Icarus Verilog and GTKWave Installation and User Guide

The ELEC0010 lab work involves the use of the SystemVerilog hardware description language to design and simulate the operation of digital systems. Use the open-source apps **Icarus Verilog** to compile and simulate your designs, and **GTKWave** to analyze the simulated signal waveforms. This document provides instructions on downloading, installing and using these tools. Instructions are also given on installing and using the **Visual Studio Code** app, which you can use to edit the SystemVerilog code and enter commands for Icarus Verilog and GTKWave.

View the accompanying video on the ELEC0010 Moodle page, demonstrating the installation and use of these tools.

Contents

1. Installing Icarus Verilog and GTKWave

The ELEC0010 lab work involves the use of the SystemVerilog hardware description language to design and simulate the operation of digital systems. Use the open-source apps **Icarus Verilog** to compile and simulate your designs, and **GTKWave** to analyze the simulated signal waveforms. This section provides instructions on downloading and installing these apps.

For Windows

- From the website https://bleyer.org/icarus/ click on the most recent version:
  - iverilog-v11-20210204-x64_setup.exe
- Click **Open** to install
- Accept the license agreement
- Choose the destination folder in which to install Icarus Verilog (you can leave this as the suggested default folder)
- Choose to carry out a full installation by leaving both 'Install MinGW Dependences (DLL libraries)' and 'Install GTKWave' checked, and click **Next**
- Click **Next** to select the default Start Menu folder
- Select the additional tasks –

  Create a desktop shortcut

  Add executable folders to the user PATH
- Click **Next**
- The program will now start installing
- Click on **Finish**

For macOS

- Download and install the Homebrew package manager:

  ```
  /usr/bin/ruby -e "$(curl -fsSL
  https://raw.githubusercontent.com/Homebrew/install/master/install)"
  ```

  (This will not work on macOS 10.15 Catalina. If you're using Catalina, visit https://brew.sh)
- Then install the following software:

  ```
  brew install icarus-verilog
  brew cask install scansion
  ```
- When running **Scansion** for the first time, you will need to go to your Applications folder, secondary/right-click it, then press **Open**. This is because of macOS's security features.

For Linux

- Use your package manager, on Ubuntu for example:

  sudo apt-get install -y iverilog gtkwave

2. Installing Visual Studio Code

While a standard text editor can be used to write and save the SystemVerilog code, and a command line tool (such as Command Prompt in Windows) can be used to operate Icarus Verilog and GTKWave, it is more convenient to use a source code editor. If you don't already have a source code editor installed on your computer, the **Visual Studio Code** app is recommended. This section provides instructions on downloading and installing Visual Studio Code, and installing an extension for SystemVerilog coding.

Installing Visual Studio Code

- From the website https://code.visualstudio.com/download download the version of Visual Studio Code suitable for your operating systems (Windows, Mac or Linux).
- Accept the license agreement
- Select the destination location (e.g., use the suggested folder)
- Click **Next**
- Select the Start Menu folder
- Choose 'Create a desktop icon'
- Click **Next**
- Select **Install**
- Select **Finish**
- Visual Studio Code will now open. Maximise the window.

Installing a SystemVerilog extension

- Installing a SystemVerilog extension in Visual Studio Code will be helpful when writing your code, as it recognizes and highlights SystemVerilog keywords.
- Select the **Extension icon** on the left hand side
- In the search box, type **SystemVerilog**
- From the list of extensions, choose 'Verilog-HDL/SystemVerilog/Bluespec SystemVerilog'
- This extension will then be installed

Open a new terminal

- From the toolbar at the top of the Visual Studio Code window, select **Terminal > New Terminal**
- A terminal will appear in the lower half of the window, in which you can enter the commands to compile and simulate your SystemVerilog code using Icaurus Verilog, and to open the GTKWave signal waveform analyzer.
- You are now ready to start writing SystemVerilog modules.

3. Using Icarus Verilog

The SystemVerilog code is compiled and simulated by Icarus Verilog, using commands typed into the Terminal window in the Visual Studio Code app. The following three commands will be used.

To compile the testbench code, together with all the modules being tested:

iverilog -g 2012 -o df_<testbench module name>.vvp <testbench module name>.sv

To perform the simulation using the vvp file created by the compilation:

vvp <testbench module name>.vvp

To open the GTKWave signal waveform analyzer:

gtkwave

4. Icarus Verilog design example – a 2-input AND gate

To get started using Icarus Verilog and GTKWave, complete the demo exercise designing and simulating a 2-input AND gate, following the steps below:

- Use your computer's file manager application to create a new folder in which to store your SystemVerilog files, and other files used in the design and simulations.
- In Visual Studio Code, from the toolbar at the top, select **File > New File**.
- Left-click on **Select a language**. From the drop-down menu, select **System Verilog**
- Type in your module code. For the 2-input AND gate, this is:

```
module and2 (input logic a, b,
             output logic y);
assign y = a & b;
endmodule
```

- Select **File > Save As…**, navigate to the folder you created and save the file with the name **and2.sv**. Note that the filename must match the module name (in this case, and2), and the file extension must be .sv (referring to the fact that it is SystemVerilog code).

- Next, create the testbench. This will also be a text file you create containing SystemVerilog code, and we'll choose the module name 'and2_tb'. From the toolbar, select **File > New File**. Left-click on **Select a language**. From the drop-down menu, select **System Verilog**.
- Start typing your testbench module code:

```
`timescale 1ns/1ps                  // Define time units and resolution
`include "and2.sv"                  // Include all modules being used in this module

module and2_tb;

logic a, b, y;                      // Declare the variables being used by the testbench

and2 dut (a, b, y);                 // Instantiate the module under test. dut (device
                                    // under test) is the name we have chosen

initial begin                       // Single pass behaviour which starts at time 0 ns
  $dumpfile("and2_tb.vcd");         // Dump variable changes in the vcd file
  $dumpvars(0, and2_tb);            // Specifies which variables to dump in the vcd file

  a = 0; b = 0; #20;                // Set variable values and set time delays
  b = 1; #20;
  a = 1; #20;
  b = 0; #20;
end

initial begin                       // Single pass behaviour which starts at time 0 ns
  $monitor("t = %3d, a = %b, b = %b, y = %b \n", $time, a, b, y);
end                                                          // Print variable
                                                             // values on screen
endmodule
```

Notes on the testbench code

- In the **`timescale** statement at the start of the testbench, the first value specifies the time units and the second is the time precision for the simulation.
- All modules being used by the testbench module must be specified using the **`include** statement. In any module, an **`include** statement must be used for every module which it makes use of (i.e., in a hierarchical design).
- The **$dumpfile** system task dumps variable changes into the Variable Dump Changes (vcd) file, which is used to plot the results.
- The system task **$dumpvars** (<level>, <module name or variables>) specifies which variables will be dumped into the vcd file. In the example above,

$dumpvars (0, and2_tb), the level 0 refers to the top level module (i.e., the testbench module), and the module name of the testbench (and2_tb) is specified.

- The system task **$monitor** prints the variable values on the screen, whenever a variable value changes. Values can be displayed in decimal (%d), binary (%b) or hexadecimal (%h).
- **$time** is the value of the simulation time.

- After entering the testbench code into Visual Studio Code, select **File > Save As…**, navigate to the folder you created and save the file with the name and2_tb.sv. Note that the filename must match the module name (in this case, and2_tb), and the file extension must be .sv (referring to the fact that it's a text file containing SystemVerilog code).
- In the Terminal window in Visual Studio Code, navigate to the folder containing your SystemVerilog module files. Use the following commands:
  - To view the files in the current folder, type **dir** (this stands for **dir**ectory)
  - To navigate to the required folder, type **cd** <folder_name> (this stands for **c**hange **d**irectory)
  - To move up one level in the hierarchy of folders, type **cd ..**
- Next, to compile all the modules, type in the Terminal window:

  iverilog -g 2012 -o and2_tb.vvp and2_tb.sv

- Correct any errors in the code reported in the Terminal window during the compilation, and, if necessary, compile again until compilation is successful (no error messages).
- Run the simulation by typing the following command into the Terminal:

  vvp and2_tb.vvp

- The simulation time and variable values generated by the simulation will be listed in the Terminal window, by the **$monitor** system task you included in the testbench:

  VCD info: dumpfile and2_tb.vcd opened for output.
  t = 0, a = 0, b = 0, y = 0
  t = 20, a = 0, b = 1, y = 0
  t = 40, a = 1, b = 1, y = 1
  t = 60, a = 1, b = 0, y = 0

- The final step is plotting the signal waveforms using the GTKWave waveform analyzer, as described in the next section.

5. Using GTKWave signal waveform analyzer

- Open the GTKWave signal waveform analyzer by typing **gtkwave** in the Visual Studio Code Terminal.
- The GTKWave analyzer will open in a new window. Maximise this window.
- From the toolbar at the top of the GTKWave window, select **File > Open New Tab**, and select the Variable Change Dump file (**and2_tb.vcd**).
- Left-click on the **and2_tb** filename on the left-hand side, and the variables a, b and y will appear on the lower left-hand side. Select all three variables, and left-click on **Append** in the bottom left.
- Click on the **Zoom Fit** button in the toolbar at the top of the window, and the a, b and y signal waveforms will appear (Figure 1).

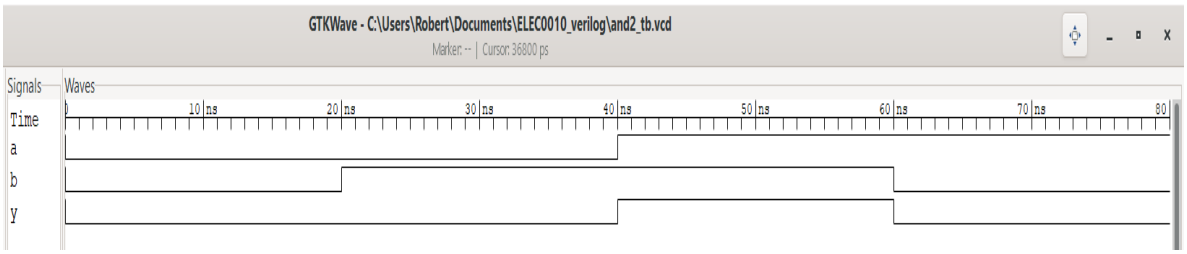

Figure 1 Signal waveforms a, b and y for the 2-input AND gate simulation, plotted using GTKWave waveform analyzer

- To include waveform plots from GTKWave in your lab report, it's recommended to use the following steps:
  1. Select **View > Use Black and White**.
  2. Remove the grid, by selecting **View** and unchecking **Show Grid**.
  3. Take a screen shot and crop the resulting png image file in PowerPoint or an image editing app.

- You must close the GTKWave window before you can continue to use the Icarus Verilog app.