CS223 Laboratory Assignment 3

Modeling Decoders and Multiplexers in SystemVerilog

Lab dates and times:

Section 1:	30.10.2017 Monday 08:40-12:25
Section 2:	31.10.2017 Tuesday 08:40-12:25
Section 3:	30.10.2017 Monday 13:40-17:25
Section 4:	31.10.2017 Tuesday 13:40-17:25
Section 5:	26.10.2017 Thursday 08:40-12:25
Section 6:	27.10.2017 Friday 08:40-12:25

Location: EA Z04 (in the EA building, straight ahead past the elevators)
Groups: Each student will do the lab individually. Group size = 1

Preliminary Report (30points)

Today's lab needs considerable advance preparation. These advance designs and SystemVerilog models should be prepared in advance, and assembled neatly into a Preliminary Report with a printed cover page and printed pages for the schematics and SystemVerilog codes. Each page should have a proper heading. The contents of the report should be as follows:

a) A cover page which includes the following: course name and code number, the number of the lab, your name and student ID, date, <u>number of your trainer pack (remember lab policies. You must always use same pack number).</u>
b) Behavioral (dataflow) SystemVerilog module for Part A-a. Prepare a testbench for it.
c) Schematic and structural SystemVerilog module for Part A-e. Prepare a testbench for it.
d) Schematic and SystemVerilog module for Part B.
e) Behavioral SystemVerilog module for Part C-a.
f) Schematic, structural SystemVerilog module and testbench for Part C-b.
g) Schematic and SystemVerilog module for Part C-d.

The Preliminary Report will be turned in at the <u>start</u> of lab. You may need a copy of your designs and SystemVerilog programs with you in the lab to refer to or possibly correct and change it. In this lab, you don’t need to connect your BASYS-3 board to the Beti board. Working with standalone BASYS-3 board and having it connected to your computer is enough. Create a new Xilinx Vivado Project to do each part. Use appropriate names for files and folders.

Part A: Decoders (30 points)

Decoders are widely used in digital design, as a building block. Although they themselves can be built with logic gates, but their function is often described (and modeled in SystemVerilog) rather than their structure. As you will see, decoders can be composed into larger decoders.

A 2-to-4 decoder decodes a 2-bit input binary number by setting exactly one of the decoder's 4 outputs to 1. Unless it has an enable signal, one and only one output of a decoder will ever be 1 at the same time, corresponding to the current value of the inputs. With an enable signal, it is possible to make all the outputs be 0, when the decoder is disabled. When enabled, it behaves as described above. Decoder outputs are mutually exclusive, and in fact are the minterms of the inputs.

a) Write code: Give the SystemVerilog code which models a 2-to-4 decoder with enable, in the behavioral style. (This means modeling with Boolean equations, using continuous assignment statements.) Name your SystemVerilog module *D2to4_decoder*. In the port list, let the outputs be first (from most significant to least significant), then the inputs (from most significant to least significant), then the enable signal.
b) Simulate it: Using the SystemVerilog testbench code, verify in simulation that your 2-to-4 decoder with enable is working correctly. (Be sure to compare the order of the ports in your module with the order of the ports in the instantiation of *D2to4_decoder* in the testbench, to make sure they match 1-to-1.) Note that the testing is complete, using all possible input combinations. When you are convinced that it works correctly, show the simulation results to the TA. Be prepared to answer questions that you may be asked.
c) Make FPGA project: Now, follow the Xilinx design flow to synthesize, create programming file, and download your 2-to-4 decoder to your BASYS-3 FPGA board.
d) Test it: Using the switches and LEDs on BASYS-3, test your decoder. When you are convinced that it works correctly, show the physical implementation results to the TA. Be prepared to answer questions that you may be asked.
e) Write code: Design and give the structural SystemVerilog code for a 3-to-8 decoder using only two 2-to-4 decoders with enable, one inverter, and connecting wires. Show the block diagram of this 3-to-8 decoder (using block symbols for the decoders) with the 3 inputs and 8 outputs of the block, and the correct connections of the decoders, inverter and wires inside the block. Use ONLY these components.
f) Simulate it: Do part b, this time for 3-to-8 decoder. Finally show it to your TA.
g) Make FPGA project: Do part c, this time for 3-to-8 decoder.
h) Test it: Do part d, this time for 3-to-8 decoder. Finally show it to your TA.

Part B: Implementing Boolean functions using decoders (10 points).

Given the function F(w,x,y)=m(0,1,5,6), implement it using your 3-to-8 decoder and OR gate in SystemVerilog. Connect function inputs to switches and function output (OR output) to one LED on Basys-3. Using the switches and the LED that you have assigned, test your function. When you are convinced that it works correctly, show the physical implementation results to the TA. Be prepared to answer questions that you may be asked.

Part C: Multiplexers and Boolean function implementation (30points)

A multiplexer ("MUX" for short) is another higher-level building block, used widely in digital design. A M-to-1 multiplexer has M data inputs and 1 data output, and allows only one input to pass through to the output. A set of select inputs determines which input to pass through. If a MUX has an enable input, then it is possible to disable the MUX and force a 0 onto the output, regardless of input values. MUXs can be composed into larger MUXs, as you will see in this part of the lab.

a) Write code: Give behavioral SystemVerilog module for 4-to-1 multiplexer with enable.
b) Write code: Write structural SystemVerilog code for an 8-to1 multiplexer by using two 4-to-1 multiplexers with enable. You can use a 2-input OR gate, and an inverter.
c) Simulate it: Using the SystemVerilog testbench code, simulate 8-to-1 multiplexer and show it to your TA.
d) Test it: Give the SystemVerilog module for G(w,x,y,z)=m(1,2,5,7,8,9,10,12,15) function, using one (not two) 8-to-1 multiplexer. You need to use one of four inputs in multiplexer inputs, and the other three inputs are used in multiplexer select pins (refer to Fig 2.60 in textbook). Test your circuit using switches as input and an LED as output. Show your circuit and test to TA. Be prepared to answer questions that you may be asked.

Submit your code for MOSS similarity testing

Finally, when you are done and before leaving the lab, you need to upload the file *StudentID_SVerilog.txt* created in the Implementation with FPGA part. Be sure that the file contains exactly and only the codes which are specifically detailed above. If you have multiple files, just copy and paste them in order, one after another inside text file. Check the specifications! Even if you didn't finish, or didn't get the SystemVerilog part working, you must submit your code to the Unilica Assignment for similarity checking. Your codes will be compared against all the other codes in all sections of the class, by the MOSS program, to determine how similar it is (as an indication of plagiarism). So be sure that the code you submit is code that you actually wrote yourself! All students must upload their code to the 'Unilica>Assignment' specific for your section. Check submission time and don't miss it before leaving the lab. After taking a backup of your work, don't forget to delete it from computer. Because students of other sections will work with your system too.

Clean Up

1) Clean up your lab station, and return all the parts, wires, the Beti trainer board, etc. Leave your lab workstation for others the way you would like to find it.

2) CONGRATULATIONS! You are finished with this Lab and are one step closer to becoming a computer engineer.

NOTES

--Advance work on this lab, and all labs, is strongly suggested.
--Be sure to read and follow the Policies for CS223 labs, posted in Unilica.

LAB POLICIES

1. There are three computers in each row in the lab. Don’t use middle computers, unless you are allowed by lab coordinator.
2. You borrow a lab-board containing the development board, connectors, etc. in the beginning. The lab coordinator takes your signature. When you are done, return it to his/her, otherwise you will be responsible and lose points.
3. Each lab-board has a number. You must always use the same board throughout the semester.
4. You must be in the lab, working on the lab, from the time lab starts until you finish and leave. (bathroom and snack breaks are the exception to this rule). Absence from the lab, at any time, is counted as absence from the whole lab that day.
5. No cell phone usage during lab. Tell friends not to call during the lab hours--you are busy learning how digital circuits work !
6. Internet usage is permitted only to lab-related technical sites. No Facebook, Twitter, email, news, video games, etc--you are busy learning how digital circuits work !
7. If you come to lab later than 20 minutes, you will lose that session completely.
8. When you are done, DO NOT return IC parts into the IC boxes where you’ve taken them first. Just put them inside your Lab-board box. Lab coordinator will check and return them later.