Lab 4

Introduction to SystemVerilog, FPGA, EDA, and 16-bit Adders

In this experiment you will transition from the TTL physical logic elements to RTL design on FPGA using SystemVerilog. You will understand the basic syntax and constructs of the SystemVerilog, as well as the basic skill required to operate Quartus II, a EDA tool for FPGA synthesis and simulation. You will also learn the relevant performance analysis using Quartus II, as well as the basic optimization procedure in Quartus gearing towards area and power. Finally, you will implement a carry-ripple adder, a carry-lookahead adder, and a carry-select adder in SystemVerilog. You will then analyze their performance in terms of area, power, and the maximum operating frequency.

Assignment

- Download and install Quartus II (15.0) (only available for Windows and Linux) if you decide to work on your own machine. Note that you should install Quartus II Web Edition, Device Support for Cyclone IV, and ModelSim Altera Edition (**includes Starter Edition, please select Starter Edition during the setup**). 15.0 is the recommended version. If you have a Mac, you should install Windows 10 through Boot Camp (available free of charge for engineering students from the University Webstore) or use Parallels Desktop (available for a fee from the Webstore).
- Read the Introduction to SystemVerilog (pdf) (**UPDATED: 09/11/17**) and the Experiment 4 section of the lab manual (pdf) (**UPDATED: 09/25/17**).
- Read and complete the tutorials in Introduction to Quartus II in the lab manual. (**UPDATED: 09/18/17**)
- Check out the Quartus II Tutorial Video (a video version of the Introduction to Quartus II document in the lab manual. A 16-bit ripple adder is implemented step by step in this video.) (**UPDATED: 09/15/17).**
- Check out the Modelsim/Testbenches intro video
- **Complete the 8-bit serial processor exercise in IQT using the provided logic processor files (zip download),** extending the 4-bit logic processor to an 8-bit version, and confirming that the simulation for your demo (test-bench for 8-bit processor is included)
- Read the Lab 4 description in the lab manual and complete the Lab 4 Pre-Lab and design before the lab section.
- **Design the three adders described in the lab manual by modifying the provided code (zip download).**
- Work on Lab 4 report and Lab 5 Pre-Lab after the lab section.
- (Optionally for your reference) Look at the tutorial on using schematic capture in Quartus (not the recommended method in ECE 385)
- (Optionally for your reference) Modelsim quick reference for .do files for generating the waveforms in ModelSim. The do files enable you to add the signals and configure the waveform properties easily. However, for the "force" part of the do files, we strongly recommend using tesebenches to achieve it. Refer to IQT page 23 for tutorial about testbenches.
- View the report outline for guidelines on writing the report.
- Mirrored Quartus II 15.0 files (Quartus II 15.0 (Win64), ModelSim Altera Edition, Cyclone IV Support)

Document Updates

- We are trying to make this course better by fixing errors and updating the documents. However, this results in several differences between the latest online version and the printed lab manual.
- **09/11/17** Introduction to SystemVerilog (pdf): Added *"The simulator, ModelSim, sometimes interprets untyped **output** as wire, so it is recommended to **explicitly declare the types at the interface**."*
- **09/12/17** Experiment 4 section of the lab manual (pdf)**:** Added descriptions about the correct design of the 4x4-bit hierarchical Carry-Lookahead Adder.
- **09/18/17** Introduction to Quartus II in the lab manual: Removed the descriptions about creating testbenches inside ModelSim, which is known to be buggy. Please follow the instructions starting from IQT. 21 and write testbenches in SystemVerilog in Quartus.
- **09/18/17** Experiment 4 section of the lab manual (pdf): Added clarification that the 16-bit CSA should be composed of 4-bit Carry Ripple Adders.
- **09/22/17**: **In order to make the timing analysis to make sense, you will need to add an "SDC" file** (or this lab4.sdc) **to your project. Follow IQT 18 for how to do so.** If you see fmax being greater than 200MHz, it is not correct. Typical values fall in 60-100MHz.

Demo

- 1.0 point: Functional simulation completed successfully for the 8-bit serial processor (annotations necessary)
- 1.0 point: RTL block diagram of the 8-bit logic processor extended from 4-bits. This is automatically generated using Quartus (you must demonstrate the generation of this block diagram to your TA from source)
- 1.0 point: Correct operation of the Carry-Ripple Adder on the DE2 board
- 1.0 point: Correct operation of the Carry-Lookahead Adder on the DE2 board using a 4x4 hierarchical design (TA's will look at code)
- 1.0 point: Correct operation of the Carry-Select Adder on the DE2 board using a 4x4 hierarchical design (TA's will look at code)

FAQ

- **"I have compiled and tested my code for an 8-bit processor. It works 100% on the DE2 board. However on the simulation, everything seems backwards. When reset is low, the simulation zeroes everything out, when LoadA is 1 it loads into B.**

However, on the DE2, it works perfectly fine"

- The FPGA buttons are active low. That is, a push is recognized as a '0' and doing nothing is recognized as a '1'. The code for the the exercise has inverted the signals for you so that's why it works on the board. You just have to flip the signals so '1' means it's a push.

- **"How should the block diagram be made in SystemVerilog?"**
  - Your block diagram for the SV labs should look like the Schematic diagram from Figure 6 of IQT.27, rather than the top-level block diagram like Figure 1 of p.3.2. If you do your top level entity the schematic way like Figure 6 of IQT.27, you simply have to print that out. From Quartus II you can generate a block diagram through 'Tools -> Netlist Viewers -> RTL Viewer'.
- **"The lab asks for an annotated design simulation in the report, what exactly are we supposed to be annotating/pointing out in the simulation? Is there an easy way to do this in Quartus?"**
  - That means you'll have to create your own waveform and put comments on the simulated results. "Annotation" means you need to tell us what's going on in each step of the waveform. That is, you need to put comments along the simulation saying what's going on in the particular locations (e.g. here's the value for A... here's the value for B... Here we're adding A and B.... here's the result of A+B... here we're hitting "execute"... here we move to state x... here we're doing this in state x... and so on in the waveform)
- **"Also for the state machine can we simply use the one provided by Tools->Netlist Viewers->State Machine Viewer"**
  - Yes. It will show the state flow with 'A'->...->'J' as state representations (since that's how you defined it). However, you will need to explicitly write out what each state is (e.g. Reset->Shift 1->Shift 2->...->Halt) like a regular state machine, so other people can clearly see what's going on. The bottom line is, you'll have to make your state machine understandable standalone. You could also sketch your own state machine if you wish to.
- **"Should the carry-select adder and carry-lookahead adder be made of 16, 1bit adders or made of 4, 4bit adders?"**
  - They should be made in 4-bit components, since flat versions are very inefficient. Demo points will not be given for flat design!

- **"I'm getting a licensing error for ModelSim when I try to simulate the test-bench in the tutorial."**
  - ModelSim defaults to using the licensed version when launched from Quartus. You can fix this by:
    1. Selecting the tools dropdown menu
    2. Choosing Options..
    3. Selecting EDA Tool options
    4. Changing the line that reads c:\altera\15.0\modelsim_**ae**\win32aloem to read c:\altera\15.0\modelsim_**ase**\win32aloem (add the s)
    5. **If you are in the lab machines, ModelSim is installed under c:\altera\16.0\modelsim_ase\win32aloem**
- **"I get the error message 'Illegal reference to net XXX' in ModelSim, how should I fix it?"**
  - ModelSim requires all output signals to have their datatypes explicitly declared. Hence, simply replace "**output [3:0] XXX**" with " **output logic [3:0] XXX**" at the interface of the module and the error message will go away.
- **"When setting the value (forcing) signals in ModelSim using the GUI Wave Editor (as in IQT.8), the waveform for the forced signals appears incorrect (inconsistent with values, or change/glitch as you scroll) - though the simulation output is correct."**
  - As far as we understand, this is a bug in ModelSim's wave editor, likely because very few users use the GUI tools to set test stimuli. Instead, use a SystemVerilog test bench, as described in the tutorial video(s) and the included example testbench for the 8-bit logic processor.