Lab 1: Introduction to EEL4712 Digital Design Lab

EEL 4712 – Spring 2021

Objective:

The objective of this lab is to introduce the software and hardware development tools to be used in EEL 4712 to design, construct, and test digital circuits. In particular, you will review the use of the Quartus software package from Altera for the synthesis of a digital design. You will be creating an 8-bit counter using the schematic features of Quartus, and a 4-bit ripple-carry adder using VHDL. Both designs will be synthesized in Quartus and simulated in ModelSim. Also, you will be introduced to the DE10-lite board, the Digilent Analog Discovery, and to the techniques of using an oscilloscope and logic state analyzer (LSA) to test a constructed digital circuit.

Required tools and parts:

Quartus Prime, ModelSim, DE10-lite board, Digilent Analog Discovery, Signal Tap.

Pre-requisite:

You must be "up-to-speed" with Quartus and ModelSim before coming to lab. Perform any tutorials posted on the website. Also, **download and read** the DE10-lite and Digilent Analog Discovery documents before coming to lab.

Pre-lab requirements:

1. Read the following documents before you come into the lab:
   - Tutorials for LSA and Oscilloscope
   - Tutorial for SignalTap II Logic Analyzer
   - Tutorial for ModelSim
   - DE10-lite Board documents

2. Simulate an 8-bit counter schematic from a block diagram file.
   - Download the provided counter.bdf file.
   - Creating a Quartus project, selecting any one of the **MAX II FPGAS**. IMPORTANT: This is not the FPGA on your board, but Quartus does not currently support timing simulations for the the MAX 10 FPGA, so we are using a MAX II for this simulation.
     - On the EDA Tools Settings screen, under "simulation", select ModelSim-Altera as the simulator and choose VHDL as the format.
   - Implement the design by clicking the "Compile Design" option.
   - Perform a **functional** simulation for the circuit. A functional simulation ignores the timing of the device and assumes that all signals update simultaneously (i.e., all propagation delays are 0).
     - Create a project in ModelSim-Altera. Add the following files to the project. First, in your Quartus project directory, there should be a simulation/modelsim directory. Inside of that directory is a .vho file with the same name as your project. Add this file to the ModelSim project. Also, add the provided testbench counter_tb.vhd.
     - Compile both files. If there are errors, try compiling again because an incorrect compilation order can cause this problem. If there are errors in the testbench, you likely didn't name your I/O correctly in your Quartus schematic.
     - Select Simulate->Start simulation
     - On the Design tab select work->counter_tb, which is the testbench.
     - In the Objects window, select the signals you would like to monitor and drag them into the Wave view.
     - Click the Run –all button, or alternatively, keep clicking the Run button until a "Simulation Finished" message is printed.

- In the Transcript/Console window, check for any failed assertions messages. If something went wrong in your design, these error messages will tell you when the errors occurred.

- Perform a **timing** simulation for the circuit.
  - Fortunately, not much changes for a timing simulation. Assuming you already have the project created, select Simulate->Start Simulation. You might need to select End Simulation if you are still doing the functional simulation.
  - Quartus uses a .sdo file to annotate simulations with actual propagation delays. To include this in your ModelSim project, select the SDF tab (on the dialog box that comes up after Start Simulation). Add the .sdo file that is in the quartus_project/simulation/modelsim directory. This should be the same directory as the .vho file that you previously added. **IMPORTANT: In the "Apply to Region" text box, make sure to type /UUT**. I'll explain this later, for now just remember to do it.
  - Everything else is the same as the functional simulation. Note that in the resulting waveforms, the outputs are delayed by a small amount, instead of changing immediately on every rising clock edge.

Turn in on e-learning: the design of the 8-bit counter (.bdf file) and screenshots of working functional and timing simulations that show "Simulation Finished" without any assertion errors.

3. Design and simulate a 4-bit ripple-carry adder in VHDL.
   - Create a full adder entity in VHDL, using the template provided on the website (fa.vhd). Do not change the names of any port signals.
   - Perform a *functional* simulation of the full adder in ModelSim using the provided testbench (fa_tb.vhd).
   - Create a 4-bit ripple-carry adder in VHDL using a structural architecture (e.g., port map statements) that combines four of your full adders into a 4-bit ripple-carry adder. Make sure to use the template provided on the website (adder.vhd). Do not change any of the port signals.
   - Perform a *functional* simulation of the ripple-carry adder in ModelSim using the provided testbench (adder_tb.vhd).
   - Perform a *timing* simulation of the ripple-carry adder by first synthesizing the code in Quartus, and then importing the .vho and .sdo files into ModelSim. Use the same testbench as the previous step.

Turn in on e-learning: all VHDL and simulation screenshots for pre-lab step 3

4. **IMPORTANT**: To run your code on the board, you have to create a new Quartus project and select the FPGA on your board. When creating the Quartus project, make sure to use the following options: On the Family & Device Settings screen, select the Board tab at the top. Next, select the MAX 10 DE10 – lite board. **Unselect the "Create top-level design file" option.**

5. Using Quartus, assign pins and download the 8-bit counter from the pre-lab to your board. (Ensure that all your unused pins are reserved as "**Input Tristated**" when you assign pins.

This option can be found under 'Assignments->Device->Device and Pin Options->Unused pins'. **Remember to do this for every project you make**.)

- Read the pin assignment tutorial included with lab 1 for instructions.
- Assign all inputs (except for the clock) to the switches and/or buttons
- See the user manual for pin assignments (http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=1021&FID=a13a2782811152b477e60203d34b1baa)
- For more information on pin assignment, here is a tutorial video: https://www.youtube.com/watch?v=XH4OajcYYjA

Pre-lab turn in instructions:

To submit your pre lab please create a folder that is named your UFID. Inside create another folder P1,P2,P3... for each part that contains the VHDL files for that part. If a report is needed then please include it in the UFID folder. Then zip and upload the folder which is named your UFID. There is nothing to turn in for parts 4-5, but they should be done before lab to ensure you have time to demo the functionality during lab. If you run into problems, you can get help during the lab section without any penalty.

In-lab procedure:

1. The TA will give you a demonstration of
   - the use of the SignalTap II LSA
   - the use of the Digilent Analog Discovery

2. Show the correct functionality of the counter using either the in-lab LSA or the Digilent Analog discovery.

3. Based on the Tutorial for SignalTap II Logic Analyzer, use the SignalTap II tool from Quartus to obtain a waveform display of your 8-bit counter. Demonstrate the output to the TA.