CSE 141L

Altera Quartus II & ModelSim Walkthrough

These instructions are a step-by-step walkthrough to start using Altera Quartus II and ModelSim. I did the walkthrough for QII 13.0 and MS 10.1b. Both have more recent versions, but there shouldn’t be any significant differences. If there are, please let me know.

## Software Installation

I’m assuming you already have Quartus II and ModelSim installed. If not, go to <http://www.altera.com/products/software/quartus-ii/web-edition/qts-we-index.html> . You’ll have to create an account and log in.

## Using Quartus II

1. Run Quartus II by launching the shortcut Quartus II 13.0 Web Edition in the Start menu. The first time you run it, you will probably be asked for licensing options. Select Run the Quartus II software and click OK to continue. You will be allowed to use Verilog or SystemVerilog (but please don’t mix them up). This walkthrough will use SystemVerilog.

Notice in the labs that you are asked to turn in a combination of schematic and Verilog. This will walk you through an example that does exactly that – highest-level schematic with Verilog underneath.

1. Create a new project by clicking *Create a New Project (New Project Wizard).* The next five steps will help you setup your project:
   1. In page “Directory, Name, Top-Level Entity”, enter the working directory for your project, that is, the directory where sources and all intermediate files will be stored. It is strongly advisable to create a new directory for each of your projects. For 141L it is a good idea to create separate projects for each of the labs 2, 3, and 4. Enter a name for your project in the second field. For this walkthrough let’s name our project walkthrough. The top-level design entity can be changed later on, so don’t worry about it at this point. Click Next.
   2. In page “Add Files”, you can add any existing files to your project. No existing files for this walkthrough, so click Next to go to the next page.
   3. In page “Family & Device Settings”, you must select the FPGA board model for which Quartus will generate hardware. We will not use FPGAs in this class, but you need to select family Cyclone IV and device EP4CE115F29C7, so that your timing information is comparable to other students. Next.
   4. In page “EDA Tool Settings”, make sure that ModelSim-Altera is the tool selected for simulation and SystemVerilog HDL the language.
   5. Finally, press Finish to create your project.
2. Our walkthrough project will consist of three files, the top level schematic, the verilog design file and the testbench. Let’s add the Verilog file first, which will then allow us to create a symbol we can add to the schematic. Click *File→New...* and select SystemVerilog HDL File under Design Files. Click OK. In the textbox that opens, paste the following code, which implements a simple register with a reset signal and a write enable:

module program\_counter

(

input [7:0] newpc\_i,

input clock,

input wenable\_i,

input reset\_i,

output [7:0] pc\_o

);

reg [7:0] pc, pcnext;

assign pc\_o = pc;

always\_comb

begin

if (reset\_i) begin

pcnext = 0;

end else if (wenable\_i)

pcnext = newpc\_i;

else

pcnext = pc;

end

always\_ff @(posedge clock)

pc <= pcnext;

endmodule

and save the file as program\_counter.sv

1. Now create a circuit element that you can paste into a schematic. Go *File->Create/Update->Create Symbol Files for Current File*.
2. Open a schematic file, File->New…, then select “Block Diagram/Schematic File”. OK. In the schematic window, select the icon that looks like an *and* gate (Symbol Tool). In the Libraries pane of the dialog that comes up, select your project directory, and you should see the created symbol program\_counter. Select it, then OK, then place the symbol in the schematic window. It should be a block with four inputs on the left and one output on the right. Now select the “Pin Tool” (right next to the Symbol Tool), and select “Input”. This will allow you to place four input ports, connected to the four programcounter inputs. Click on the pointer tool. Give each input port a name, by right clicking the input port and selecting Properties… Give them the names (should be obvious which is which) *newpc[7..0], clk, wenable,* and *reset*. Place an output port connected to the symbol’s output, and give it the name *currentpc[7..0].* Now we are done adding things to this file, save it as walkthrough.bdf. Now one last thing you need to do with this file. Since Modelsim does not work with schematics, we need a Verilog file. Click *File->Create/Update->Create HDL Design File from Current File…* Select Verilog in the next dialog and then OK. It will create walkthrough.v.
3. Now we need the test bench. *File->New…*, then SystemVerilog HDL file, then OK. Paste in the following:

module testbench();

// Declare inputs as regs and outputs as wires

reg [7:0] newpc;

reg clk;

reg wenable;

reg reset;

wire [7:0] currentpc;

// Initialize all variables

initial begin

clk = 1; // initial value of clock

reset = 0; // initial value of reset

wenable = 0; // initial value of wenable

newpc = 2; // initial value of newpc

#10 reset = 1; // use reset to set pc to 0

#10 reset = 0; // end of reset pulse.

#10 // wait a cycle, make sure the PC doesn't change

#10 wenable = 1; newpc = 4; // set wenable high, pc should change

#10 newpc = 5; // try a few PC values

#10 newpc = 7;

#10 reset = 1; //see if reset works when wenable high

#10 reset = 0; wenable = 0;

end

// Clock generator

always begin

#5 clk = ~clk; // Toggle clock every 5 ticks

// this makes the clock cycle 10 ticks

end

// the following creates an instance of our program\_counter register.

// I copied this code verbatim from the walkthough.v that was

// generated by Quartus when I created the .v file from the .bdf.

program\_counter b2v\_inst(

.clock(clk),

.wenable\_i(wenable),

.reset\_i(reset),

.newpc\_i(newpc),

.pc\_o(currentpc));

endmodule

and save the file as testbench.sv. Notice the piece at the end that should be exactly the same as a section of walkthrough.v. If it isn’t, fix it or just paste in what you find there.

1. Start synthesis by selecting the bdf file and clicking *Processing->Start Compilation*. After synthesis completes, you should see a compilation report with various statistics and several diagnostic messages produced in the bottom window. It’s good practice to skim through any warnings or (worse) critical warnings that arise – and even better to fix them. Errors will also show up there.

## Timing analysis with Quartus (note: still working out some kinks in this part. Can skip for now if you’d like.)

After Quartus has analyzed (and synthesized) your design, it allows you to review important results about your circuit’s timing, area and clocking, among others. The following list mentions some of these results that appear in the “Compilation Report” tab:

* Maximum clock frequency. You can see the maximum clock frequency that guarantees correct execution of your design in TimeQuest Timing Analyzer→Slow Model→Fmax Summary. Note that this refers to the device (FPGA) you have configured your project with. Note: In purely combinational simulation, this has nothing to do with the testbench clock since, in this case, signals are assumed to propagate with zero delay; instead, this value is significant for actual hardware and timing simulation – in those cases, setting the clock to a frequency higher that Fmax will most probably produce wrong results.
* Worst-case timing paths, under TimeQuest Timing Analyzer→Slow Model→Worst-Case Timing Paths→Minimum Pulse Width: ’clk’. This is a list of all register-to-register paths in your design (“From Node” and “To Node” columns), sorted by delay time (“Data Delay” column). Remember that the maximum clock frequency is determined by the critical path delay, which is the path of worst- case timing. Therefore, to optimize for clock cycle time, you need to find ways to minimize the maximum delay. In a good design, it is usually the case that the majority of paths have close- to-worst delay. In certain cases you might be able to even visualize a path by right-cliking on it and selecting Locate→Locate in RTL Viewer.

Note – I don’t really expect every group to do careful timing analysis, but I know some will want to, and you may get a performance boost if you do so.

## Using ModelSim

After completing the steps in the previous section, you should have set up project walkthrough with three files that will be needed by ModelSim: testbench.sv, walkthough.v, and program\_counter.sv. You can start Modelsim from within Quartus, if you like, with Tools->Run Simulation Tool->RTL Simulation, or just run it directly. Once the ModelSim main window is open (may have to close first dialog):

1. *File->Change Directory* to your project directory.
2. Select Compile→Compile... Select all your source files (walkthrough.v, program\_counter.sv, testbench.sv) and then click Compile. If it asks if you want to create a “work” library, say yes. Click Done after you’re done. Any errors during compilation will produce a message in the “Transcript” window. (Note: ModelSim is more picky with Verilog than Quartus; don’t be surprised if something appears to compile fine in Quartus but fails in ModelSim. Go and fix it.) You should now see all modules that compiled successfully under node work in window “Library”, after you expand the work library at the top. Start the simulation by double clicking on testbench.
3. If everything went fine, you should see a bunch of new windows, including “sim”, “Objects” and “Wave”. If the waves are not visible, select View→Wave. (Note: Screen real estate is tremendously precious while surfing the waves, so you might want to undock the waves window, maximize it and hide any toolbars you don’t use.)
4. We must now select the waves we want to inspect. In “sim” window, click on testbench, then in “Objects” window select all signals, right click on one, and select *Add to→Wave→Selected Signals.* You can leave the simulation time in the toolbar at “100 ps”, but normally you would have to increase this.
5. Press Run (icon with a page and a down arrow next to it, just to the right of 100 ps) in the same toolbar and a bunch of signals should appear in the waves window.
6. Zoom in and confirm that the program\_counter is operating correctly.

## ModelSim Wave Tips

...that can help you work your way through tangles of signals

Suppose that, in the middle of a simulation, you modify your code and want to simulate again without losing your settings. Go to the “Library” window in ModelSim, select the files you modified, right-click on them and click Recompile. Then on the “Waves” window, click Restart on the toolbar and click OK on the dialog box that appears. ModelSim is now ready to simulate from t = 0.

To change values to decimal, hex or other base: select some signals on the left pane, right-click on them and select a radix under Radix. Note the difference between unsigned and decimal.

You may change the scale of the time axis by right-clicking on it and selecting Grid & Timeline Properties....

You may separate your signals in categories with dividers. That’s useful when you have tens of signals. Right-click on the left pane and select Add→New Divider. Then, drag signals around.

Zoom in and out quickly by holding Ctrl and scrolling with the mouse wheel. Zoom across the whole simulation by pressing the blue magnifying glass on the toolbar. Zoom across a particular timespan by holding Ctrl and dragging with the left mouse button.

Find signal transitions and edges using the “Wave Cursor” toolbar. Find particular signal values by doing Edit→Signal Search....

“Bookmark” particular events by using multiple cursors. Also, the cursor snaps on a vertical edge or a transition if you click a few pixels to its right.