ECEN3002 - Lab 10

Fall, 2020

Due Thursday November 5, 11:59pm – 25 Points

After completing this assignment, you will submit your Quartus archive file (.qar).

The purpose of this lab is to practice using SystemVerilog (SV). You are given an existing design and will go through and modify the design to update to SV. You should refer to the Lecture 19 slides for the details of SV syntax. A secondary goal of this project is to demonstrate different ways to program your board.

You are provided with a .qar Quartus archive of a project that interfaces to the LTLC2308 8 channel, 12 bit analog to digital converter chip on the DE10-Standard board. The LTC uses a 4 pin SPI interface for programming and data interface. The project sets up the LTC2308 to continuously sample the channel programmed selected using either SW[2:0], or by setting the channel using In-System Sources and Probes interface.

The ADC project consists of 4 source files. A brief description of each file is provided here, but you will see that the source files contain comments as well. The archive file contains all necessary files for the PLL and ISSP blocks so you will not need to modify these blocks.

a) adc.v

This top level file instantiates a PLL, the adc_control logic module, as well as a binary to BCD conversion block and the required 7 segment decoders. You can also see that various signals were connected to the 2 x 20 pin GPIO header on the board to allow connection of an external logic analyzer.;

b) adc_control.v

If interested, you can read the LT2308 datasheet to see how the timing works. The adc_control block generates all the proper timing signals and sets the sample rate.

c) bin2bcd.v

Converting a binary value to a multi-digit BCD display is a tricky problem. I found some code to do this at the link listed in this file. The code uses looping constructs to make the conversion. Generally use of loops in synthesizable code can be problematic, but the code simulated fine, so I extended the original code to add one more BCD digit.

The purpose of this block is to take the 12 bit input value from the LTC2308 and convert that binary value to 4 BCD digits.

d) seg7.v

Simple 7 segment decoder that can display 0 – F on a hex display. For this design, the values are always in the range 0 – 9.

Part 1: Open Project and Program Board

1) Open the archive file in Quartus. You will find that a programming file (adc.sof) already exists in the quartus directory. Program the board (see next step) to confirm proper operation before making any changes or before recompiling in Quartus.

2) To program your board, open a command prompt. Execute the jtagconfig.exe program that is found in the <quartus_install_directory>\20.1\quartus\bin64 directory. JTAG config reports the name of your download cable (should be DE-SoC) and also displays the JTAG IDs of the two JTAG blocks in the FPGA – SOCVHPS first, 5CSEBA6 second.

Note: The Quartus executables should already be in your path.

3) At the command prompt, navigate to the directory location containing the adc.sof programming file, and then type the following:

```
quartus_pgm -c DE-SoC -m jtag -o p;adc.sof@2
```

and verify your board programmed correctly. You should see a value in the 4 leftmost HEX displays. The value will vary but the first two digits should be 18.

In the programming command line above, the -c switch identifies the cable, -m sets the programming mode as jtag. The -o switch identifies the programming file. You must include the @2 to indicate you are programming the second JTAG device in the chain.

4) Now that you have confirmed that the Quartus archive is good and that the project works on your board, Part 2 will walk you through modifying the source code.

Part 2: Convert project to SystemVerilog and verify correct operation

1) In Quartus, change the project compiler setting to SystemVerilog. This is done using Assignments > Settings > Compiler Settings/Verilog HDL Input.

2) Change the file extensions of the 4 input files from .v to .sv. Remove and add the files to the Quartus project if required. Rerun the Analysis & Synthesis step in Quartus to confirm that no errors are seen.

3) Modify the source files with the following changes to SystemVerilog syntax:

a) Replace wire and reg data types with the logic data type everywhere. Remember you must add the keyword logic to all inputs and outputs also.

b) always_ff for all synchronous always blocks

c) always_comb for all combinatorial always blocks. Remember that there is no sensitivity list.

d) Add block names to the begin/end blocks.

e) In the adc.sv file, , replace .reset_n(reset_n) with .reset_n in the adc_control instantiation, to take advantage of the implicit .name port connections of SV.

f) Add the unique keyword before the word case in all case statements.

g) Once you have made all the changes and recompiled your project, program your board using the Quartus Programmer.

h) Open the Tcl console within Quartus. Open the console using View > Utility Windows > Tcl Console. At the Tcl console, type

qexec “quartus_pgm adc.cdf”

qexec is a method to execute a Windows or Linux executable file from Tcl. The .cdf file was created when set up the programmer in step g) above.

qexec is a method to execute a regular executable file from Tcl. Since the project contains a .cdf file, that is all you need.

Part 3: Testbench framework

You won’t create a complete testbench, but it is instructive to see how the implicit port connections can assist. One reason a testbench is tricky is because we need a model for the LTC2308 device, since we are communicating with the chip on the board.

a) Create a testbench file with a .sv extension.

b) Create signals for all the ports in adc.sv. Name the signals the same name as the adc.sv ports, and make the signals type logic.

c) When you instantiate adc, use this syntax:

adc U0 (.*); // This is the entire instantiation

c) Create a clock.

d) Initialize the inputs to some value, and run the simulation for a short time.

e) Set up your project for simulation. Run Analysis & Synthesis, then launch the simulation in Modelsim.

f) If you get an error in Modelsim regarding pll.vo, do these steps:

1. If you have a pll_sim folder in your project, copy the pll.vo file from there into the quartus directory.

g) If you don’t have a pll_sim folder in your project, regenerate the PLL in Quartus first, then copy the pll.vo file to the quartus directory. The simulation should now run. The simulation doesn’t show anything useful, but is confirmation that the implicit port connections method in SystemVerilog works.

Part 4: Program the Flash on your board

Quartus normally generates a programming file for serial JTAG programming (.sof). To program the Flash chip on the board, a different format programming file is required. The instructions that follow are specific to the DE10-Standard board.

To program the flash chip on the DE10-Standard board with this adc program:

a) File > Convert Programming Files, and click the Open Conversion Setup Data button.

b) Select the .cof file, and change the Programming file type to Indirect Configuration File (.jic), make sure the EPCS128 is selected, and the Mode field set to Active Serial.

c) Highlight SOF Data in the Input files to convert pane, click Add File, and select the adc.sof file.

d) Clock on Flash Loader in the Input files to convert pane, click on Add Device. When the Select Families window opens, highlight Cyclone V on the left side, and select SCSXFC6D6 (next to the last selection) on the right side. Click OK to dismiss window.

e) Click Generate and close window.

f) Open the Quartus Programmer. Double click on the adc.sof file, and when the Select New Programming File window opens, select the .jic file.

g) In the line that lists the .jic file, click on the first box in the Program/Configure column, then click the Start button. It will take 30 seconds or so to erase the Flash device, then you will see the green Progress bar in the upper right slowly increment until programming is complete. It will take about 5 minutes to complete programming. Once complete, you may need to turn your board off and back on to see the result.

h) The factory program that came with the board has now been replaced with the adc program. If you want to get the factory program back, the instructions can be found in the DE10-Standard User’s Manual, chapter 5. Note that Terasic has included a batch file to automate the steps you performed above.

The FPGA gets reprogrammed with a special program that then reads the programming file and programs the Flash part on the board.