ECEN 2350 – Lab1

August, 2020

Testing Icarus Verilog and GTKWave

Icarus Verilog (or iVerilog) is both a Verilog compiler and a Verilog simulator. A Verilog compiler converts your Verilog source files into a netlist file representing the circuit or digital logic behavior of your source code. iVerilog does not target a specific digital technology, but creates a generic logic representation that can be simulated to validate circuit operation.

Once a design has been successfully compiled, iVerilog can then be used to simulate the design. The output of the simulation can be a text file, or a .vcd file (value change dump). The .vcd file is a text file that can be read by graphical waveform display tools such as GTKWave to enable creation of graphical simulation output.

1. Creation of Verilog source file and Verilog testbench file

a. Create a text file using your editor of choice. This would be an excellent opportunity to install Visual Studio Code (details here). You must use an ASCII editor (not Microsoft Word or Google Docs). Name the file my_and.v

b. Type the following text into my_and.v: Note that cutting and pasting may result in control characters from this PDF document that will cause compilation to fail. Also, Verilog is a case sensitive language.

```
module my_and (in1, in2, out1);
input in1, in2;
output out1;

assign out1 = in1 & in2;

endmodule
```

The above code implements a simple AND gate. We will study how to construct Verilog circuit descriptions in class. A quick explanation is that every Verilog module in wrapped by `module / endmodule`. The `module` statement contains the name of your component and a list of all the input and output ports. The `input` and `output` statements simply identify the signal direction. The `assign` statement creates the actual logic, assigning the logical AND of the two inputs to the output.

c. Open a command or terminal window, switch to the directory containing my_and.v, and type this command: iverilog my_and.v . If you are using Visual Studio Code (VSC), you can open a terminal window in VSC and run from there. If you see an error stating that iVerilog is not found, it is likely that you need to add iVerilog to the Windows PATH variable, or add a symbolic link for Linux, or the equivalent for IOS.

a. If you are quickly returned to the command prompt with no additional output, the compilation worked. Note that the file a.out was generated, this is the output of the Verilog compilation step. You can force iVerilog to rename the output file by adding -o <desired output filename> to the iVerilog command.
b. If your input file contains errors, some cryptic message will be output to the command window. Correct any errors until compilation succeeds.
c. Keep the command/terminal window open.

To simulate a design in Verilog, you need to create a file that contains language constructs specific to simulation. We call this a testbench file (you may see test harness used as well). A testbench file does not compile into logic, but is used to drive signals into the inputs of your design (in1 and in2 in the my_and design), and to capture output created by your design (out1 in the my_and module).

d. Create a testbench file (again, must be an ASCII file). Name the file tb.v, with the following text. Note that the very first character is a back slanted apostrophe, not a single quote. This character is typically on the ~ key to the left of the 1 key. **In the final $display line, replace the output text shown with your name.**

```
`timescale 1 ns / 100 ps
module tb();

reg a, b;
wire c;

my_and U1 (.in1(a), .in2(b), .out1(c));

initial
    begin
        $dumpfile("output.vcd");
        $dumpvars;
        $display("Starting simulation");
              a = 0;
              b = 0;
        #10   a = 1;
        #10   b = 1;
        #10   a = 0;
        #10   $display("Simulation ended.");
              $display("<Put your name here>");
        $finish;
     end

initial
   $monitor($time, "  a = %b,  b = %b,  c = %b", a, b, c);
endmodule
```

The testbench file has a lot more going on, but here is a quick explanation:

1. The `timescale direction defines the units of time in the simulation. The first time value is the unit of simulation time, while the second time value defines how a non-integer time value or calculation will be rounded. In tb.v, time values are interpreted as delay values x 1 nanosecond, with rounding done to 100 ps or 0.1 ns precision.
2. Note the module / endmodule statements. Also note that a testbench has no input or output ports (the port list is empty).
3. The module my_and is instantiated in the testbench. Another way to say this is that you are testing one instance or one occurrence of your my_and module. It is possible to create as many instances of a module as is desired in a Verilog project. In this case, we assign a signal or variable to drive in1 (a), in2 (b), and to capture the value of out1 (c).
4. Any code inside an initial block will be executed in order, and executed only one time. In this first initial block, we define an output file ($dumpfile) to capture the simulation output for later display in GTKWave. We tell the simulator to capture all signals ($dumpvars), and then display a user define message ($display). Next, values are assigned that drive the my_and inputs. The syntax #10 means to delay 10 time units before executing the next line. $finish instructs the simulation to end.
5. The $monitor system task (which must be in its own initial block) will output the formatted string containing the signal values whenever one of the signals changes value. This output is directed to STDOUT (normally your terminal or command window).

2. Simulate the my_and design

a. In the command window, type these commands:
   i. iverilog tb.v top.v
   ii. vvp a.out > out.txt (simulation output will be written to file)
   iii. vvp a.out (simulation output will be written to console)
   iv. If you renamed the a.out file by using the -o option in iVerilog, use your new filename in place of a.out.
b. As before, getting a command prompt with no output signifies success. If you see errors, correct them until simulation succeeds.
c. Notice that two new files are created, out.txt and output.vcd.
d. Open out.txt, you should see this:

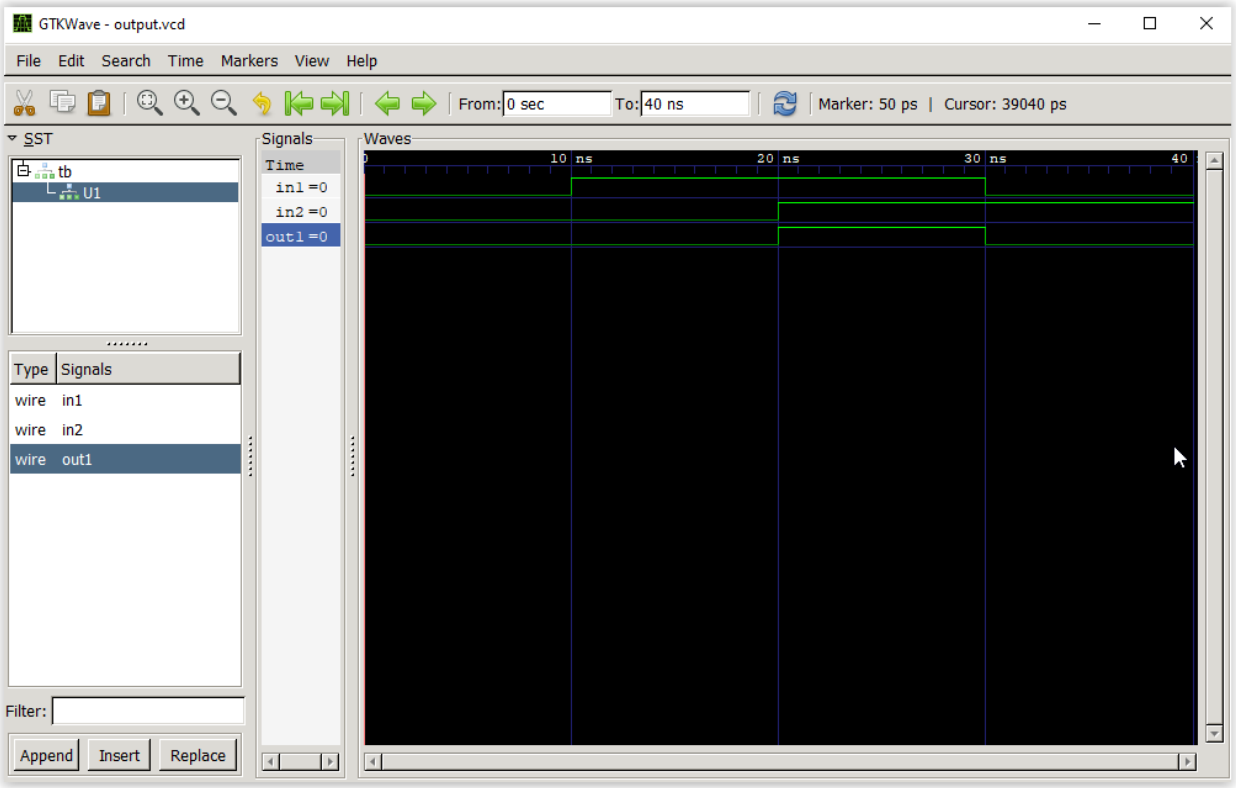
```
Starting simulation
                   0  a = 0,  b = 0,  c = 0
                  10  a = 1,  b = 0,  c = 0
                  20  a = 1,  b = 1,  c = 1
                  30  a = 0,  b = 1,  c = 0
Simulation ended.
<your name here>
```

The first value is a timestamp, followed by input values and the output value.

e. Take a screen capture of the terminal / console window showing the simulation output. You will need the screen capture to complete the Lab1 assignment.

3. Display simulation output in GTKWave

a. At the command prompt type this command: gtkwave output.vcd
b. The GTKWave window will open. In the upper left pane you will see the name of the testbench module (tb). Clicking the plus sign will show you the instance name of your module under test, U1.
c. Click on U1, then select each of the signals in1, in2, and out1 and click the Insert button. This will move the signals into the waveform viewer area. If necessary click the Zoom Fit button. You should see this



d. Take a screen capture of the GTKWave display above, you will need the screen capture to complete the Lab1 assignment.

4.

GTKWave is a waveform viewer, a tool that can read .vcd files and convert that information into a graphical view. If you refer back to tb.v, you will see that in1 and in2 were assigned the value of 0 at t = 0. At t = 10ns, in1 was set to 1, at t = 20ns in2 was set to 1, and at t = 30ns, both inputs were assigned the value of 0. Notice that the value of out1 is the logical AND of the two inputs. Finally, the simulation ends at t = 40ns.

Using GTKWave to simulate an AND gate is the C programming equivalent of Hello World, trivial but demonstrating functionality. As your designs become more complex, use of simulation will become more important. You can choose whether a purely text based simulation output is sufficient, or if a graphical view is more helpful, both options are available.