ENGN 1630 – Fall 2019

Simulating XC9572XLs on the ENGN1630 CPLD-II Board Using Xilinx ISim

You will use the Xilinx ISim simulation software for the required timing simulation of the XC9572XL CPLD programmable logic chips that you use for the early labs. At your option, you may also do functional simulations after you define the operations with Verilog but *before* you try to map and route the design. The difference between timing and functional simulations is the estimation of delays caused by the gates in the device that implements your design. As such, timing simulation is done only after the circuit itself has been put into your CPLD or FPGA. Timing simulation requires data back from the design tools with the internal layout of the design and with the characteristic delays of the internal gates. These data come in the form of two files generated by the Xilinx place and route tool. The files are found in the `\netgen\fit` or `\netgen\par` directory. They are called `<your-project-name>_timesim.v` and `<your-project-name>_timesim.sdf`. (The .v file has the netlist in structural Verilog format while the .sdf file has the timing information in an industry-standard format. Take a look at them in Wordpad just to see what they look like. If you cannot find these files, it may mean you have not yet implemented your design.)

You may also simulate Labs B and C that use Xilinx FPGAs. The PROMs and SRAM realized in the Spartan FPGAs simulate in exactly the same way as the CPLD simulations. This hand-out discusses how to do functional simulations for an entire project that is written just in Verilog modules. It also covers timing simulation whether the source code is strictly Verilog modules or is a mix of a schematic wrapper with Verilog.

To set up a functional simulation (behavioral simulation):

1. Open the ISE Project Navigator and create a new project, or open an existing project. Set the Design Properties with “ISim (VHDL/Verilog)” as the Simulator. To set the simulator for an existing project, you can use the “Project” pull-down menu, or right-click on any of the objects in the “Hierarchy” view and select “Project Properties.”

2. You cannot simulate a project module or modules if there are syntax errors or improper logic. You can do functional simulation after synthesizing your design, that is, after XST runs. Timing simulation cannot be done if the system does not map and route to completion. Because your systems are very small, we recommend you use the usual procedures to implement your design all the way to a bit file for programming. Once you have done this for a Verilog module or design, “mymodule.v,” that you now want to simulate, right-click on the top module .v file in the “Hierarchy” pane and choose “New Source” or use the menu option “Project/New Source”. Select Verilog Test Fixture (not Verilog Module) and give your file a name such as "testbench_mymodule." Click Next, and you'll be prompted to associate the file with a module; choose “mymodule,” click “Next,”

then click “Finish.” The file will be added to your project.

3. ISE creates a skeleton test fixture (known as a test bench) for you, using Verilog language constructs. In other words, the test bench is itself a module, which instantiates your design, stimulates its inputs, and observes its outputs. The mymodule module is instantiated as the "unit under test" (uut). The inputs to the module are registers ("reg") because they are assigned in a procedural block. (The initial block between "begin" and "end" is an example of a procedural block.) The outputs from the module are wires. [Test benches can be extremely powerful and can simplify testing very complex systems. A good introduction to some of their features is at the following link and is also available on the class website: https://people.ece.cornell.edu/land/courses/ece5760/Verilog/LatticeTestbenchPrimer.pdf.]

4. The simulator cannot simulate something whose starting state is unknown. The simulation begins with an “initial” block at the "initial begin" line. All input values are initialized there by default. Your code to manipulate individual signals goes under the "Add stimulus here" line. In this “initialize” section, it is a good idea to reset all registers in the module so that the simulation starts from a known state.

5. It is also necessary to initialize some or all register variables *in your design code*. For example, any state register or counter needs to have initialization in an initial block within the Verilog module that defines it. (An “initial” block has no effect on your hardware, is ignored in synthesis and cannot be counted on if the initial condition is critical to the operation you intend. It is strictly for this timing simulation.)

6. The module is tested by assigning different values to the inputs in the test bench, then running ISim to observe the outputs. Between assignment statements, delays are inserted using “*#n*,” where *n* is the number of nanoseconds of delay. You can add as many assignment statements as you want after that “Add stimulus here” comment. For example, the following code tests what happens when each of the inputs *a*, *b*, and *cin* is high separately, with those inputs changing every 25 ns after the 100-ns reset interval is completed:

```
// Add stimulus here
// t = 100 ns at this point
#25;    a = 1'b1;
// now t = 125 ns
#25;    a = 1'b0;   b = 1'b1;
// now t = 150 ns
#25;    b = 1'b0;   cin = 1'b1;
```

As another example, the following code defines a clock signal with a period of 100 ns, that is, with transitions every 50 ns:

```
//Define a clock input
```

```
always begin
#50 clk = ~clk;
end
```

This code is an always block and cannot be within the initial block. Place it right after the "end" statement of the test bench initial block. Assuming "clk" is initialized (whether to zero or one does not matter), this code produces positive-going transitions every 100 ns in a 50% duty cycle signal for as long as the simulation runs.

7. Near the top of the "Design" window there are two radio buttons – select the one marked: "Simulation." To run the simulation in ISim, click on the test fixture module in the "Hierarchy" pane to highlight it, expand the "ISim Simulator" option in the Processes window, and double-click "Simulate Behavioral Model." ISim will "elaborate" the design, then open it and run the test code in your test bench file, displaying the results in a waveform window "Default.wcfg," along with other information about the instances and processes in your simulation.

8. Usually you show at least all the signals that go to actual pins on your timing diagram. To make your display easier to read, reorder the display so that inputs are at the top and so that related signals that can form buses, *e.g.*, the state bits in Lab 5, are together in descending order. If those signals were defined as buses in the source code, they will already be grouped for display, but can be broken out individually by clicking to the left of the bus name. It is generally good practice to have clocks at the top, other inputs in the middle and outputs toward the bottom. Within inputs and outputs try to have signal information flow from top to bottom. Thus, in Lab 5, state bits should be above display segment lines. (The signal list on the structures tab usually has both individual signals and a single line for the whole bus. The latter is easier to use.) Reordering is done by dragging and dropping signal names in the "Name" pane. In Lab 5, you would probably want your state bits to be expressed as a bus with hexadecimal labeling. The radix for signal display can be changed by right-clicking on values in the "Value" pane.

9. The series of button controls across the top of the "Default.wcfg" window is largely self-explanatory. Start by "zooming to full view" to see the entire simulation time period (1 μs by default). Enter a time increment in the small window in order to continue the simulation for a specified time. Drag the yellow slider to a time of interest; use the button controls to set markers, advance between signal transitions, measure time intervals, etc. However, note that since the simulation at this stage is purely functional, there will be zero delay from, for example, the clock rising edge to latch outputs being clocked by that signal. The simulation can be restarted after changes to the display, but if design or test bench changes are needed, close the ISim window and start again.

10. You may want to observe internal signals that are not inputs to or outputs from the module. On the left side of the workspace window, click in the "Instances and

Processes" pane to expand your “test bench” object, and then expand the “uut” to see all of the internal lines in the module. Right-click or drag-and-drop to add an internal line to the waveform window. You will not see any new waveforms though until you continue the simulation for some additional time, or restart or relaunch the process.

11. When you close Isim, you will have a chance to save the display window as a “.wcfg” file. Take advantage of this to save the work of rearranging the window when you go back to do the final timing simulation.

To set up a full timing simulation:

1. When you process your .v and .ucf files to make the programming files for any Xilinx device, including CPLDs, be sure you also run the process to “Generate Post-Fit Simulation Model.” After you have the programming file ready, go to the Design/Processes pane and expand the “Optional Implementation Processes” until you see the “Generate Post-…” choice. Run that process by double clicking it. This process makes two files, one with the structure as it fits in the device and one with the timing information.

2. Near the top of the “Design” window there are radio buttons to select “Implementation” or “Simulation.” If “Simulation” is selected, just below that there will be a drop-down menu to select either “Behavioral” or “Post-Fit.” Now if you select “Post-Fit,” the “Process” window should show the app for “Simulate Post-Fit Model.” Double-clicking “Simulate Post-fit Model” runs the same simulation you have set up previously, this time using the correct gate delays from the .sdf file.

3. When the timing simulation opens, you will not see the same signals or signal order as in the behavioral simulation. Open the .wcfg file you saved to retrieve that arrangement. You will probably need to relaunch the simulation to see all results from all signals.

4. By default, the delays used by ISim for the various gates and flip-flops of a CPLD are typical rather than worst case. Expand the simulation traces to show the required propagation delays and timing relations for whichever lab you are simulating. Violations of setup and hold times will appear as red traces marked with X’s for values. (Although it is optional, you might find it interesting to see how violations of setup or hold appear. Increase the clock frequency until you see at least one red trace.)

(Although it is good practice to run timing simulations for both MAX and MIN conditions, but ISim does not offer those tests for CPLD simulation, only for FPGA simulation. Under MAX conditions, the gate delays represent the worst-case operating conditions of the device, defined as the maximum operating temperature, the minimum voltage, and the worst case process variations. Under these conditions, the data paths of the device have the maximum delay possible. At the same time, it is assumed that the clock path delays are the minimum

possible relative to the data path delays, so that this situation reasonably validates adequate setup times for the device — data signals will arrive late and clock signals will arrive early. Conversely, under MIN conditions, the gate delays represent the best case operating conditions for the device, defined as the minimum operating temperature, the maximum voltage, and the best case process variations. Under these conditions, the data paths of the device have the minimum delay possible. At the same time, the clock path delays are assumed to be the maximum possible relative to the data path delays, so that this situation reasonably validates adequate hold times for the device — data signals will arrive early and clock signals will arrive late. Right-click on the "Simulate Post-Fit Model" process and select "Process Properties." Then look for pull-down menu "Delay Values to be Read from SDF" and select either "Setup time" or "Hold time." If this feature is implemented, that is, for FPGAs, then re-running the simulation would show any logic errors that result from these *parametric* variations. These ISim simulator options have no apparent effect on CPLD simulations.)