter hits the maximum, set the debounced output as invalid (meaning no key is being pressed) and stop incrementing.
6. Verify that it you connect the output to a 7-segment display, you get what you expect.

4.2 Display the result

In your main routine, connect it all together.

1. Create an array of six 4-bit reg variables to hold the six hex digits you can display. Create a 6-bit reg variable to indicate whether there's anything in each digit and wire this up to your 7-segment decoders.
2. You will also need a 1-bit reg variable to remember the state of debouncedValid from one clock cycle to the next.
3. Instantiate a copy of your Scan module and wire it to the keypad. Instantiate a copy of your Debounce module and wire it to the output of Scan.
4. Create an always block that's entered on the positive edge of the clock.
5. If debouncedValid = 1 on this clock and it was 0 on the last clock, you have a new keystroke. Shift all the current digits left and insert the new one on the right.
6. Add a reset function to your always block, with reset tied to the leftmost button on the DE1-SoC board.
7. Demo your design and turn in your code.