Guide to Tools for ECE241/ECE253

Revision : 1.5 of September 25, 2020

This document provides information about accessing the tools used by the digital hardware courses in the *Digital and Embedded Systems Lab* (DESL) in ECE at the University of Toronto. You can access the tools remotely or install them on your own machines.

1 Remote Access

All of the tools described here are available on the ECF PCs and on the PCs in the DESL.

Instructions for ECF
`http://www-ug.eecg.toronto.edu/msl/handouts/Steps_to_connect_to_ECF_computers_remotely.pdf`.

Instructions for DESL
`http://www-ug.eecg.toronto.edu/msl/handouts/Steps_to_connect_to_DESL_computers_remotely.pdf`.

2 *logisim*

The *logisim* simulator is a nice open source tool for doing digital logic simulation. It should run on any system supporting Java.

2.1 Web Site

`http://www.cburch.com/logisim/`

2.2 Notes

1. You will need to have Java installed.

For Mac users, if you are running Catalina, you will find that *logisim* will not start. You can get around this by just clicking on the *logisim* jar file. In the directory where you untar-ed the downloaded file, the jar file can be found in:

`Logisim.app/Contents/Resources/Java/logisim.jar.`

For simplicity of finding it, you can copy it to your desktop.

2. You can learn how to use *logisim* by first going through the tutorial. After starting *logisim* go to Help>Tutorial. The user guide is also there if you want to learn more.

3 Quartus Prime Lite

You will use Intel Quartus Prime to implement Verilog designs on FPGAs. The lab material has been based on Version 18.0, so we recommend that you install that version. You will need about 14GB of disk space and 6-8GB of memory in your computer. Windows 10 and Linux are supported, but see `https://www.intel.com/content/www/us/en/programmable/support/support-resources/download/os-support.html` for details. While Quartus is supported on several flavours of Linux, *ModelSim* is supported on fewer. It can be made to run on Ubuntu 18.04, but with some amount of pain!

To install Quartus on your own machine go to `https://www.intel.com/content/www/us/en/programmable/downloads/download-center.html`.

Select Version 18.0 and click on the *Lite Edition*, which is the free version.

On the next page, to minimize what is installed, click on the *Individual Files* tab. Download *Quartus Prime*, *ModelSim-Intel FPGA Edition (includes Starter Edition)* and *Cyclone V device support*. The DE1-SoC board in our labs contains a Cyclone V. Run *QuartusLiteSetup* to do the installation.

3.1 Tutorials

The Intel University web site at `https://software.intel.com/content/www/us/en/develop/topics/fpga-academic/learn/tutorials.html` has a number of useful tutorials. You should do the Verilog version of *Introduction to Intel Quartus Prime Software (standard or lite)*. Without the board, you can still do everything up to programming, configuring and testing the design on the FPGA.

4 The *fake_fpga* GUI

Use this GUI if you do not have access to a physical DE1-SoC board.

To give as close an experience to running on a real board as possible, the *fake_fpga* was developed. The *fake_fpga* is a simulation environment GUI that can give you the experience of interacting with LEDs, displays, buttons and switches in the same way as you would on a physical DE1-SoC board. The exact same Verilog code that works on the *fake_fpga*, will also run on the DE1-SoC board. The main difference is that the circuit will run much faster in real hardware than it does in the simulated environment.

4.1 Java Version

To run *fake_fpga* you will need to have Java JDK 14 or newer to be installed. You can download JDK 14 at `https://jdk.java.net/14/`.

For installing on Windows, go to `https://java.tutorials24x7.com/blog/how-to-install-openjdk-14-on-windows`. This will make JDK 14 the default version of Java. However, please read Section 4.3 if you want to run *logisim* as well.

For installing on Linux, please search online. It will vary depending on the Linux release and version you are using.

4.2 Installing *fake_fpga*

To install *fake_fpga* go to `https://github.com/UofT-HPRC/fake_fpga` and then go to the latest release available under Releases on the right of the window. Follow the instructions there to install and test that the demo example works.

4.3 Java Conflicts

You can run *logisim* with the default Java installed with your Windows, i.e., if you do not do anything to your version of Java. To run *fake_fpga* you will need Java JDK 14 so to be able to run both, you can do this:

1. Download the `Windows/x64` zip file at `https://jdk.java.net/14/`.
2. Unzip the file to your preferred installation directory.

3. Click through the installed folder into the `bin` directory.

4. Copy the entire path in the address bar at the top including the `bin` by `Right Click > Copy address as text`.

5. Open File Explorer and navigate into the *desim_win32* folder that you downloaded for *fake_fpga* and then into the `gui` folder. There you will find a `run.bat` file.

6. Open `run.bat` to edit it.

7. Add a line at the beginning

   ```
   PATH (paste the path you copied);%PATH%
   ```

   The first line should now look something like:

   ```
   PATH C:\ ... \openjdk-14.0.2_windows-x64_bin\jdk-14.0.2\bin;%PATH%
   ```

8. You should now be able to run the GUI by clicking the `run.bat` file.

4.4 Using *fake_fpga*

Download *u_of_t_scripts.zip* and read the `README.txt` file. There are also videos at the download site that may help you.

A few things to note:

1. To make things easier to explain and minimize the need to modify too many things, the name of the top module of the design you want to run with *fake_fpga* should be `main` and for simplicity, put it in a file called `main.v`. You should start a new Quartus project with `main.v` as the top-level module. The `lab_template.v` file shows an example of what your top module should look like. This is necessary so that the test bench in `tb.v` does not need to be changed as it will instantiate a module called `main` with the port list as shown in `lab_template.v`.

   If the name of the module you want to use with *fake_fpga* is not named `main` and does not fit the port list required, you can just instantiate it within the `main` module. For example, if your top module is

   ```
   module part2(SW,LEDR);
   ```

   and it is in a file called `part2.v` then copy `lab_template.v` and name it `main.v`. Replace the `Write code in here!` comment with an instantiation of your `part` module:

```
part2 P2(SW,LEDR);
```

Copy the contents of `part2.v` and paste it after the `endmodule` of `main.v`. An alternative to pasting `part2.v` is to make sure you include `part2.v` as part of the design files of the project. If you do that, you will also have to add `vlog part2.v` in the `run_sim` script. Follow the directions in the `README.txt` that came with the `u_of_t_scripts.zip` file to compile your design in Quartus.

2. The Verilog code that you write is called the Register-Transfer Level (RTL) description of your circuit. The `compile_sim.tcl` script will create the synthesized version of your design and leave it in a file called `main.vo`, which is also a Verilog file that can be simulated. The `main.vo` file is called the synthesized netlist and is essentially the gate-level version of your circuit. Simulating the synthesized netlist is the best test of whether the Verilog code that you have written will work on the real hardware as it is possible to synthesize a circuit that behaves differently than the simulation of the original Verilog RTL code. When this happens, the most likely reason is that you wrote poor Verilog code that did not follow the rules of writing synthesizable Verilog.

3. When you start the GUI, you can leave it running between multiple tests. When you are finished a test, hit *Stop Simulation* on the GUI, and then *Reset Signals*, which sets all switches, buttons and displays back to their default starting states. Then when you start another test, it will connect to the GUI and you should see `Connected to simulator` in the *Messages* window of the GUI.

4. When you are finished and want to close the GUI, you should first close the GUI and then the command window that would have opened at the same time when you started the GUI, or if you are using Linux, close the terminal window where you started the GUI. This kills the remaining thread that is holding onto the server port.