Simulation

Introduction

In this lab you will simulate your SPI interface design using the ModelSim HDL simulator. You may want to do both labs at the same time since the simulations may help you complete the SPI interface lab.

Simulation

Although simulations compile faster and allow you to view the behaviour of your design, there are some important differences between simulation and hardware synthesis.

Register Initialization

HDL simulators assume that uninitialized variables have the value 'x' (undefined). This helps detect uninitialized values that could cause unpredictable behaviour. However, FPGA programming files generated by Quartus initialize register values. This means your simulations will not match the results obtained with FPGA hardware unless your HDL code initializes *all* registers at the start of the simulation.

The simplest way to do this is to initialize registers as part of their declaration. For example:

```
logic transmit = '0, transmit_next ;
...
logic [10:0] count = 11'b111_1111_1111, count_next ;
...
logic miso = '0, miso_next ;
...
logic [15:0] d = '0, d_next ;
```

and any other registers in your design.

Note that for this simulation the controller should be initialized to an inactive state. This means the count variable should be initialized to all 1's (`11'b111_1111_1111`).

The `x` register in the multiplexed display code should also be initialized to prevent the digits from being displayed as undefined:

```
   logic [15:0] x = '0 ;
```

Time Scale

Since simulations run much slower than real time it's not practical to simulate certain things. For example,the debouncer module waits for signals to be stable for 64k clock cycles. Since we don't need to test the debouncing circuit this is a waste of simulation time.

A version of the debouncer with zero delay, **`tb_debounce`**, is available on the course web site.

Procedure

You will use the "Modelsim - Intel FPGA Starter Edition" which can be downloaded for free from the same web site as Quartus and is likely already installed.

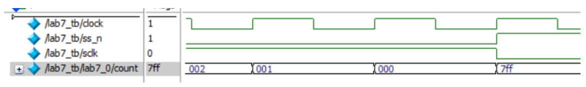
Download the **`lab7_tb.sv`** testbench and **`tb_debounce.sv`** from the course web site to your Lab 7 project folder.

Follow the procedure in Appendix A to create a simulation project, add the **`lab7.sv lab7_tb.sv`** and **`tb_debounce.sv`** files to the project and compile them. After fixing any errors, run the simulation. The Transcript window should show any messages generated by the testbench and the Wave window should show the signal waveforms.

If your design is not working properly, you will want to use the simulation output to figure out why. You can examine signals inside your lab7 module (e.g. the counter or the shift register) or in the testbench (e.g. the clock, control and interface signals).

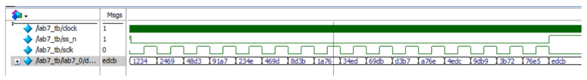
For example:

- a few clock cycles before and after the counter "wraps around" and slave-select goes inactive:



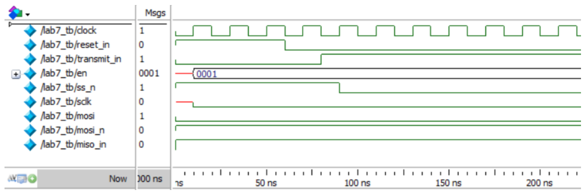
```
/lab7_tb/clock             1
/lab7_tb/ss_n              1
/lab7_tb/sclk              0
/lab7_tb/lab7_0/count      7ff    002  001  000  7ff
```

- the clock slave-select and data register values over complete simulation as seen from the lab7 module:

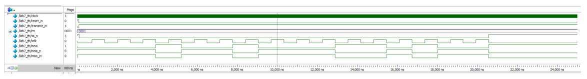


```
                 Msgs
/lab7_tb/clock    1
/lab7_tb/ss_n     1
/lab7_tb/sclk     0
/lab7_tb/lab7_0/d...  edcb   1234 2469 48d3 91a7 234e 469d 8d3b 1a76 34ed 69db d3b7 a76e 4edc 9db9 3b72 76e5 edcb
```

- the initial values of the signals in the testbench:



- the full simulation duration as seen from the testbench:



Hints

- Make sure you don't have any undefined (x, red) values in your simulation output.
- Work on part of your design at a time. For example, make sure the controller counter is decrementing properly and generating `sclk` and `ss_n` as you expect.
- Beware of the Modelsim editor. It has some unusual quirks such as "undo" undoing *all* of the changes since the last version of the file. You may be better off using Notepad++ or the Quartus editor.

Report

Submit a PDF file to the Assignment folder for this lab containing the required identification information and the screen captures listed below.

1. the first 200ns of the simulation as seen from the `lab7_tb` showing the module inputs and outputs (you may omit the LED outputs)
2. the complete simulation duration (25 μs) showing the values of the counter and data (shift) register in your `lab7` module. Display these values in hexadecimal (right-click on a waveform trace and select Radix > Hexadecimal)

Appendix A - Simulation with Modelsim

A simplified procedure is as follows:

1. start the program (on Windows select Intel FPGA ... > Modelsim ...)
2. if this is the first time you simulate these files, select File > New > Project..., select the folder where your files are located as the Project Location and enter a suitable Project Name (e.g. `lab8`), then click OK,
3. select Add Existing File and select the file(s) that contain the entities you want to simulate, including your design and the testbench, then select Close.
4. otherwise, if you had already created a simulation project and it's not already open, select File > Open, select Files of type: Project Files (*.mpf) and select the project file,
5. select Compile > Compile All to compile all the files in your project into the `work` library,
6. if there are syntax errors you will need to fix the error(s), save the file and go back to step 5,
7. otherwise select Simulate > Start Simulation; select your testbench module from the `work` library and select OK,
8. drag the signals you wish to view from the 'Sim' or 'Object' windows to the 'Wave' window (use the View menu to open windows)
9. select Simulate > Run -All; this will run the simulation until it's complete
10. the Transcript window will contain output from the testbench
11. the Wave window will show the waveforms (select the Wave window, click on '+' and Wave > Zoom > Full); you can use a screen capture utility (e.g. Windows Snip tool) to save the waveforms
12. if the results are not as expected, correct the errors, run Compile > Compile All, Simulate > Restart..., click OK and Simulate > Run -All