ECE 491

Laboratory 3 – Simulation and Testbenches

September 17, 2007

Goals

- To gain experience using the Verilog Simulator (Modelsim).
- To gain experience using testbenches to verify Verilog modules.

Requirements

You will develop a self-checking testbenches for the following modules from Lab 2:

1. The Generic Counter. Your testbench should create multiple instances of the counter with different parameters and verify that each instance operates properly using exhaustive simulation.
2. The Clock Divider. It should verify that your clock divider correctly divides the input clock period by a factor of 500,000, and should also verify that the duty cycle of the clock is 50%. HINT: Use for loops combined with `@(posedge clk)` statements to delay multiple clock cycles before testing the output.

All Verilog code should follow the Coding Guidelines discussed in class last week.

Deliverables

1. Demonstration to Lab Instructor of successfully operating testbenches.
2. A short technical memorandum which describes (a) what was done, (b) what you learned, and (c) what difficulties you encountered.
3. Verilog listings of all files, including generic counter, generic counter testbench, clock divider, and clock divider testbench.Timing diagram printouts showing correct simulation of your counter testbench.
4. Transcript printout showing correct operation of your clock divider testbench.
5. Entries in your Lab Notebook documenting design ideas, the steps taken to create and debug your design, any pitfalls you encountered, and recorded data (if any). Each Lab Notebook entry should be dated and signed by all students in the lab group.

Background

In Labs 1 and 2, we compiled Verilog code directly into hardware and tested our designs in hardware. This approach works well for small designs, but becomes limited as designs become more complex designs. These designs are typically *verified* using a Verilog simulator. The Xilinx tools are designed to work with the Modelsim simulator, which is a product of Mentor graphics. To properly use a Verilog simulator, code must be added to the synthesizeable modules to generate input *stimulus* and check the output *response* of each module. This code is usually called a *testbench* or *test fixture*.

Testbenches are written using *behavioral code* – code that only works during simulation and that cannot be compiled into hardware. A typical testbench contains a Verilog "`initial`" block that contains a sequence of assignment statements combined with delay operators to apply a set of input vectors to the *device under verification (DUV)*. *Self-checking* testbenches include code that tests the outputs of the DUV to check that they are correct. If not, they typically print error messages.

An example testbench was presented in class that verifies the correct operation of a BCD counter. In this lab, you will extend this testbench to verify the parameterized counter you designed in Lab 2. You will also use this testbench as a starting point for a testbench that verifies the operation of the Serial Transmission circuit designed in the next lab.

In the Lab

1. To gain experience using Modelsim, create a new project and add the BCD counter file "bcdcounter.v" from the class website (specify "Synthesis/Imp. + Simulation" in the "Adding Source Files…" popup menu). Next, add the testbench file "bcdcounter_bench.v". Be sure to specify that this file is a "Simulation Only" in the "Adding Source Files…" popup menu.
2. Click on the testbench file in the upper-left panel of the Project Navigator. Next, in the lower-left panel, double-click "Simulate Behavioral Model". Modelsim will start in a separate window and the simulation will run. Note that the timing diagram window shows waveforms of the signals in the testbench file, while the transcript window shows the
3. The testbench for the BCD counter contains an intentional error where it checks for the wrong value. Correct this error and re-simulate the circuit to show that it now simulates without error messages. NOTE: You must quit Modelsim if you want to edit the testbench in ISE and re-simulate.
4. Create a new project and add your parameterized counter and the testbench that you wrote to test the parameterized counter. Simulate your counter and testbench and make sure that it correctly verifies the operation of your counter. Print the timing diagram that results and include it with your report.
5. Create a new project and add your clock divider circuit and testbench. Simulate the circuit and make sure that it correctly verifies the operation of your circuit. Print the transcript window showing that your simulation completed correctly.