Elec/Comp 326
Fall 2013
Homework Set 3
Verilog Implementation

Overview

This assignment deals with implementing (in Verilog) two combinational logic modules that implement an ALU and an Instruction Decoder respectively. These modules will be use in your final project, so take care to implement it carefully and verify its correctness. Not only will you loose credit on the homework, but debugging your project with buggy component modules promises to be a real nightmare.

1 Preliminaries and Logistics

A brief description (relevant to this assignment) of the processor is available in the accompanying document: **Notes on Processor Instruction Set**. Read and familiarize yourself with it.

- On CLEAR, create subdirectory **hwk3** in your **elec326** subdirectory and connect to it.
- Copy all the files needed for the homework from their home position using the command:

  cp ~pjv/326/hwk3/* .

  Do not forget the "dot" at the end of the command.
- Check that all the files that have been copied by executing the command **ls -l**. You should see several files:
  - Verilog source files skeletons of the modules you will write: **alu.v**, **decoder.v**
  - Verilog source files of testbench modules: **alu_testbench.v**, **decoder_testbench.v**
  - File **alias.txt**

The text file **alias** shows the path name for the Verilog compiler: **~pjv/326/Verilog/bin/iverilog**. To save time and effort it is worthwhile to set up an alias, like **iv** for instance, to avoid entering the full name each time. You can then invoke the Verilog compiler as: **iv file1.v** to compile the source file **file1.v**. Every time you login to CLEAR, you can copy-and-paste the line in the **alias** file to the terminal — **alias iv ' ~pjv/326/Verilog/bin/iverilog'**. Or you can put it permanently in your **.bashrc** file.

2 Assignment Details

- Begin with **alu.v**. It describes the specification of the module that implements the ALU for the processor being designed. Study the input and output interface for the module and the actions required.
- Study the module **alu_testbench.v** and understand how it exercises the module in **alu.v**.
- Compile the two files using **iv alu.v alu_testbench.v**. It should create a file called **a.out**. Execute the file using the command: **./a.out**, observe the output, and relate it back to the source files.
- Code a Verilog implementation of the ALU in the file **alu.v**. A straightforward behavioral implementation is sufficient. Compile **alu.v** in isolation till all compile errors have been removed. Then compile it along with **alu_testbench.v** and verify the output.
- Change the test bench module and apply different inputs that test the implementation of your module to your satisfaction.
- Now **repeat the above steps** for the second module **decoder.v** along with its test bench module **decoder_testbench.v**.

Submission Details: Will be announced. The default will be similar to labs 1 and 2. In case we require this assignment to be submitted explicitly, we will announce it using email and Owlspace.

Please start early. When you are under time pressure, logistical issues, compile errors, and run-time weirdness will magnify your frustration. It all works, but only when you approach it in a relaxed manner.

Due Date: 11:59 pm, Friday, November 1
GOOD LUCK