Lab 3 – Sequential Logic

Introduction

The Lab 3 project will extend the functionality of Lab 2 project, by adding:

- An input register to capture or "freeze" the binary value from the slide switch inputs;
- Synchronize the asynchronous slide switch inputs, by using the two flip flop synchronizer circuit;
- Debounce the pushbuttons (all that are used in your design), using the provided debouncer code and its testbench (debounce.sv and debounce_tb.sv). The debounce circuit diagram is shown below.

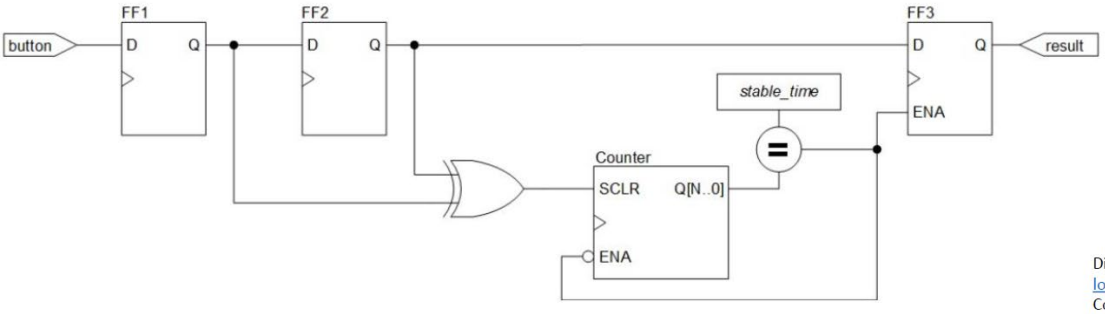

Diagram from https://forum.digikey.com/t/debounce-logic-circuit-vhdl/12573
Code written by instructor based on diagram.

As you progress your design project, be sure to use your top level module only to instantiate and connect lower level modules. Do not put logic designs, e.g. your multiplexer code, into your top level module. Rather, make a separate multiplexer module and instantiate and connect it within your top level module. This helps in a several ways:

- It facilitates hierarchical design, so that you can perform unit testing on individual modules to verify their functionality, before incorporating them into your design project.
- It facilitates reuse of your multiplexer module, where you can just plunk your module somewhere in another part of the design or in another design project.
- It keeps your top level clean, so that you can see your design as a block diagram and understand the signal flows between modules, through their connections.

In addition, consider when it's appropriate to combine related modules into a "mini top level," to create a higher level of abstraction and simplify your design. This was demonstrated by the 7-segment display subsystem, where three lower level modules were connected structurally to form the 7-segment display subsystem. The 7-segment display subsystem didn't make the design any more efficient from a hardware perspective, but it facilitated understanding the system for the designer by reducing clutter and unnecessary complexity from the lower level modules.

We will also follow synchronous design principles, so whenever we have a `posedge` in an `always` block, it shall be `posedge clk`, and not `posedge someOtherSignal`. Only one clock for the entire system. Use the concept of an enable signal, to avoid driving other signals as clock signals.

You will submit your Design Record to the D2L Dropbox prior to your TA evaluation.

Procedure

1. Prior to the lab, watch the all the lecture videos for this week and the previous week.
2. We will first recreate a clean copy of Lab 2, so that we have a reference to go back to, if the iteration for Lab 3 fails. Make a new directory for Lab 3 and copy and paste all your Lab 2 *.SV* and *.XDC* files, into the new Lab 3 directory. Rename the file and the module names from Lab 2 to Lab 3. Run the Implementation and download the new bitstream to verify that you have successfully created a working copy of your Lab 2.
    - **DO: copy and paste a snip of your top level RTL Schematic, into your Design Record.**
3. Add the *debounce.sv* and *debounce_tb.sv* files to your project. Set these files as "top" (recall, the three little dots). Read the debounce files and set their parameters correctly for our design specification: we have a **100 MHz** clock on the Basys3 board, and we will assume a pushbutton bounce time of **50 ms**. Run a simulation and verify that the clock period and the stable time are correct. Use the simulation "markers" to measure the time, i.e. use the "add marker" button in the simulator to add an additional marker and the time difference between the two markers will be shown in the simulation window.
    - **DO: Write a statement in your Design Record what parameter values you changed, i.e. what are the *new values* and in *which file or files* were they changed and explain *why*. Comment on the relationship between the *bounce time* and the *stable time*.**
    - **DO: Zoom in to the simulation waveform and use the markers to measure the clock period, paste in snip from the waveform and markers' time reading in your Design Record, to show the correct clock period.**
    - **DO: Use the markers to measure the stable time in the simulation waveform and paste a snip in your Design Record to show the stable time.**
    - Optional: Take a look at the RTL Schematic of the debouncer and notice that the above image is much more readable and understandable, compared to the RTL Schematic.

4. We'll get back to the debouncer later. We are going to add a register to hold a binary value from the switches inputs. Be sure to set your "top" back to your top level module.
5. Design a register module to capture the switches inputs. The purpose of the register is to store the binary value of the current switches inputs, like the memory function of your calculator. Then, under user control, this stored value will be presented to the 7-segment displays. But, first we will design the register and test it with a simple testbench. The register's basic specifications are:
    - 16-bit inputs and outputs
    - Enabled by a pushbutton (you pick which one). So, when the button is pushed, the current value of the switches inputs is stored in the register (FYI, as a hint, this is a clock enable). When the button is not pushed, the register ignores the switches inputs (as you would expect).
    - Clocked by the 100 MHz clock, i.e. use `posedge clk`.
    - Active high synchronous reset, i.e. use the `reset` signal.

    Write the design and testbench code for the register, and test it in simulation.

    DO: copy and paste a snip of the simulation waveforms that show the register working.

6. Incorporate the register into the top level, and ensure that the pushbutton is correctly edited in the .XDC constraints file. Make changes to the multiplexer and the rest of your design so that either of 4 values are presented to the 7-segment displays (in any order):
    - BCD value of the current switches position
    - Hexadecimal value of the current switches position
    - BCD value of the stored register value
    - Hexadecimal value of the stored register value

    You should probably refer to the RTL Schematic of your Lab 2, to help visualize where to place the components and where to make changes, as this is not trivial. You'll also need to think about how to be able to switch between the four values, how will you control that, do you need additional signals for the user? Test your design on the Basys3.

    DO: when completed, copy and paste a snip of your top level RTL Schematic, into your Design Record.

7. Create a synchronizer module for the slide switches. This is basically two 16-bit registers in series, and they won't be enabled like the storage register you have already designed. Connect this synchronizer module to the switches inputs in your top level, to ensure that the signals are safe for the rest of the system, i.e. synchronized.
8. In your top level, connect the debouncer module to the pushbuttons, with one debouncer for each pushbutton that you are using, except `reset`. Test your design on the Basys3 and verify the correct functionality.

    DO: when completed, copy and paste a snip of your top level RTL Schematic, into your Design Record.

    DO: take a snip of the Timing window and paste it into your Design Record. Show the calculations for the maximum clock frequency for your design, based on the WNS.

    DO: show your working design to your TA.

Optional: Modify your top level testbench testbench to demonstrate the functionality of your design.

Optional: Create test code to count the transitions (or "bounces") on the slide switches or pushbuttons. You would create an "edge-detector" for the signal changes, which would generate a one-clock cycle pulse for every positive or negative going edge. Then these pulses from the edge detector would go to the enable input signal of a counter, and the counter value would end up going to the 7-segment displays. Reset will clear the counter, so that you can test another input signal. You may have bouncing on zero-to-one or one-to-zero transitions, or none at all. This depends on the Basys3 board, its circuitry, and its pushbutton/slide switches.

Deliverables

By the end of the lab period **or** the beginning of the next lab period, demonstrate to the TA:

1. Your Basys3 board running with the latest iteration of the Lab 3 project, be prepared to explain the design and any part of the Vivado design and programming flow.
2. Optional: Your latest simulation and be prepared to explain the signals and their meaning.
3. Your Design Record document. Ensure that the Design Record is uploaded to the D2L Dropbox, before seeing the TA.

Your TA may ask any team members at random to answer any questions. Work together to ensure that all team members fully understand the lab project and its deliverables. You may also be asked to demonstrate your ability to proceed through the entire design flow, including:

1. Create and start a new project.
2. Synthesize and download your design to the Basys3.
3. Simulate your design and setup the waveforms.

Rubric

As the term progresses, the expectations will increase as your skills develop. All team members are required to participate in the entire lab period and to present their project to the TA. Missing team members will not receive a mark for the project. The three strikes policy outlined in the Course Outline, will be in effect for all labs.

Points	Criteria
4	Fully complete and high quality, questions answered well.
3	Fully complete and high quality, some questions not answered well or had to have other team members answer.
2	Mostly complete or answers are weakly answered by team members
0	Below acceptable for credit.

There is no 1 point given. Fractional points, e.g. 2.5, are not given.