ECE275: Sequential Logic Circuits
Lab 8: Simulation

Pascal Francis-Mezger

November 14, 2021

Contents

0 Lab Overview

The goal of this lab is to utilize the ModelSim simulator to simulate your Verilog code without the use of the FPGA. This is an incredibly important part of FPGA design, as it can be very difficult to diagnose issues with your code when running on the FPGA. The simulator allows you to see specific states of logic with very tight timing constraints on inputs and outputs.

1 Part 1: Analyzing a Verilog Program Utilizing the ModelSim Software

1.1 Installing the ModelSim software

You will need to use the Altera Modelsim software for this lab. You should first check if you already have it installed from the initial installation of Quartus at the beginning of the semester. You can check in the Windows "Add or remove programs" section if Modelsim is installed. If it is not, you can download it from `https://fpgasoftware.intel.com/13.0sp1/?edition=web`. After the download is finished, install the software. Pay attention to the installation directory question, as you will need this later. I left it as the default `C:\altera\13.0sp1` and will use that for an example later in this writeup. If you modify it, keep that in mind for specifying the Modelsim location later.

1.2 Create a Simple Verilog Program

Create a new Quartus project, and from your top level utilize the Verilog module shown below. This should be the same code from the end of lab last week. Instantiate the module with SW[0] as your clock, SW[1] as R, SW[2] as S, and LED[0] as Q. Test the code on your FPGA board to make sure it is working before moving on to the next step. It should set the Q output on the rising edge of the clock pulse if S is set, and reset the Q output on the rising edge of a clock pulse if R is set.

```verilog
module RS_Latch(
        input Clk,
        input R,
        input S,
        output Q
);
        wire R_g, S_g, Qa, Qb /* synthesis keep */;
        assign R_g = R & Clk;
        assign S_g = S & Clk;
        assign Qa = ~(R_g | Qb);
        assign Qb = ~(S_g | Qa);
        assign Q = Qa;
endmodule
```

1.3 Create a Testbench

A testbench is essentially a Verilog module that sets timing for simulating the Verilog code. Normally you would create the testbench as a seperate Verilog file (as we learned last week how to include external Verilog files/modules in your code), but in this case feel free to keep it in the same file as your top level for easy troubleshooting. If you are interested in learning more about testbenches outside this lab, or are trying to troubleshoot this lab, you can read more about Modelsim testbenches in this Intel document: `https://ftp.intel.com/Public/Pub/fpgaup/pub/Teaching_Materials/current/Tutorials/Verilog/Using_ModelSim.pdf`.

```
‘timescale 1ns / 1ps
module testbench ( );
        reg Clk;
        reg R;
        reg S;
        wire Q;
        RS_Latch rs1 (Clk,R,S,Q);
        initial
        begin
                Clk <= 0; R <= 0; S <= 0;
                #20 Clk <= 0; R <= 0; S <= 1;
                #20 Clk <= 1; R <= 0; S <= 1;
                #20 Clk <= 0; R <= 0; S <= 0;
                #20 Clk <= 0; R <= 1; S <= 0;
                #20 Clk <= 1; R <= 1; S <= 0;
                #20 Clk <= 0; R <= 0; S <= 0;
        end
endmodule
```

I have written the testbench for you to use in the Verilog code shown above. The first line "‘timescale 1ns / 1ps" sets the time deltas for changing input values at 1ns, and simulation precision at 1ps. Setting a lower simulation precision can give higher accuracy, but will increase simulation time. The inputs for simulation are then defined as registers, and outputs as wires. Next, the modules to be simulated are instantiated. In this case that is the RS_Latch module. The next section is contained by the initial, begin, and end lines. The code inside sets the initial values for the inputs, and then times for them to be modified, based on the the timescale value. In the code shown in the example, the Clk, R, and S inputs are all 0. 20ns later, the set bit is modified to a 1. 20ns after that, the Clk bit is also modified to a 1. This continues through a reset cycle as well. This means that when the simulation is run, the Q output should turn on after 40ns, and then turn back off 60ns later.

1.4 Set Path to ModelSim

To utilize your testbench you will first need to tell Quartus what simulator to use, and its location. In Quartus, go to the tools menu, and then towards the bottom, select options. In the window that appears, in the top left under general, select the option for EDA Tool Options. Here you will find ModelSim-Altera towards the bottom. Click the three dot icon to the right of the ModelSim-Altera, and then browse to your ModelSim installation directory. You will need to go deeper into the directory for the path it is looking for. In my case that was `C:\altera\13.0sp1\modelsim_ase\win32aloem`. You will have to modify this path if you selected a different installation directory for your ModelSim installation.

1.5 Run Simulation

After you have selected your Modelsim installation, you can move on to the actual simulation. In Quartus go into the tools menu, go to the Run Simulation Tool section, and choose the RTL Simulation option. If you get an error, then you likely have not installed ModelSim or have not set your path correctly to your ModelSim installation.

1.6 Interact with the Simulation

The last step should have opened a separate window, ModelSim Altera. By default, none of the waves you will want to view will be selected, and your testbench will likely not be selected. To select your testbench, you will need to expand the work option under the libraries on the left. As long as you have run compilation on your top level, the testbench should show under the work section. Double click on it to set it as active. Now under objects and processes you should see the inputs and outputs from the testbench. Right click on each of Clk, S, R, and Q and choose the option to add wave. You should now see a wave graph on the right, but it will not contain data until you actually run the simulation.

The option to run time steps of the simulation are at the top of ModelSim program, next to a box that says ”100ps”. This is the time step value the ModelSim simulation will simulate each time you hit the run option. This is significantly shorter than the amount of time before the first input will change state (100ps vs 20ns). You can either increase this value and use the run button, or farther to the right there is a run -all option that will run the full simulation time. This should simulate out to 120000ps, which by default will not fit well on your graph. To see all the data easily, right click on the graph and choose the zoom full option. With this you should see the full waveforms that show a full set/reset cycle of the RS_Latch circuit.

1.7 Part 1 Completion

Modify your testbench file to use different time values for the clock, set, and reset. Show your TA how this effects your waveform output in the simulation, and explain to them the changes you made.

2 Part 2

For part 2, create a testbench for one of your previous labs. Test against several input values, and show on the simulation the output matches your expectations. It is possible to loop or create a clock for sequential code in the testbench. View the testbench on page 9 of `https://ftp.intel.com/Public/Pub/fpgaup/pub/Teaching_Materials/current/Tutorials/Verilog/Using_ModelSim.pdf` for an example of using more advanced logic for testing in your testbench.