CS203: Lab 2

Designing 3 input XOR gate in Verilog

Objective: Understand basic syntax of Verilog
Language: Verilog
Points: 20
Release date: 8 September 2020
Deadline: 21 September 2020

Detailed problem statement:

1. Module xor2: Create two input xor2 where you will write verilog code using primitive gates : AND, OR, NOT. Follow structural design syntax. Structural design would be similar to figure. You need to define all intermediate signals as wires.

   Module definition:
   ```
   module xor2(out, in1, in2);
   ```

2. Module xor3: Use xor2 module in designing 3 input xor3 module in structural design syntax.

   Module definition:
   ```
   module xor3(out, in1, in2, in3)
   ```

3. Module xor3_b: Model 3 input XOR gate in xor3_b.v using dataflow statements and bitwise logical operators &, I and ~ (that is, bitwise and, or and not).

   Module definition:
   ```
   module xor3_b(out, in1, in2, in3)
   ```

4. Create testbench.v for xor3 and xor3_b. Instantiate both xor3 as well as xor3_b in the same testbench. Give all possible stimulus values, print them on screen using $display. Dump the vcd file using $dumpvars and observe the output in waveform viewer.

Logistics:

- Module definitions should be the same as specified in the problem statement.
- xor2, xor3, and xor3_b modules should be in a single file named **`lab2_xor.v`**
- **`testbench.v`** will be a separate file containing testbench.

- Zip both the files into “Lab2_<entry_number>.zip” and submit on moodle. Use only zip for compression, do not use rar, tgz, bgz or any other compression.

Evaluation:

- Lab will be evaluated automatically using scripts, so it is important to follow the above instructions and naming conventions.
- Use only the latest version of iverilog. Using any other verilog compiler can lead to compatibility issues.
- Course policy of late submission and plagiarism, as given on course webpage would apply.

Installing iverilog on Linux box

(Following instructions given in https://github.com/steveicarus/iverilog)

1. $ git clone git://github.com/steveicarus/iverilog.git
2. $ sh autoconf.sh
3. $./configure
4. $ make
5. $make install
   Read README.txt file for more details and more options.

Running iverilog

Write your verilog and testbench in your favorite editor.

Assume you have xor2.v as module file and xor_tb.v as testbench. To compile these files

$ iverilog xor2.v xor_tb.v

The above command will generate a.out
Running a.out will execute the verilog model.
$ ./a.out

To view waveforms, you need to install gtkwave.
$ gtkwave ha.vcd

You may also use an online verilog editor/compiler. But final files should run using iverilog.

iVerilog for Windows:

I have tried installing iVerilog from prebuild binaries given by bleyer:
https://bleyer.org/icarus/

You have to use command line to compile verilog programs and run.

For example: My installation was at
C:\iverilog

All the below commands can be executed from any directory. You can create a directory for your lab assignment and do these experiments there.

For compilation, we write
c:\iverilog\bin\iverilog.exe ha.v ha_tb.v
This produced a.out file

To execute a.out file, we need to do:
c:\iverilog\bin\vvp.exe a.out

To see vcd file:
c:\iverilog\gtlwave\bin\gtkwave.exe ha.vcd

In GTKWave, on left pane, you can see module names, and signals. Select the signals and press insert to view the waveform.