Programming Assignment 1: An Introduction to Verilog

CS141 Spring 2019

Due February 8 at 5pm

Assignment

This assignment is meant to make sure that your development environment is working properly and to get you thinking about boolean logic, Verilog, and the process of digital logic design on an FPGA.

Take a look at the structural verilog guide on Canvas as well as the Verilog cheat sheet in the `guides` folder for help getting started with Verilog.

The `main.v` file defines a module with an 8-bit input and an 8-bit output. We have set it up so that each bit of the input signal corresponds to one of the 8 switches on the FPGA and each bit of the output will control the corresponding LED on the FPGA.

Your assignment is to use `assign` statements or module instantiation in the main module to achieve the following:

1: A simple gate

Turn on `led[0]` only when `switch[0]` and `switch[1]` are both on.

2: Implication

Logical implication (often written as $a \rightarrow b$) has the meaning "if a, then b." That is, if a is true then b is true, such that $a \rightarrow b$ is only false when a is true and b is false. This function can be expressed with the following truth table.

a	b	$a \rightarrow b$
0	0	1
0	1	1
1	0	0
1	1	1

Implication is useful for proving mathematical theorems but we will implement it on the FPGA. Please implement the following:

`led[1] = switch[2]` $\rightarrow$ `switch[3]`.

3: Evens and odds

Turn on `led[2]` only when an even number of switches are on.

> Hint: You may want to consider the case with only 2 switches instead of 8 first, and make a truth table. See if you recognize the function.

4: One or the other

If `switch[4]` is on then set `led[3] = led[0]`. Otherwise set `led[3] = led[1]`.

5: Testbench

To complete this assignment you must write a testbench that tests your implementation of the previous parts. We have started it for you in `testbench.v` by making a test for part 1. Your job is to write tests for parts 2, 3, and 4. The test should display an error if something is incorrect and increment the error counter.

Since this assignment only asks you to write simple combinational logic, writing the test is a little repetitive but it's still useful to get more familiar with how simulation works with Verilog.

6: Synthesis

Make sure that when you load the program onto the FPGA board everything is working correctly. If you don't know how to synthesize your project in hardware, see the Xilinx guide on the course website.

Deliverable

Submit the following on Canvas:

1. Your Xilinx project containing
   - A `main.v` file where the LEDs are assigned according to the logic described above and `led[7:4]` are unused and should be kept off.
   - A `testbench.v` file that tests the correctness of all the parts for every possible switch assignment. You may test using logical functions or with a truth table.

Important: The full Xilinx project should be zipped and uploaded after cleaning up the project files. Please see the submission guidelines for details on how to prepare your project for submission. The guidelines are on Canvas under `Files->programming assignments->guides`

In addition, you should schedule a slot for a brief demo so you can show us that your FPGA synthesis is working and so that we know your development environment is good to go. Demos will happen on Friday the 8th, and the signup sheet will be posted in an announcement on the course website.

If you have any questions, please come to office hours or post on Piazza.