# CS152A Lab 2 Workshop 2

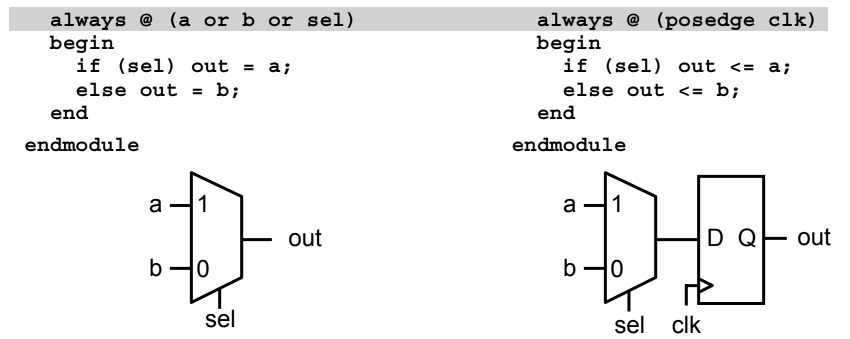
Workshop 2 includes questions related with the sequencer project. For questions, you should append your answers to the end of the regular report. The questions/tasks are listed in the order they appear in the original code.

Note: you’ll have to simulate with the original source code to answer the questions.

## [Implementation] Clock Dividers

In the nexys3.v file, there is a signal/reg named clk\_en. The clk\_en represents a clock that is much slower than the master clock clk. Read the section of code that’s relevant to clk\_en and try to understand how the clock divider is implemented.

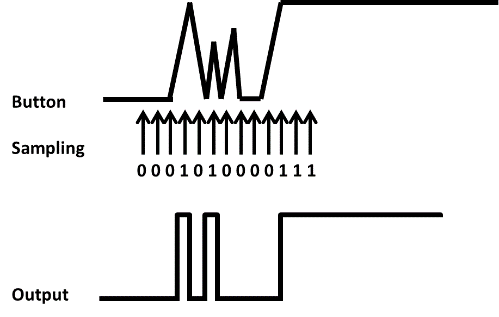
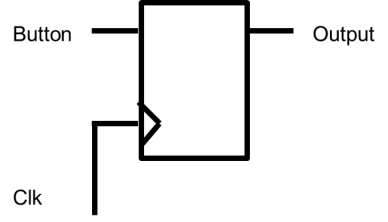
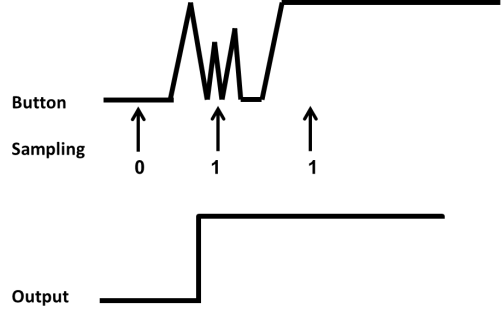
1. Add clk\_en to the simulation waveform tab and then run the simulation again. Use the cursor to find the periodicity of this signal (you can select the signal and use arrow keys to reach the exact edges). Capture a waveform picture that shows two occurrences of clk\_en, and include it in the lab report. Indicate the *exact period* of the signal in the report.
2. A duty cycle is the percentage of one period in which a signal or system is active: , where D is the duty cycle, T is the interval where the signal is high, and p is the period. What is the *exact duty cycle* of clk\_en signal?
3. What is the value of clk\_dv signal during the clock cycle that clk\_en is high?
4. Draw a simple schematic/diagram of signals clk\_dv, clk\_en, and clk\_en\_d signals. It should be a translation of the corresponding Verilog code, such as the following. NOTE: for other lab reports you don’t have to draw diagram with this much detail. A diagram outlining high level module interactions is sufficient.



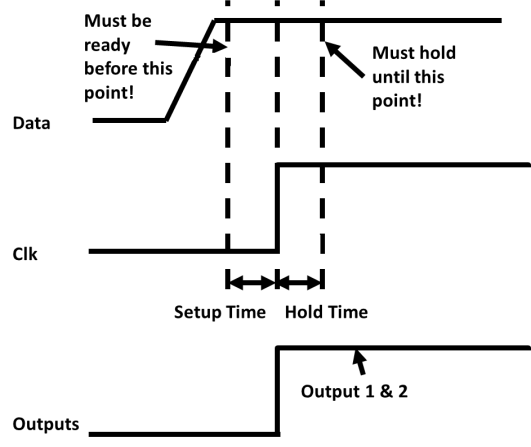
## [Implementation] Debouncing

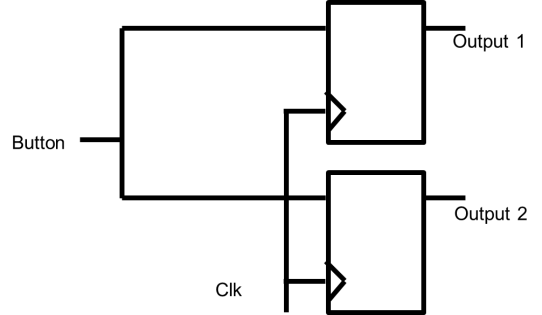
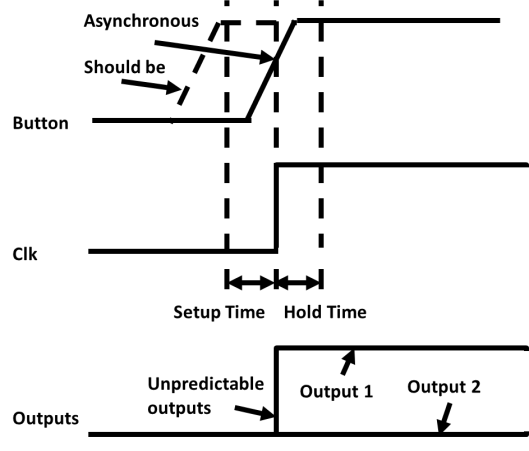
In this project we input instructions using buttons. Due to physical contact instability, the buttons are very bouncy: it means that when you press the button once, it generates an oscillation of signals due to poor metal contact, and thus causing considerable **noise** in the input. A simple solution to filter out the noise is sample at a frequency lower than that of the noise to. In that case, the glitches, which may otherwise be considered by the system as a valid button push, will be filtered out (see the figure below).

In the figure we are showing a flip flop that samples “Button” at every positive edge of the clk and outputs “Output”. If we sample at a very fast frequency (using a high frequency clk), we may get a sequence such as “0001010000111”, where the extra 0s and 1s caused by the noise. We can simply solve the problem by sampling at a lower frequency

Another problem, **metastabilty**, exists because of the asynchronous nature of the button and slider switch input. To ensure reliable operation in digital circuits, the input to a register must be stable for a minimum time before the clock edge (register setup time or tSU) and for a minimum time after the clock edge (register hold time or tH), which is the signal timing requirements in its simplest form. In synchronous systems, the input signals must always meet the register timing requirements. Yet in the case of the buttons and slider switches, the signal may come at any time, causing unpredictable (and possibly different) outputs to the other modules connected to the signal. A simple solution to the problem is to use a flip-flop after the asynchronous input, and regard the output of the flip-flop as the input. With that, the outputs to all other modules will be consistent.





In this project, the author combined debouncing with edge detection to make sure that every button push is counted precisely as once. That one valid push is represented as the signal inst\_vld . Read the relevant code and use the simulation as your aid, answer the following questions in your lab report.

1. What would be the impact if we use clk\_en instead of clk\_en\_d in the statement at line 102 of nexys3.v?
2. Instead of clk\_en <= clk\_dv\_inc[17] (line 76), can we instead have clk\_en <= clk\_dv[16] (which effectively creates a 50% duty cycle clk\_en)? Why?
3. Include waveform captures that clearly show the timing relationship between clk\_en, step\_d[1], step\_d[0], btnS, clk\_en\_d, and inst\_vld.
4. Draw a simple schematic/diagram of the signals above. It should be a translation of the corresponding Verilog code.

## [Implementation] Register File

The sequencer’s register file is located in a file called seq\_rf.v. It stores the values of the four registers. Take a look at the source code and see if you can understand how it is implemented. Answer the following questions in the lab report.

1. Find the line of code where a register is written a non-zero value. Is this sequential logic or combinatorial logic?
2. Find the lines of code where the register values are read out from the register file. Is this sequential or combinatorial logic? If you were to manually implement the readout logic, what kind of logic elements would you use?
3. Capture a waveform that shows the first time register 0 is written with a non-zero value.
4. Draw a circuit diagram of the register file block. It should be a translation of the corresponding Verilog code.