ECE 3544: Digital Design I

Project 1 (Part B): Simulation in the ModelSim Environment; Continuous Assignment Models

**Objectives**

This project will improve your familiarity with the ModelSim tools, timing models, and with the use of test benches. Additionally, it will allow you to compare the functionality and coding structure of Verilog models that use primitive gates and those that use continuous assignment.

**Requirements**

You must have the current version of ModelSim-Altera Starter Edition installed on your computer. The instructions in this assignment are consistent with that version of ModelSim. Even if these instructions are consistent with other versions of ModelSim, you should use the version indicated here.

*You will use both ModelSim and Quartus, but you will only carry out the Analysis and Synthesis step in Quartus. The DE1-SoC Board is not required for this project.*

**Project Description**

As it was in Part A, your first step will be to simulate two pre-designed circuits:

* The 74138 3-to-8 decoder, and
* A 4-to-16 decoder that uses two of the 3-to-8 decoders as structural blocks.

This time, however, the basis for describing the 74138 chip is a continuous assignment model using dataflow operators, instead of the primitive gate model you used in Project 1A. *In your lab report, comment on the differences that you perceive in continuous assignment models that use dataflow operators, and structural models that use primitive gates. Your commentary should include discussion on design differences and implementation differences.*

After using test benches to simulate the decoders and verify that the changes in the models do not cause differences in behavior, you will repeat the design and implementation process of Project 1A, except that your target model will be continuous assignment. You should be able to simulate your design and verify that its behavior matches that of the primitive gate circuit you designed in Project 1A.

**Instructions**

*Part 1: Simulate the decoder modules.*

1. Follow the instructions in Project 1A for making a project for the Project 1B Starter Files.
2. Study the example files to make sure that you understand how each model works in Verilog. Start with the “new” sn74138.v. This is a Verilog behavioral model – specifically, one that uses continuous assignment and dataflow operators to describe the same 74138 chip that you saw in Project 1A.

Next, look at decoder4to16.v. This is the Verilog model for a 4-to-16 decoder made from two 3-to-8 decoders. Even though this model does not contain the same explicit inverter that you saw in the model from Project 1A, you should be able to determine the manner in which it performs the same operation.

You should recognize the two test benches as being the same ones that you saw in Project 1A. This should make sense, since you are seeking to simulate the behavior of the same circuits. Review the two test benches to ensure that you still understand how they function as stimulus files.

1. Follow the instructions in Project 1A to compile the files in the project and to simulate the two decoders. Include screenshots of your results showing proper operation of the two decoders. Comment on whether the change in the modeling technique affects the behavior of the two decoders.

*Part 2: Create your own models and test bench.*

Create the magnitude comparator circuit that you designed and implemented in Project 1A, except this time implement it with a *continuous assignment model using dataflow operators*.

module comparator(valA, valB, aGTb, aGEb, aLTb, aLEb, aEQb, aNEb);

input [2:0] valA, valB;

output aGTb, aGEb, aLTb, aLEb, aEQb, aNEb;

The circuit accepts the same two 3-bit inputs representing unsigned binary numbers and provides the same six outputs that represent the various results from comparing the two values, as indicated in Project 1A.

Follow the instructions in Project 1A for using the starter files, naming your module and files accordingly, and creating a new project and add your modules to it. The test benches that you created in Project 1A should suffice for this implementation. Make sure to change the names to match the convention that you follow in this project. You should include waveforms from the test-bench in your report that show the proper operation of your design.

|  |
| --- |
| *This is important!*  Comment on whether the change in the modeling technique affects the behavior of this model, as compared to the one you derived in Project 1A. You should assume that simply translating your structural solution to one using continuous assignment and dataflow operators is not an acceptable way to take advantage of the operators that you have at your disposal and **will incur penalty as a result**. Since one of the goals of using behavioral models is to facilitate productivity gain that don’t usually arise from structural models that use primitive gates, you should approach the modeling of the system accordingly. |

Your modules and test bench should include proper header information, contain reasonable comments, and be neatly formatted. The starter files provide examples of header information; make sure that you fill in the header comments.

*Part 3: A first look at synthesis.*

|  |
| --- |
| *This is important!*  I have made a pin assignment for this top-level module. This will permit you to do a full compile of the top-level module and use it to program your board if you wish. In future assignments that use the DE1-SoC board, **you will have to do the pin assignment on your own**. **Do not program your board with an implementation that does not have a complete pin assignment.** It can damage your board. |

To compare the implementations from Projects 1A and 1B, we have to establish a way for Quartus to analyze, synthesize, and fit them to the FPGA.

1. Open the Quartus archive. In the **Project Navigator** window, open the top-level module by double-clicking on comparator\_top in the **Hierarchy** pulldown tab or on the file having that name on the **Files** pulldown.
2. Follow the instructions in the comments of the top-level module. To add files to a project, choose the **Files** pulldown in the **Project Navigator** window, then right-click on the Files folder in the window and choose **Add/Remove Files in Project**. In the window that appears, choose the … button to navigate to the folder containing your source files, then click **Add** to add them to the project. (You’re likely to have an easier time if you copy the files to the project1b\_restored folder that results when you open the archive.)
3. After adding the source files to the project, choose one of them as the active comparator instance by leaving it uncommented. In the **Tasks** window, double-Click on the **Fitter (Place and Route)** task. This should cause Quartus to perform both the **Analysis & Synthesis** task and the **Fitter (Place and Route)** task.
4. After these steps complete, there should be a **Compilation Report** tab for the top-level module.

* Open the **Analysis & Synthesis** folder. Click on the **Resource Usage Summary** report. Include a screenshot of the contents of the summary where you highlight the **Estimate of Logic utilization** and the **Combinational ALUT usage for logic**. Expand this option and highlight the function usage.
* Open the **Fitter** folder, then open the **Resource Section** sub-folder. Clock on the **Resource Usage Summary** report. Include a screenshot of the contents of the summary where you highlight the **Logic utilization**.

Review these sources for more information on the logic resources that make up the Cyclone V FPGA:

* Section 1 of the [Cyclone V Device Handbook](https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-v/cv_52006.pdf)
* [Adaptive Logic Module](https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/reference/glossary/def_alm.htm)
* [Adaptive Look-up Table](https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/reference/glossary/def_alut.htm)
* [Logic Array Block](https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/reference/glossary/def_lab.htm)

1. Choose **Tools** > **Netlist Viewers** > **RTL Viewer**. The RTL Viewer shows a schematic of the design after Analysis but prior to fitting. Instances in the top-level module are represented by expandable boxes. Click the + sign in the corner of a block to expand it for view. There should only be one module in the design, so expand it and take a screenshot of the RTL schematic.
2. Choose **Tools** > **Netlist Viewers** > **Technology Map Viewer (Post Fitting)**. The Technology Map Viewer shows a schematic of the design after fitting and shows the utilization of low-level resources such as ALMs. Expand the module and take a screen shot of the technology mapping.

*Repeat these Steps 3 to 6 for the other comparator*. *Use the information you gather from the reports and the schematics to address the following:*

* Discuss the features of the RTL schematics that stand out to you.
* Compare the resource usage of the two implementations.
* What conclusions about the approach to modeling can you make from your comparison of the resource usage?

**Project Submission**

Your submission should include the following files. Submit these items on Canvas by the deadline as individual files. *Do not place the files into an archive*.

* A project report. Your report should be in Word or PDF format. Include your PID in your report filename, *e.g.*, project1Breport\_jthweatt.pdf.

Your report should contain the following elements:

* + The cover sheet included in the project materials. **I have included the cover sheet as a Word document, so that you can place it in your report document before you convert it to PDF**.
  + A restatement of the assignment’s objectives.
  + A section that addresses the questions posed in the project specification.
  + A section that describes the process by which you derived the design of your circuit and the manner in which you implemented it in Verilog. **In particular, you should comment on how applicable you believe your design process and your method for implementing it in Verilog are to performing more general and larger-scale designs and implementations. Compare your responses here to the ones you gave in Project 1A for primitive gate models.**
  + A section that shows the simulation of the decoders and the results of your game circuit testing. **Discuss how you formulated the tests contained in the test bench that you had to generate, and how you verified the correctness of your implementation.**
  + A section that contains the reports and schematics from Quartus.
  + A section where you discuss your conclusions and the lessons you learned from the assignment.
* The Verilog source files and testbenches for your comparator circuit. Name your files according to the convention described in the starter files. **Your code should be documented appropriately.**