Chapter 15O

Online Content: Programmable Logic

15O.1 Online resources

15O.1.1 Simulation and error checking

The lack of detailed error reporting in the WebFPGA IDE can be ameliorated somewhat by using EDA Playground at https://LAoE.link/EDA_Playground.html. EDA Playground provides simulation for a number of languages, but for our purposes the ability to compile Verilog and provide useful error reporting are the most valuable. In fact, it is not a bad idea to repeatedly test your Verilog code in EDA Playground until it compiles error free before copying it to the WebFPGA development environment.[1] A clean compile will not guarantee your code works as you intend, but if you can't get your code to compile you will never find out if it works.[2]

Be warned that Verilog code that runs fine in simulation may not work on an actual FPGA. For example, the simulator is perfectly happy using multiple `aways@` blocks to set the value of the same signal. In hardware, however, this is akin to connecting multiple gate or flip-flop outputs together – something we know does not work and results in an error from the WebFPGA synthesizer.

To use EDA Playground, make sure the "Testbench+Design" selection is set to "SystemVerilog/Verilog," and select "Icarus Verilog 0.10.0 11/23/14" in the "Tools & Simulators" listbox. Next, copy your Verilog code into the righthand window (design.sv) and press the "Run" button: see Fig. 15O.1.

[1] WebFPGA uses a slightly older version of Verilog so it is possible that some code using newer features may compile in EDA Playground but not in the WebFPGA IDE. The only one we are aware of is the ability to include parameters in the module definition.

[2] For some reason EDAPlayground does not like apostrophes when copied from our Notepad++ editor. If you get a syntax error it may be because the character looks correct but is not the version EDAPlayground likes. This is typically a problem in constant definitions. Delete the copied apostrophe and type a new one directly into EDAPlayground.

EDAplayground includes a waveform viewer to show the results of a testbench as a timing diagram. The "Open **EPWave** after run" checkbox under "Run Options" should be checked to open the waveform viewer after simulation.

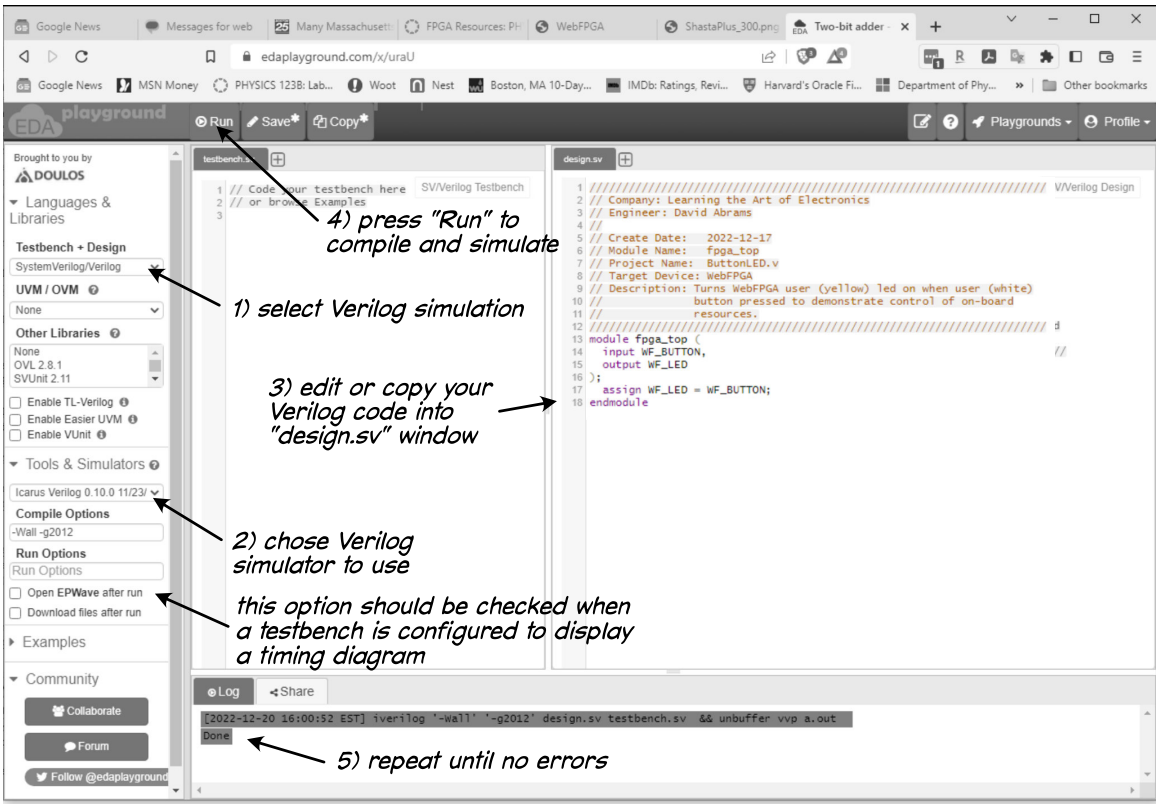

Figure 15O.1: Using EDAplayground to test Verilog code.

Documentation and tutorials are at:
https://LAoE.link/EDA_Docs.html

EDAPlayground also allows you to write a testbench program to exercise your code and display a timing diagram of the result. Here is the EDA Playground simulation of the ButtonLED design with a simple testbench that presses and releases the button: https://LAoE.link/EDA_ButtonLED.html. Press "Run" to see the simulation. Since both the button and the LED are active low, the output should follow the input. Try changing the `assign` statement to invert the assignment and try the simulation again.

Appendix E at https://LAoE.link/Testbenches.pdf offers an introduction to writing testbenches for your designs. In addition, there are many additional online resources available to get you up to speed. You may want to start here:
https://LAoE.link/Testbench_Tutorial.html

15O.1.2 Learning Verilog

HDLBits:
https://LAoE.link/HDLBits.html
The HDLBits site has about 175 Verilog problems in increasing complexity with a verification engine to check if you get the solution correct or not. It is worth working your way though the problems to learn the language and see the kind of things an HDL is used for. If you create an account and log in, the site will track your progress.

8bitworkshop:
https://LAoE.link/8bitworkshop.html
8bitworkshop has a nice interactive Verilog test environment with examples ranging from very simple to more complex. You can use it as an alternative to EDA Playground to check your Verilog code.

15O.1.3 Additional FPGA resources

NandLand:
https://LAoE.link/NandLand.html
This is a beginners introduction to Verilog.

Verilog in One Day:
https://LAoE.link/VerilogInOneDay.html
A very complete on-line Verilog tutorial.

ZipCPU:
https://LAoE.link/ZipCPU.html
This tutorial includes formal verification of HDL designs.

Chip Verify Verilog tutorial:
https://LAoE.link/ChipVerify.html
A tutorial that uses EDAplayground for examples and includes testbenches.

Introduction to FPGA - Digikey Youtube videos:
https://LAoE.link/FPGA_Videos.html
A 12-part video series on FPGA development using open source tools.

15O.2 Alternatives to the WebFPGA

With the exception of the memory block example of §18S.2, our programmable logic labs use only basic logic cells and none of the special features in the WebFPGA Lattice iCE40 (such as memory arrays, DSP blocks or specialized communication interfaces). That means that almost any FPGA breakout board with at least 24 externally accessible I/O pins should work in place of the WebFPGA.[3] Most boards will have at least one on-board user button and LED, but if not you may have to add those externally.

If you choose to use an alternative board, you will have to modify the I/O pin connections appropriately since the pin numbers of your board will not be the same. Also, many of our labs make use of the 16Mhz oscillator on the WebFPGA. Most of the alternative boards should include a clock oscillator as well, although no doubt at a different frequency. You will have to adjust your clock divider accordingly if the clock is used for precise timing.

The biggest difference will be in the development environment. At least one board below (the STEP FPGA) promises a web-based environment similar to the WebFPGA. This has the advantage of not only allowing you to starting to develop applications quickly, but also provides compatibility with almost any operating system. Other boards using either the open source IceStorm tools or the FPGA manufacturer's toolset will have a longer learning curve and may not work with every OS. In each case, the method of connecting internal signal to external pins will vary, you will have to adjust the labs to the IDE's pin connection method.

STEP FPGA ($59):
https://LAoE.link/STEP_FPGA.html
FPGA: Lattice MachXO2 LCMXO2-4000HC (4320 LUTs)
Development Tools: Cloud-based IDE, Lattice Diamond IDE
The STEP-MXO2Core FPGA development board is an excellent (albeit more expensive) alternative to the WebFPGA. The hardware includes a similarly sized Lattice FPGA, a 3.3V voltage regulator, eight user switches (four pushbutton and four DIP switches), eight user LEDs and two seven segment displays. The breakout board is 0.1 inch narrower than the WebFPGA, allowing access to an additional row of pins on the breadboard and it uses an included USB C cable to connect to your computer. Like the WebFPGA, it uses a zero setup web-based development environment, so it runs on any OS and you can begin development in a few minutes. While the STEPFPGA IDE is more full featured (it supports multiple source files and allows you to save and recall projects), it similarly does not provide schematic display of the synthesis results.

The STEPFPGA installs its flash memory as a disk drive to allow you to copy the bitstream file, alleviating the necessity for a WebUSB compatible browser. The IDE has a very nice separate page for graphically connecting header pins and internal resources to Verilog module ports: see Fig. 15O.2. The only change you should need to make from the WebF-

[3] But unless otherwise noted, we have not tested any of the boards listed below. We suggest them as possible alternatives based solely on their published specifications.

PGA physical builds is moving a few I/O connections where the STEPFPGA designates a pin as unavailable for I/O. All LAoE Verilog code should synthesize and run without any changes other than adjusting the clock where necessary (the oscillator on the STEPFPGA runs at 12MHz, not 16MHz). Our STEPFPGA came with a well written tutorial explaining each of the many sample programs included with the IDE. We have successfully tested the STEPFPGA with the Verilog examples in §15S.1.3. A version of Chapter 15L using the STEPFPGA is available at https://LAoE.link/LAoE_Chapter_15LS.pdf

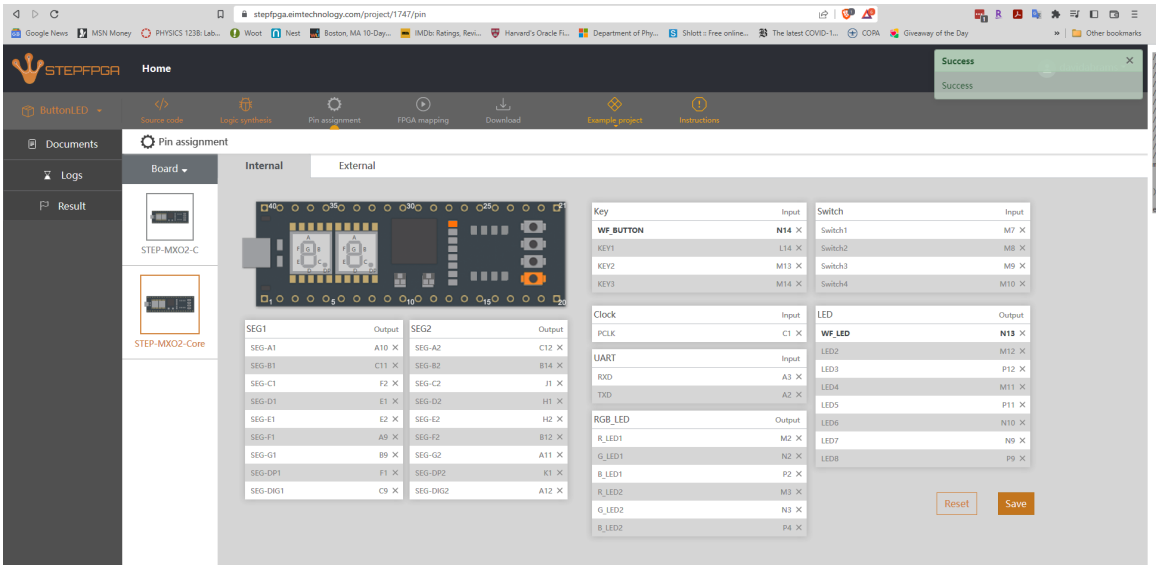

Figure 15O.2: SETFPGA IDE pin assignment page for ButtonLED example.

FireAnt ($38):
https://LAoE.link/FireAnt.html
FPGA: Efinix Trion T8F81 (7384 LEs)
Development Tools: Efinity Software IDE
The FireAnt includes a 3.3V voltage regulator, two user pushbuttons, four yellow user LEDs and a 33.33MHz oscillator. The FireAnt breakout board is 0.1 inch narrower than the WebFPGA, allowing access to an additional row of pins on the breadboard. In addition, it brings out more I/O pins to header pins and does not reserve any of them for special purposes.

Unfortunately, it uses a traditional IDE, a 880Mb download with a separate download and install of the USB drivers (on Windows – Linux is also supported), with a significant learning curve. Additional downloads are required for optional simulation and graphical display. The IDE requires entry of the IC pin numbers so you will have to refer to the Pinout Diagram in the FireAnt Quick Start Guide (https://LAoE.link/FireAnt_Pinout.html) to translate between header pins and FPGA pin numbers. We have successfully tested the FireAnt with the Verilog examples in §15S.1.3.

iceFun FPGA Board ($37):
https://LAoE.link/IceFun.html
FPGA: Lattice iCE40 HX8K (7682 LUTs)
Development Tools: IceStorm, F4PGA, Lattice iCEcube2
Includes four user pushbuttons and 32 user LEDs.

UPduino v3.1 ($30):
https://LAoE.link/UPduino.html
FPGA: Lattice iCE40UP5K (5280 LUTs)
Development Tools: IceStorm, F4PGA, Lattice iCEcube2

iCE40 Feather($60):
https://LAoE.link/iCE40-feather.html
FPGA: Lattice iCE40UP5K (5280 LUTs)
Development Tools: IceStorm, F4PGA, Lattice iCEcube2
Notes: only 20 I/O available on header pins; additional 4 I/O available on solder pads

TinyFPGA BX($38):
https://LAoE.link/TinyFPGA_BX.html
FPGA: Lattice iCE40LP8K (7680 LUTs)
Development Tools: IceStorm, F4PGA, Lattice iCEcube2
Notes: Limited I/Os available on header pins; additional I/Os available on solder pads. (The *TinyFPGA EX* with a larger FPGA and more I/O pins is in development. See https://LAoE.link/TinyFPGA_EX.html).

Please send comments or corrections to: authors@LAoE.link