University of California, Davis
Department of Electrical and Computer Engineering

EEC180B DIGITAL SYSTEMS II Spring Quarter 2014

LAB 1: ModelSim / Quartus II Tutorial and Lab Exercise

Objective: This tutorial covers the complete design flow for implementing a high-level Verilog design on the Altera DE2 board. ModelSim is used for functional and timing simulation, while Quartus II is used for synthesis and device programming.

Setting up ModelSim and Quartus II

In order to run ModelSim or Quartus II on the department Linux PC's in 2110 or 2107, you must run the appropriate setup script to modify the *.software* file in your home directory. Run the setup script for ModelSim with the command:

```
setup modelsim
```

Similarly, run the setup script for Quartus II using the command:

```
setup quartus
```

You only need to run these setup scripts *once* and then open a new window to get the correct path and environment variables.

I. Functional Simulation in ModelSim

The first step in the design process is a functional simulation of the Verilog description of your circuit. We will use the ModelSim simulator in order to simulate and debug Verilog designs both interactively and using a test bench. Although Quartus II also has a simulator, it doesn't have many of the debugging features of the ModelSim simulator.

A. Create a ModelSim project

1) ModelSim should be run from a GNOME session on the department's Linux workstations. Another option is to open an xterm window to create a suitable environment.

2) Create a new directory such as eec180b/lab1 and copy the design files into it. The design files are named "updown.v" and "tb_updown.v" and are located in the directory /afs/ece/classes/eec180b/lab1_verilog. I.e.

```
mkdir lab1
cd lab1
cp /afs/ece/classes/eec180b/lab1_verilog/*.*  .
```

3) Run ModelSim using the following command:

```
vsim &
```

4) Select **File > Change Directory** and change to the directory you created in step 1. Changing the directory will close the currently open project, if there is one.

5) Select **File > New > Project…** to create a new ModelSim project. In the "**Create Project**" dialog box, enter the following:

   Set Project Name to: **lab1**
   Set Project Location to: **the directory you created in step 1**.
   Set Default Library Name to: **work**
   Copy Settings From:
   /afs/ece.ucdavis.edu/i386_linux24/pkg/modelsim-6.3/modeltech/modelsim.ini
   (Click on the Browse button to navigate to the correct directory. You can type *.* in the File Name box to list all files in the modeltech directory. Then select the modelsim.ini file. The **@sys** in the default path will map to **i386_linux24**, as shown in the path above.)
   Select the option: **Copy Library Mappings**
   Click **OK**
   (If you are given a choice about which modelsim.ini file to use, choose the default file.)

6) A window titled "**Add Items to the Project**" will pop up on the successful creation of the project.
   Double-click the **Add Existing File** icon.
   Click the Browse button and select the two Verilog design files in your project directory and click **Open**.
   Select the option: **Reference from current location**
   Click **OK**
   Click **Close** to close the "**Add Items to the Project**" window.

B. Compile and Simulate a Verilog Design in ModelSim

1) Select **Compile > Compile All** to compile updown.v and tb_updown.v. You should not get any errors.

2) Select the **Library** tab at the bottom of the workspace pane.
   Expand the **work** library by clicking the "+" box next to it.
   You should see the updown and tb_updown modules and their paths.

3) Enter the following command at the ModelSim> prompt in the Transcript pane:

```
vsim –voptargs="+acc" tb_updown
```

The –voptargs= “+acc” argument for the vsim command provides visibility into the design for debugging purposes.

4) Select **View > Wave** to open the Wave window for simulation output. The Objects window should already be open. If not, open the Object window by selecting **View > Objects**.

5) Right-click on the tb_updown module in the Workspace pane. This will bring up a pop-up menu. Select **Add > Add to Wave** to add the test bench signals to the Wave window.

6) Next select the UUT module in the Workspace pane. This will change the signals listed in the Objects window. Right-click on the signal “Count” and select **Add to Wave > Selected Signals** in the pop-up menu. This should add the signal “Count” to the Wave window.

7) Right-click on “Count” in the Wave window and select **Radix > Hexadecimal** to change the radix from binary to hex. Do the same thing for the signal “LEDR”.

8) From the Workspace pane, select tb_updown again so that the top-level signals are in the Objects window. Since the test bench code does not drive the Updown input signal, we will control this signal interactively from the Objects window.

9) Set Updown to 1 in the Objects window as follows:
   Right-click on Updown in the Objects window and then select **Force…** from the pop-up menu. This will bring up a Force Selected Signal dialog box. Change the Value from its current setting (probably ‘x’) to 1. “Kind” can be left on “Freeze” and “Delay for” should be left at 0.

10) At the VSIM prompt in the Transcript window, type **run 100 ns** to run the simulation for 100 ns. You should see the first several Clock cycles in the Wave window. (To view the Wave window, it can be helpful to “undock” it by clicking the button in the upper right of the window between the “+” and the “x” buttons.) You can then resize the window to a convenient size. You can also use the zoom controls by selecting **View > Zoom** or by right-clicking on a signal trace in the Wave window and using the pop-up menu.

11) Set Updown to 0 and simulate for several cycles until the Count value rolls over from 0000 to FFFF. Continue counting down for a few more Clock cycles.

12) Set Reset_n to 0 and simulate the asynchronous reset of the counter.

13) Select **File > Print Postscript** from the Wave window to print your simulation waveforms. *Print your functional simulation waveform for your lab report.*

14) Other useful features in ModelSim include setting breakpoints and single-stepping through the Verilog source code, while watching signal changes in the Objects window. We will not discuss this in detail here. However, you can experiment with setting breakpoints by opening up the source files and left-clicking next to a red line-number. Right-clicking will bring up a pop-menu that allows you to edit, disable or remove breakpoints. You can even set a Breakpoint Condition so that, for example, the simulation stops on a certain line when a condition such as Count=0000000000001000 is true.

    To single-step, you can just type "step" at the prompt in the Transcript window or select **Simulate > Run Step**.

15) *Demonstrate your simulation to your TA and have the TA sign your verification sheet.*

II. <u>Synthesis and Device Programming in Quartus II</u>

Now that we have verified the functional correctness of the updown counter, we can synthesize the design and download it to the Altera DE2 board.

A. Creating a Quartus II Project

1) Run Quartus II with the following command:

    quartus &

2) If your Quartus Window has a menu option **Max+PlusII** or if you don't get multiple window panes such as Message and Entity frames, you may have Quartus configured to run with a Max+PlusII look-and-feel. To change to the Quartus II look and feel select **Tools > Customize…** and select the Quartus II button under Look & Feel. Also, under the Toolbars tab, make sure that Standard Quartus II is selected.

3) Create a new project by selecting **File > New Project Wizard…**

    An introduction page will come up displaying general information. When you are done with it, click **Next**.

    On p. 1, specify a working directory for the project. This should be the project where your updown.v file is located. Also, give the project and the top-level design entity a name. (Typically, these have the same name.) For this tutorial, name the project and top-level design module "**updown**" since we have used the name **updown** for the top-level Verilog module. Click **Next**.

    On p. 2, you can select design files to include in the project. Since you typically won't have an existing file, just click **Next**. We can add design files in other ways.

On p. 3, select Family as **Cyclone II** and Target Device as "**Specific device selected in 'Available devices' list**". Select **EP2C35F672C6** from the list of devices and click **Next**.

On p. 4, under Simulation select Tool Name as **ModelSim** and Format as **Verilog**. Click **Next**.

On p. 5, click **Finish** to complete the creation of the new project.

4) Normally, you might select **File > New…** and then select **Verilog HDL File** to open up the Quartus II text editor to enter your Verilog code. However, in this tutorial, you have been given the design files. Instead, select **File > Open**. Select "**updown.v**" and click the box "**Add file to current project**".

   Note that only "updown.v" will be part of the Quartus II project. The test bench file, "tb_updown.v" is only for simulation in ModelSim and will not be used in Quartus II.

```
module updown (SW, KEY, LEDR, LEDG);
input [0:0] SW;             // Updown switch 1=up, 0=down
input [1:0] KEY;            // KEY[1] = Clock, KEY[0] = Reset_n
output [15:0] LEDR;         // Display binary count (active high) on Red LEDs
output [1:0] LEDG;          // Display Clock on LEDG[1], Reset_n on LEDG[0]
wire Clock, Reset_n, Updown;
reg [15:0] Count;

assign Clock = KEY[1];
assign Reset_n = KEY[0];
assign Updown = SW[0];
assign LEDR = Count;
assign LEDG[1:0] = {Clock, Reset_n};

always @(negedge Reset_n, posedge Clock)
        if (Reset_n == 0)               // active-low asynchronous reset
                Count <= 0;
        else if (Updown)                // if Reset_n != 0, Clock rising edge
                Count <= Count + 1;     // count up if Updown=1
        else
                Count <= Count - 1;     // count down in Updown=0

endmodule
```

Figure 1. Verilog code for example updown counter circuit

5) If you make any changes to the file, select **File > Save** to save it. You can close the file, if you wish.

B. Compiling a Design in Quartus II

1) Since we intend to download this design to the Altera DE2 board, we need to specify pin assignments so that the input and output signals are mapped to the proper switches and LEDs on the DE2 board. A convenient way to make the required pin assignments is to import the file **DE2_pin_assignments.csv** that is available in the directory /afs/ece/classes/eec180b/lab1_verilog.

   Copy **DE2_pin_assignments.csv** from /afs/ece/classes/eec180b/lab1_verilog to your project directory (or perhaps into your eec180b directory since the file will be used in all labs).

   To import the pin assignments, select **Assignments > Import Assignments…** and specify the file name or browse to the file location on your system and then click **OK**.

   Note that this is a very important step! If you forget to specify pin assignments, your design will ***not*** work when downloaded to the DE2 board because the I/O pins will not be mapped correctly. Also, it is important to realize that the pin assignments in **DE2_pin_assignments.csv** only work if the port names used in your Verilog module are exactly the same as the pin names given in the file. This is the reason we specified the port names SW[0], KEY[1], KEY[0] for the input signals instead of more descriptive names such as Clock, Reset_n and Updown.

2) To compile the design, select **Processing > Start Compilation** or click on the Start Compilation icon on the toolbar. If there are any compilation errors, fix them and re-compile the design.

C. Running the Quartus II Classic Timing Analyzer and RTL Viewer

1) Once your design has compiled, you can check its maximum frequency with the Quartus II Classic Timing Analyzer Tool. Select **Processing > Classic Timing Analyzer Tool**. You should see that the circuit can operate with a substantially shorter period than 10 ns. Click on **Report** to see a summary of the timing analysis.

   Exercises:

   1. *What are tsu, tco, tpd, and th? What do each of these times signify?*
   2. *What is the critical path of your circuit and the maximum frequency of operation?*

For more information on using the QuartusII Classic Timing Analyzer, refer to the Quartus II Handbook set, Chapter 8 of Volume 3: Verification. This material can be found on-line at:

http://www.altera.com/literature/hb/qts/qts_qii53004.pdf

2) Another interesting tool is the RTL Viewer, which allows you to see a schematic representation of your synthesized design. Select **Tools > Netlist Viewers > RTL Viewer** to see the type of circuit produced from the Verilog HDL code.

Exercise:

3. Examine and print the schematic of your synthesized circuit. Explain how the up-counting and the down-counting functions are implemented. Verify the logic used for counting down.

D. Downloading a Design to the Altera DE2 board using Quartus II

1) Start Quartus II on the download station. Select **Tools > Programmer** to bring up the device programmer tool.

2) Click on the **Add File…** button on the left-hand side of the Programmer window. Browse to the folder containing your **updown.sof** file, select it, and click **Open**.

3) Verify that the Hardware Setup is configured for USB-Blaster and that the DE2 board is powered and connected to the PC with the USB-Blaster cable.

4) Click the **Program/Configure** box and then click **Start**. The file should be loaded into the Altera DE2 board. Use pushbuttons KEY1 and KEY0 and switch SW0 to test your circuit.

5) Demonstrate your working circuit to your TA and have your TA sign your lab verification sheet.

III. **ModelSim Timing Simulation and Quartus PowerPlay Power Analyzer**

The procedure for performing gate-level timing simulation of a Quartus II design using ModelSim is described in detail in the Quartus II Handbook set: Chapter 2 of Volume 3: Verification. This chapter is available on-line at

http://www.altera.com/literature/hb/qts/qts_qii53001.pdf

The material in this section is adapted from this Altera document for our specific environment. Please refer to the on-line material if you desire more information.

Gate-level timing simulation is a ***post*** place-and-route simulation using (in our case) worst-case delays. However, before you compile your design, you will set configuration options to produce a Verilog output file (**.vo**) and a Standard Delay Format Output file (**.sdo**), which will both be used in the gate-level simulation.

The Quartus II PowerPlay Power Analyzer tool is described on Altera's website at:

http://www.altera.com/literature/hb/qts/qts_qii53013.pdf

As with the gate-level timing simulation, the PowerPlay Power Analyzer tool requires that your design is synthesized and placed and routed on the target device. In our case, we will use the ModelSim gate-level timing simulation results to generate signal activity and static probability information in a Value Change Dump file (**.vcd**).

A. Configuring Quartus II for Timing Simulation and Power Analysis

1) Open your Quartus II project. On the **Assignments** menu, click **EDA Tool Settings**. The **Settings** dialog box appears.

2) In the **Category** list, click the "+" icon to expand **EDA Tool Settings** and select **Simulation**. The **Simulation** page appears.

3) In the **Tool name** list, select **ModelSim**.

4) Under **EDA Netlist Writer options**, in the **Format for output netlist** box, select **Verilog**. By default, the post-synthesis netlist will be written to the <*project_directory*>/**simulation/modelsim** directory.

5) Check the box **Generate Value Change Dump (VCD) file script**. This should also add checks in the boxes for **Map illegal HDL characters** and **Enable glitch filtering**. Also fill in the **Design Instance Name** with the instance name for your updown counter given in the testbench file (**UUT**). Click **OK** to close the Simulation dialog box.

6) Perform a full compilation of your design. On the **Processing** menu, click **Start Compilation**. Verify that your design compiles without errors and that the Verilog Output file (.vo), the Standard Delay Format Output file (.sdo), and a Value Change Dump (VCD) file script (.tcl) are all written to the <*project_directory*>/**simulation/modelsim** directory.

7) Exit Quartus II.

B. ModelSim Timing Simulation

1) Copy your testbench program to the <*project_directory*>/**simulation/modelsim** directory.

2) Run ModelSim from the <*project_directory*>/**simulation/modelsim** directory. Using the procedures described earlier, create a new ModelSim project and add the testbench file (.v) and the Verilog Output file (.vo) to the project.

3) Compile your design files (testbench and post synthesis netlist) in ModelSim.

4) Run the gate-level simulation by typing the following command at the ModelSim command prompt:

vsim –t 1ps –L altera –L altera_mf –L lpm –L cycloneii –L work work.tb_updown

(assuming the top level module of your testbench is named *tb_updown*).

5) Open a Wave window as before and add the testbench signals and the UUT Count signal to the Wave window. Change the radix of Count and LEDR to Hexadecimal.

6) Force SW[0] (Updown) to 1 and simulate as before.

Exercises:

4. Compare the timing simulation results with the results from the Quartus II Classic Timing Analyzer Tool. Are they in agreement?

5. Print a portion of your timing simulation waveform for your report.

C. Quartus II PowerPlay Power Analyzer

1) To generate a VCD file for the Quartus II PowerPlay Power Analyzer, restart your simulation. (Select **Simulate>Run>Restart…** or type "**restart –f**" at the command prompt)

2) Run the tcl script using the **do** command as illustrated below:

do updown_dump_all_vcd_nodes.tcl

(Use the actual name of your script file if it doesn't match the example given above.)

3) Now force SW[0] to 1 and simulate for about 4000 ns. Then force SW[0] to 0 and simulate for another 4000 ns.

4) Quit the ModelSim simulator using the command **quit**. Verify that ModelSim has generated a VCD file (.vcd) in your directory.

5) Run QuartusII and open your original QuartusII project.

6) On the **Assignments** menu, click **Settings**. The **Settings** dialog box appears. In the **Category** list, select **PowerPlay Power Analyzer Settings**.

7) Select the setting **Use input file(s) to initialize toggle rates and static probabilities during power analysis**. Then click the **Add…** button and browse

to the correct directory to add the VCD file generated during your ModelSim timing simulation. Also, select the **Perform glitch filtering on VCD files**. Write the signal activities and the power dissipation by block to the report file.

8) On the **Processing** menu, click **PowerPlay Power Analyzer Tool**. Click **Start** to run the tool. Once the **PowerPlay Power Analyzer Tool** completes, click **Report** to open the **PowerPlay Power Analyzer Summary** window.

Exercise:

6. *Print the Power Analyzer Summary for your lab report.*

IV. Lab Exercise

In this exercise, you will extend the updown counter to 32-bits and add 7-segment display outputs. Figure 2 shows a hex to 7-segment display code converter. This code converter should be used to display hexadecimal characters (0-9, A-F) on the 7-segment display. The Altera DE2 board has eight seven-segment displays, named HEX7, HEX6,..., HEX0. Each segment is illuminated by driving its control signal to logic 0; that is, they are ***active-low***. In the *DE2_pin_assignments.csv* file, the segments are identified by the indices 0 to 6 as shown in Figure 2. Thus, you should declare the 7-bit ports

output [0:6] HEX7, HEX6, HEX5, HEX4, HEX3, HEX2, HEX1, HEX0

in your Verilog code so that the output names match the corresponding names in the pin assignments file.

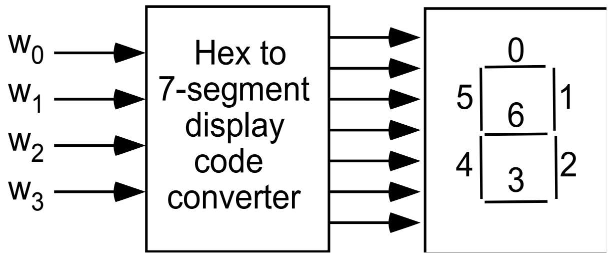

Figure 2. Code converter Block Diagram

Perform the following steps:

1. Extend the updown counter design to 32-bits. (This should be a very straightforward change.) Note that you will not be able to display the full 32-bit count on the red LEDs since the DE2 board doesn't have enough LEDs.

2. Create a Verilog module for the 7-segment code converter. Instantiate eight of your code converters into your updown counter module in order to drive the HEX7-HEX0 7-segment displays on the DE2 board. The 32-bit Count signal should be displayed in hexadecimal. You can use the template given below for your Hex to 7-segment display code converter.

```
module hex_7seg (
    input [3:0] hex,
    output reg [0:6] seg
    );

//              012_3456 (segments are active-low)
parameter ZERO =  7'b000_0001;
parameter ONE  =  7'b100_1111;
// etc.
parameter F = 7'b011_1000;

always @(hex)

case (hex)
0:  seg = ZERO;
1:  seg = ONE;

// etc.
15: seg = F;
endcase

endmodule
```

3. Perform a functional simulation of your Verilog design to verify your design's correctness. *Print a waveform showing the code converter outputs for Count values 0 through 15.*

4. Compile your design in Quartus II and download the circuit to the DE2 board. Test your circuit using the pushbuttons and switches on the DE2 board. *Have your TA verify your working circuit.*

<u>Lab Report</u>
Submit the following items for verification and grading:

1) Lab cover sheet with TA verifications for
    - 16-bit updown counter functional simulation
    - 16-bit updown counter implementation on DE2 board
    - 32-bit updown counter implementation on DE2 with 7-segment displays
2) Complete Verilog source code for Part IV, Lab Exercise. Submit your revised test bench module as well as the updown counter and hex to 7-segment display modules.
3) Functional and timing simulation waveforms
4) Answers to all exercises