Chapter 15LS

Lab: FPGA 1 using the STEPFPGA

15LS.1 Introduction to FPGA combinational logic

This online chapter is a rewrite of Chapter 15L using the STEPFPGA in place of the WebFPGA. Should you decide to try the STEPFPGA, this chapter will get you started and give you the information you need to modify the remaining FPGA labs to work with this board as well. Since the STEPFPGA uses a different model Lattice device, the EBR memory block example in §18S.2 will not work directly. You will need to consult the Lattice documentation for the Lattice MachX02-4000HC FPGA to see how memory blocks are accessed in that device if you decide to try this example.

15LS.1.1 Setting up the STEPFPGA

The STEPFPGA breakout board pairs the Lattice MachX02-4000HC (with 4320 LUTs) FPGA with 96kB of internal flash memory, a 12MHz precision oscillator, a JTAG programmer and a 3.3V voltage regulator along with two seven-segment displays, eight user-controlled LEDs, two RGB LEDs, four pushbuttons and four DIP slide switches and a USB-C connector, providing all the hardware necessary to use the FPGA. It includes 29 general purpose I/O (*GPIO*) pins available to the user: see Fig. 15LS.1.

The breakout board requires 5V power, either from the USB or directly to a header pin. We will connect the breadboard +5V supply to VBUS (pin 40) through a Schottky diode. That way, the higher voltage supply will power the board when both are connected. The 5V input is regulated down to 3.3V by the voltage regulator to drive all the components. The output of the voltage regulator is available at pin 1 labeled “3V3” on the schematic.

Start by mounting the STEPFPGA on the bottom row of the breadboard. You will leave it there even as you work on other labs that do not require the FPGA. We suggest mounting it on the bottom-left side of the breadboard so it is easy to wire to the logic slide switches, but make sure the included USB-C cable clears the pushbutton resistors so you can program the device.

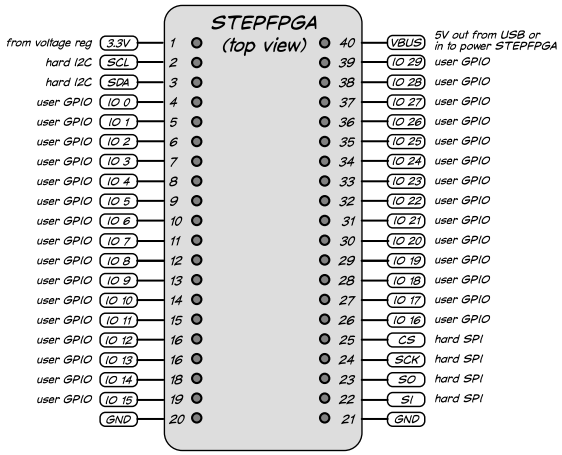

Figure 15LS.1: STEPFPGA header pin connections.

Add wires to connect the two STEPFPGA header ground pins (20 and 21) to breadboard ground and connect the breadboard $+5\mathrm{V}$ supply rail to input pin 40 on the STEPFPGA through a 1N5817 Schottky diode. Be sure to bypass the $+5\mathrm{V}$ supply to ground near the connection to the STEPFPGA. Fig. 15LS.2 shows the STEPFPGA installed with power and ground connected.

Next, you are going to connect the breadboard LOGIC SWITCHES to eight I/O pins on the STEPFPGA. Since voltages higher than 3.6V can damage 3.3V logic, check the breadboard "$+\mathrm{V}$" variable supply and make sure it measures no more than 3.6V before proceeding. We suggest anchoring the $+\mathrm{V}$ voltage adjustment knob in place with silicon adhesive or hot melt glue (both of which are easily removed later): see Fig. 15LS.3.[1] As noted in the previous chapter, we also add a fixed 3.3V supply to the breadboard: see online Chapter 14O at `https://LAoE.link/LAoE_Chapter_14O.pdf`. Also, make sure the voltage select slide switch for the LOGIC SWITCHES is set to "$+\mathrm{V}$" (3.3V). We put a small piece of tape over the upper half of this switch to keep it from being moved to the "$+5$" position.

Next, connect the eight LOGIC SWITCHES to the STEPFPGA. Switch S_1 connects to IO4 (pin 8), while switch S_8 connects to IO11 (pin 15): see Fig. 15LS.4. This completes the initial STEPFPGA installation.

15LS.1.2 Test your STEPFPGA installation

With the breadboard power off, connect a USB-C cable between your computer and the STEPFPGA. The green power LED near the USB connector should light. If it does not, check your wiring.

Open a browser and go to `https://LAoE.link/STEP_IDE.html` and click "Sign in." Enter your login information or select register to set up an account.[2] Once you are logged in, press Start and follow these steps to create the test project.

- Name the project "ButtonLED." (Note, project names cannot begin with a number.)

[1] We are being cautious because very small movements of the adjustment knob causes large changes in the $+\mathrm{V}$ output voltage. The STEPFPGA is relatively expensive and voltages greater than 3.6V may damage it.

[2] Having an account allows you to save your projects in the IDE.

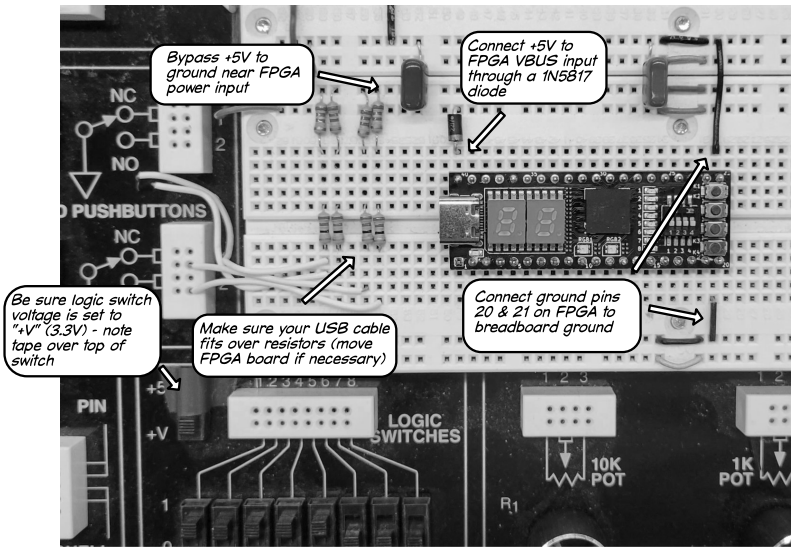

Figure 15LS.2: Installing the STEPFPGA on the breadboard.

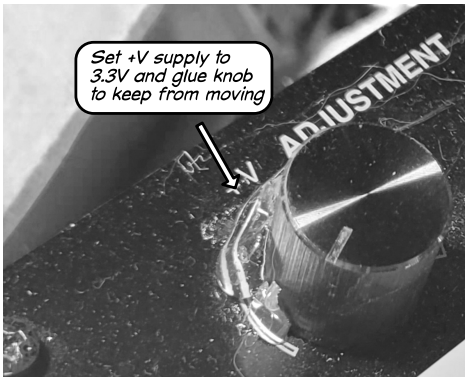

Figure 15LS.3: V+ supply adjusted to 3.3V with knob glued in position.

- Select the "STEP-MX02-Core" board.
- Project tags and Description are optional. We usually use "LAoE" and the chapter (15LS) as tags and the contents of the Verilog file header "Description:" field modified for the STEPFPGA as the description: see Fig 15LS.5.
- Click "Submit" then click on the + in a circle in the Source code tab to create the Verilog program. Select "Create a new file," name the file "ButtonLED" and copy in the following code:[3]

```
//////////////////////////////////////////////////////////////////////
// Company: Learning the Art of Electronics
// Engineer: David Abrams
//
// Create Date:   2022-12-17
// Module Name:   fpga_top
// Project Name:  ButtonLED.v
// Target Device: WebFPGA
```

[3]Code available at `https://LAoE.link/FPGA/15S_ButtonLED.v`. All the book Verilog examples, except for the memory block and open drain code, should work fine on the STEPFPGA as well, although you may have to adjust dividers for the slower 12MHz clock if timing is critical.

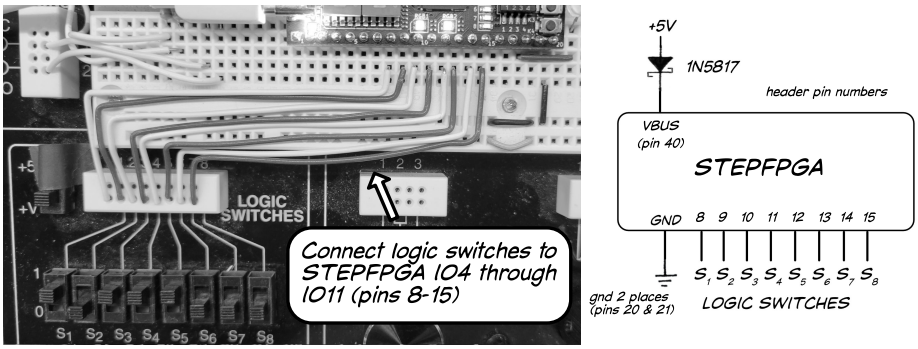

Figure 15LS.4: Initial STEPFPGA setup.

```
// Description: Turns WebFPGA user (yellow) led on when user (white)
//              button pressed to demonstrate control of on-board
//              resources.
////////////////////////////////////////////////////////////////////////
module fpga_top (
  input WF_BUTTON,
  output WF_LED
);
  assign WF_LED = WF_BUTTON;
endmodule
```

- Save your code and press "Logic synthesis." (If you do not save your code first the IDE will not synthesize it.) You should see a "Success" message in the debug log.
- Click the "Pin assignment" tab button. The STEPFPGA IDE ignores the WebFPGA pin assignments comments and uses a graphical assignment editor to connect signal names to external pins. The editor has two tabs, one for internal (on board) resources and one for external connections. For this test we are only using the "Internal" tab.
  - Click on KEY1 in the key grouping. Assign it to the WF_BUTTON signal. The upper pushbutton on the image of the STEPFPGA board should highlight to show it is now in use.
  - Click on LED1 in the LED grouping. You will notice that the dropdown no longer contains the WF_BUTTON signal, since it has been assigned to a pin. Assign the upper LED to the WF_LED signal.
  - Press Save to complete the signal assignments. (If you make a mistake, press "X" on the pin you want to unassign and it will be added back to the available signal list.)
- Press "FPGA mapping" to place and route the netlist. This creates a .jed bitstream file. When complete, select the download tab. Here you can either download the source files and the .jed file to your computer as a zip file (by pressing the upper Download choice next to the package icon) or use the lower button to download the .jed file to the FPGA. When you connect the STEPFPGA to your computer it opens as a flash memory disk. Saving the .jed file to this disk loads the bitstream into the FPGA's flash memory.
- Press the lower "Download" button to open a *Save As* dialog. Save the implement.jed file to the disk named STEPLink.[4]

[4]These instructions are based on running the IDE on a Windows computer. MacOS and Linux may see different nomenclature for disks and dialogs.

- Once the bitstream file is downloaded to the STEPLink disk, it is moved from the disk into the FPGA. You should then be able to light the top LED by pressing the top pushbutton.

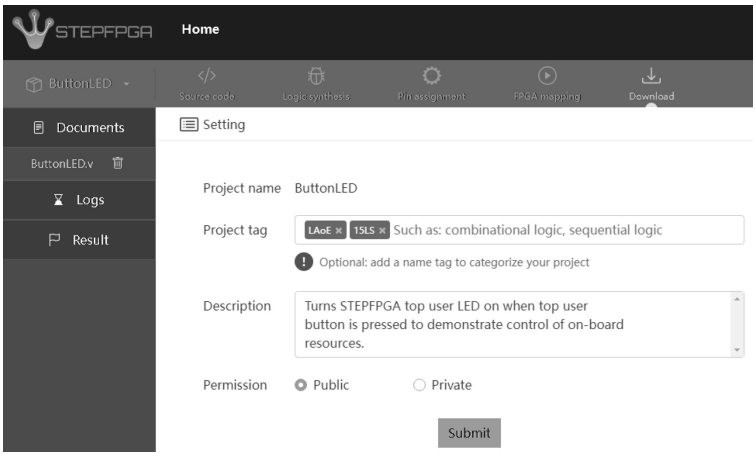

Figure 15LS.5: IDE Create project for STEPFPGA test.

You are now ready to write Verilog code and program the FPGA.

15LS.1.3 Get familiar with STEPFPGA I/O

Follow the directions in §15S.1.3 to control an external LED from one of the user pushbuttons. Fig. 15LS.6 shows how to connect the external LED to STEPFPGA IO14 (header pin 18).

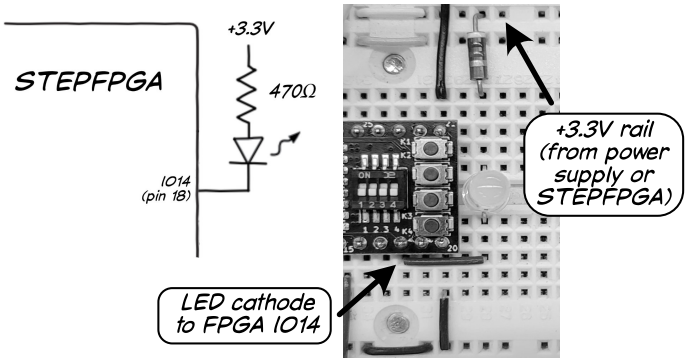

Figure 15LS.6: External LED and resistor connected between +3.3V and STEPFPGA IO14 (header pin 18).

Make sure you understand how variables in your Verilog code are mapped to external pins on the STEPFPGA in the IDE pin editor. We prefer to map external STEPFPGA pins to the same *GPIO number* (not header pin number) we used with the WebFPGA. That is why we had you connect the logic switches as we did. This makes the Verilog code self-documenting as WebFPGA @MAP_IO comments correspond to the GPIOs used on the STEPFPGA.

One change you will occasionally have to make to the code we provide is to change the clock divider where the onboard clock is used for exact timing. The STEPFPGA clock runs at 12MHz while the WebFPGA clock is 16Mhz. That means that you must decrease any clock divider by 25% (i.e. multiply the divider by 0.75) if the timing is important.

15LS.1.4 Create some simple combinational logic

Wire five unused STEPFPGA I/O pins to breadboard LOGIC INDICATORS LEDs 1 through 5. It should not matter how the two level selection switches are set, but the best choices are "+V" and "CMOS."

Write Verilog code, run synthesis and flash the STEPFPGA to implement the following combinatorial functions of logic switches S_7 and S_8 on the indicator LEDs.

LOGIC INDICATOR 1 = NOT S_7
LOGIC INDICATOR 2 = S_7 AND S_8
LOGIC INDICATOR 3 = S_7 NOR S_8
LOGIC INDICATOR 4 = S_7 XOR S_8
LOGIC INDICATOR 5 = S_7 AND (NOT S_8)

Test all [four] combinations of the input switches to check your logic functions.[5]

15LS.1.5 Build a multiplexer two ways

A multiplexer takes two inputs, `A` and `B`, and a control signal, `select`, and outputs `A` when `select` is low, and `B` when `select` is high. Here is the truth table for a one-bit mux:

select	A	B		out
0	0	0		0
0	0	1		0
0	1	0		1
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		0
1	1	1		1

Structural model: Build and test a one-bit multiplexer structurally in Verilog using gates.

Behaviorial model: Build and test the multiplexer behaviorally using the Verilog conditional operator.

15LS.1.6 Build some digital comparators

2-bit equality detector: Make a comparator that detects equality between two 2-bit numbers.

[5] Here is our solution (but try it yourself first): `https://LAoE.link/FPGA/15L_CombLogic.v`.

2-bit *A*>*B* detector: Make a comparator that detects when one of a pair of two bit number (call it A) is larger than the other (call it B). You need not make the circuit symmetrical: it need not detect $B{>}A$, only $A{>}B$ versus $A \leq B$.

15LS.1.7 Build a more efficient adder

The two-bit full adder we used to explain Verilog hierarchy in §15N.3.7 consisted of two identical full adders for simplicity. The more efficient design you built out of gates in Lab §14L.4.2 used a half adder in the least significant position. Build a hierarchical two bit adder in the STEPFPGA using both a full adder and a half adder as shown in Fig. 15LS.7.

Use logic indicators for output as follows:

LOGIC INDICATOR 6 = CARRY
LOGIC INDICATOR 7 = SUM1
LOGIC INDICATOR 8 = SUM0

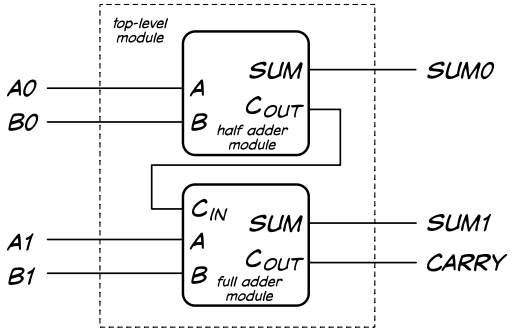

Figure 15LS.7: Two-bit binary adder.

15LS.2 Active-Low design

You have been asked to design a voting system that allows three people (A, B and C) to vote – each by pressing a *normally open* pushbutton if in favor of a proposal – and then lights one of four LEDs to indicate if none, one, two or three people voted *for* the proposition. You may use the N.O. outputs of the two pushbuttons (PB_1 and PB_2) already available on the breadboard but you will need to add an additional pushbutton for the third vote. (Does the added pushbutton need to be debounced as well?)[6] The output LEDs should be driven through MOSFETs as shown: see Fig. 15LS.8.[7]

Design the circuit: Initially, you may use any type of gate with as many inputs (with or without inverting bubbles) as you like, as well as inverters. Please use SUM-of-PRODUCTS design tech-

[6] No. Any switch bounce lasts, at most, a few milliseconds. No human is will see such a short flicker in the output.

[7] If you are short on time you can connect the outputs to logic indicators instead and skip the paragraph asking you to build two of the circuits with 74HC devices.

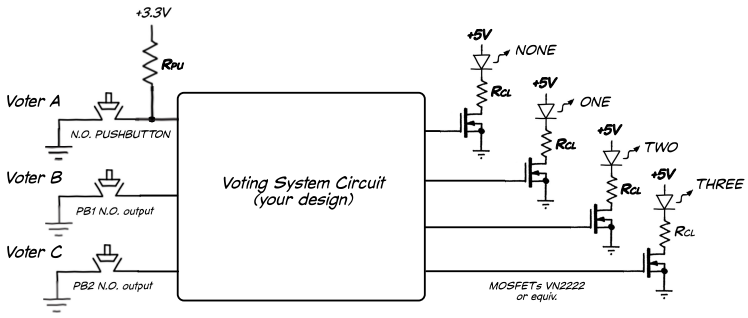

Figure 15LS.8: Block diagram of voting system.

niques and be sure to use the form of the gate that best expresses what is happening (i.e., apply deMorgan's Theorem as necessary to make the function of the circuit clear).[8]

Built and test your NONE and THREE designs with 74HC logic devices: Add the third pushbutton, build the four output indicator LED circuits and use discrete gates to build **only** the two circuits to light the NONE LED if *no buttons* are pressed and the THREE LED if *all three pushbuttons* are pressed. Note that you will have to convert your "any gates allowed design" to only use actual devices available in 74HC logic. Be sure to document what you *actually* build.

Built the complete circuit using the STEPFPGA: Now build the full voting circuit using the STEPFPGA.[9] Why is it ok to run the LEDs from $+5V$ when the STEPFPGA can only tolerate 3.3V max?[10]

[8] You may find it easier to draw the circuit for each output on a separate sheet. Otherwise, it gets very messy.

[9] Here is our solution for comparison: `https://LAoE.link/FPGA/15L_Vote3.v`.

[10] The 3.3V FPGA logic is isolated from the 5V by the MOSFETs.