UNIVERSITY OF CALIFORNIA AT BERKELEY
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Lab 4
Verilog Simulation Mapping

1 Motivation

In this lab you will learn how to use a hardware description language (verilog) to create a design. You will explore different levels of implementation including using Xilinx specific blocks and generic behavioral verilog. In addition to designing you will learn how to simulate verilog using a standard verilog simulator.

2 Introduction

Modern day circuit design is done mainly using a hardware description language (HDL.) Two popular HDLs are verilog and VHDL. In this class we will use verilog. Verilog can be written using either structural or behavioral descriptions. The code can either use portable standard verilog or manufacturer specific components such as Xilinx primitive gates. The differences are highlighted in the following table:

	Structural	Behavioral
Low	Xilinx primitive gates, Xilinx primitive FFs	Dataflow logic (boolean expressions), Implied FFs
High	Xilinx library adder, Xilinx library register	Implied adder (+ operator on bit-vector), Implied register

Verilog code can then be run through sophisticated CAD tools such as a logic synthesis tool which will generate the actual low-level gates. In the case of an FPGA the verilog will create a netlist of LUTS, FFs and CLBs. In the case of an ASIC the result may be a netlist of transistors.

A simulation can be run on the functional verilog code or after synthesis on the generated netlist. The functional simulation will verify the functionality of the verilog code but will not include any realistic timing information such as gate delays. After the code is run through a logic synthesis tool the timing simulation run on the generated netlist will include timing statistics yielding more accurate results.

3 Prelab

WARNING: ***You will not finish this lab during the lab section unless you do the Prelab beforehand!***

1. Read and understand the entire lab handout.
2. Read the *FDCE* handout. You will need to know how to instantiate the module in your design.
3. Glance through the *LogiBLOX* documentation linked from the website.
4. Create a verilog testbench to test your designs. A skeleton testbench is provided for you on the website.

5. Write the verilog for all four parts of the lab. You will need as much time in lab as possible to debug your code. Don't expect to be able to write and debug your code during the lab session.
6. Acquaint yourself with the *VeriLogger Pro*. Try simulating the verilog code for Parts III and IV using the testbench you wrote.

4 VeriLogger Pro

VeriLogger Pro is the verilog simulator for Windows that is included with your textbook. If you don't have a Windows machine, you may be able to find some freeware verilog simulators for Linux. For a simple tutorial on how to use *VeriLogger Pro* launch the program and go to *Help->Verilog Tutorial Basic Simulation*.

5 Procedure

Part I - Xilinx Low-Level

1. Create a new project.
2. Add the provided `top_synth.v` file to your project. Go to *Project -> Add Source Files(s)...*. The file will automatically be copied into your project directory.
3. Copy the `lab4.ucf` file from the website to your local computer.
4. Press the little white HDL rectangle in the *Design Entry* button to create a new verilog file. Use the *HDL Design Wizard*. When prompted for the HDL language, choose *Verilog*. Name the file `top.v`. Add four ports to your module:

Port Name	Width	Direction	Description
clk	1	Input	Clock signal
rst	1	Input	Reset signal for any DFFs
in	8	Input	The input to your design
out	8	Output	The output of your design

5. Create your design using only the following Xilinx primitives: `FDCE`, `AND2`, `OR3`, and`XOR3`. Do not use any `always` blocks. You may find it easier to create some simple modules and instantiate them multiple times.
6. Add your `top.v` and any other verilog files you created to the project.
7. Press the *Synthesis* button. Make sure the *Top level:* drop down box has `top_synth` selected. Select the correct *Target Device* settings: *XC4000XL*, *4010XLPC84* and then press the *Run* button to synthesize the design.
8. Press the *Implementation* button and select *Run*.
9. Go to *Implementation -> Set Constraints File(s)...*. Check the *Copy Constraints Data From:* box and select *Custom* in the drop down box. When the *Custom* dialog box appears, select the `lab4.ucf` file you copied earlier. Press *OK*, and then press *OK* again.
10. Press the *Implementation* button again and press *Yes*.
11. Simulate your design using the *Timing Simulator* inside the *Verification* button.
12. Program the FPGA the same way you did in Lab 3, and verify your design works.

Part II - Xilinx High-Level

1. Create a new project.
2. Add the provided `top_synth.v` file to your project.
3. Create a new verilog file called `top.v` with a module called `top` with the same port list as Part I.
4. Create your design using high-level Xilinx blocks. In order to generate the verilog modules for the Xilinx blocks you would like to instantiate, you have to use the LogiBLOX module generator:
   (a) In the Project Manager go to *Tools -> Design Entry -> LogiBLOX module generator...*
   (b) A *Setup* window should appear. Select the *Device Family* tab and choose `xc4000xl` in the drop down box. Then select the *Options* tab and check the *Verilog template* and *Structural Verilog netlist* boxes and click *OK*.
   (c) Select the Module Type you would like to instantiate and give it a name such as `adder8`. Make sure it has a bus width of 8.
   (d) After pressing *OK* some new files will appear in your project directory. Open the file called *modulename.vei* where *modulename* is the name you choose earlier. This file contains the port list to the module you created. You can now instantiate a module of that type in your `top.v`.
5. After you've finished your design make sure to add it to the project.
6. Press the *Synthesis* button. Make sure the *Top level:* drop down box has `top_synth` selected. Select the correct *Target Device* settings: *XC4000XL, 4010XLPC84*. Press the *Run* button to synthesize the design.
7. Press the *Implementation* button and select *Run*.
8. Load the pin constraints file as described in Part I and rerun the *Implementation* step.
9. Simulate your design using the *Timing Simulator* inside the *Verification* button.
10. Download your design to the FPGA and verify it works.

Part III - Behavioral Low-Level

For this part follow the same steps as in Part I, except use verilog primitives instead of the Xilinx low-level primitives. Do not use the verilog `+,-,/,*` functions for the adder. Create everything with boolean equations using `assign` statements with `&,|,^`, and `~`.

Before you enter the Xilinx tool flow, simulate your design using the *TestBencher Pro* verilog simulator. This way you can verify the functionality before worrying about the synthesis steps. After verifying the behavioral functionality, load your design into the Xilinx tool flow as described in Part I. You will also need to simulate your generated netlist as you did in Part I using the *Xilinx Timing Simulator*.

Part IV - Behavioral High-Level

For this part follow the same steps as in Part III, except use verilog high level behavioral code to describe the circuit.

Again, you must simulate your design both before and after synthesis. First using *TestBencher Pro* and then using the *Xilinx Timing Simulator*.

6 Acknowledgment

Original lab by J. Wawryznek, N. Zhou, and Y. Patel.

7 Questions

1. How many CLBs does each part use?

2. What is the maximum clock frequency for each part? Which one is the fastest? Why do you think that is the case?

3. Is it better to use the Xilinx LogiBLOX modules or is it better to write behavioral verilog? Why?

8 Checkoff

Name: _____

Name: _____

Section: _____

Part I - Xilinx Low-Level

1. Your verilog source. TA: _____ (10%)
2. Simulation waveforms. TA: _____ (5%)
3. Working design on Xilinx board. TA: _____ (5%)

Part II - Xilinx High-Level

1. Your verilog source. TA: _____ (10%)
2. Simulation waveforms. TA: _____ (5%)
3. Working design on Xilinx board. TA: _____ (5%)

Part III - Behavioral Low-Level

1. Your verilog source. TA: _____ (10%)
2. Simulation waveforms. Behaviopwral and Timing. TA: _____ (5%)
3. Working design on Xilinx board. TA: _____ (5%)

Part IV - Behavioral High-Level

1. Your verilog source. TA: _____ (10%)
2. Simulation waveforms. Behavioral and Timing. TA: _____ (5%)
3. Working design on Xilinx board. TA: _____ (5%)

Part V - Questions

1. Questions. TA: _____ (20%)

Total

50% off if the lab is handed in 1 week late TA: _____ (-50%)

Total Score TA: _____ (100%)