

EE/CSE 371:
Design of Digital Circuits and Systems

Lab0: LabsLand and ModelSim Tutorial

Using Quartus Prime Software

Please install Quartus Prime Lite Edition 17.0 on your PC for writing your code, developing your system, and simulating it on ModelSim before running the code on a remote FPGA on LabsLand (**Note**: you do not need an FPGA to run ModelSim). In the case that you use a Mac, you can install a virtual machine, which allows you to run Windows on a Mac. VMWare Fusion 11.x Pro is free for students at the University of Washington, and you can get your copy here:

<https://e5.onthehub.com/WebStore/OfferingDetails.aspx?o=c58f2cd0-42ce-e811-810b-000d3af41938&ws=a4fce2bc-ac2d-de11-a497-0030485a8df0&vsro=8>

To install the Quartus Prime Lite software, go to

http://fpgasoftware.intel.com/17.0/?edition=lite&platform=windows&download_manager=dlm3

choose version 17.0 from the top drop-down menu, download the free web edition, and install it. Note that you will have to register to be able to download the software. The file to download is the 5.8 GB tar file under the “Combined files” tab. Extract the tar file using a program like 7-zip and run the QuartusLiteSetup-17.0windows.exe file. When it asks for the components to install, make sure you select each of these:

- Quartus Prime Lite Edition
- (Free) Devices: Cyclone V
- ModelSim: Intel FPGA Starter Edition (Free)

When the software installation is done, make sure to install the USBblaster driver. Run Quartus next, and if asked about licensing just run the software (we use the free version, so no license required).

NOTE: If you have trouble accessing the website to download the software, you can download it from this google drive. (you need to log in using your netID)

https://drive.google.com/open?id=1Tl_1G_DFn6Yps0DuEh3yUAr4vXpJ48SF

Warm-up exercise on Quartus and ModelSim

This task is a refresher on creating a Quartus project, writing a SystemVerilog program, and simulating it in ModelSim.

For this task, follow along with the series of video tutorials that will walk you through the steps of creating a full adder using SystemVerilog as well as simulating the design in ModelSim. For your reference, the source code used in the tutorials is in the appendix of this document.

Please follow the tutorials in the following order:

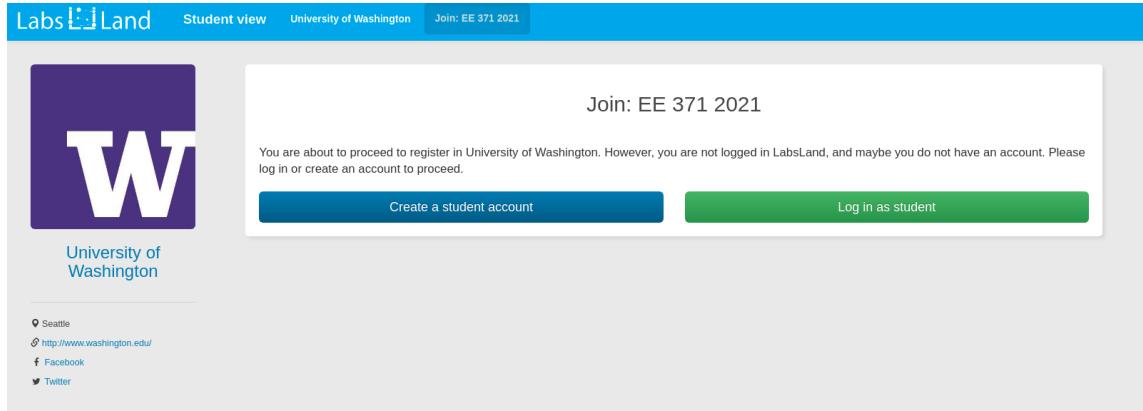
1. Launch the Quartus Prime software.
2. Create a project from scratch. Please follow the steps in the following videos and use the same project name as in the video <https://youtu.be/iLbmSTG7bpA>
3. Implement the full adder using SystemVerilog and simulate it using ModelSim. Please follow the following video: <https://youtu.be/BcvclrqZ2fc>
Note: that in the video when it refers to compiling the project for the first time, it may give you a compilation error. If you run the video for a few more seconds it'll tell you about setting the top-level modules so that the program compiles.
4. Mapping a SystemVerilog design to an FPGA. Please follow the steps in the following video and save an image of the simulation. <https://youtu.be/mnZt2iNNfp4>

After completing this task, you should create future projects in a similar fashion and use SystemVerilog and ModelSim accordingly. In all the labs, you will need to write the code and verify its functionality with ModelSim before loading it to LabsLand to run it on a remote FPGA. The FPGAs are shared and you will have about 2 minutes to use it.

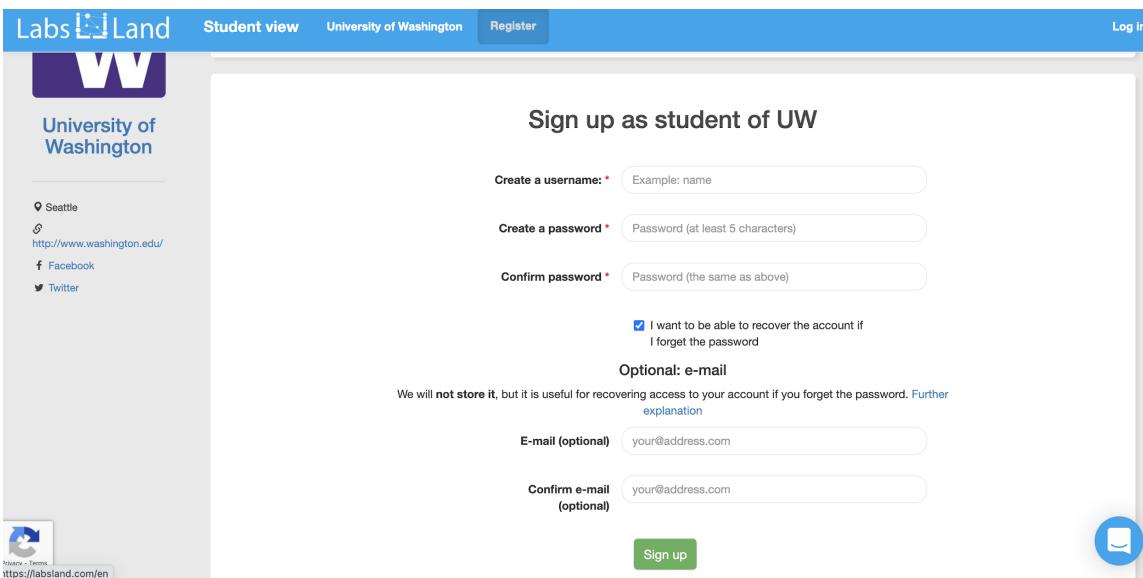
After verifying the code, now it is time to move to LabsLand, and try the full adder code on a remote FPGA.

Creating an Account on LabsLand

1. Go to the following link: <https://uw.labsland.com/standalone/join/YZHB3768>
2. Click “Create a Student Account”



3. Please sign-up by entering your desired username and password. We recommend that you also select the “*I want to be able to recover the account if I forget the password*” option in case you forget your password.

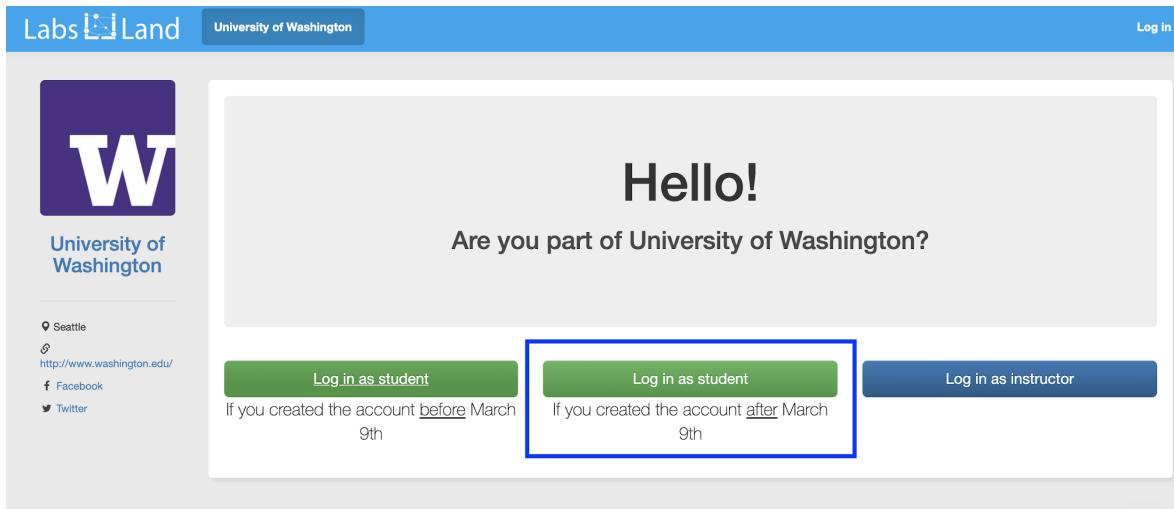


Then, click “**Sign up**” to finish the step.

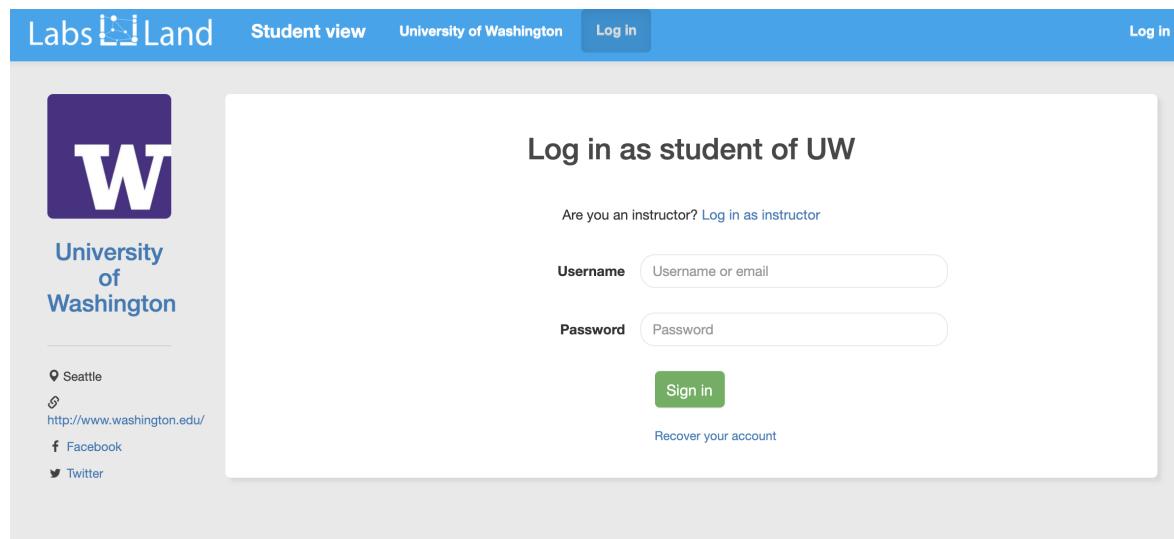
4. You have created an account on LabsLand! Please remember the credentials as you will be using this website for the entire quarter.

Logging into Your LabsLand Account

1. Go to the UW LansLand portal through this link: <https://uw.labsland.com/> and click “**Log in as student – If you created the account after March 9th**”



2. Enter your credentials and click “**Sign in**” to log in as a student.



3. After logging in, you should be able to see the EE 371's class page on LabsLand, which is shown in the figure below.

Labs Land

Student view University of Washington

Student: heransp21 Log out

Group: EE 371 2021

Intel DE1-SoC

Learn Hardware design with FPGAs using DE1-SoC

Access this lab >

Intel DE1-SoC (with audio)

Learn Hardware design with FPGAs using DE1-SoC

Access this lab >

The screenshot shows the student view of the UW LabsLand portal. At the top, it displays the UW logo and the text "University of Washington". Below this, there are links for Seattle, the website (http://www.washington.edu/), Facebook, and Twitter. The main content area is titled "Group: EE 371 2021". It features two lab options: "Intel DE1-SoC" and "Intel DE1-SoC (with audio)". Each option is represented by a thumbnail image of the hardware board, a brief description of the lab's purpose, and a blue "Access this lab" button.

4. To access the lab workspace, locate “**Intel DE1-SoC**” and click the “**Access this lab**” below it.
5. **IMPORTANT NOTE:** Whenever you are working on a lab, please always make sure that the webpage’s URL starts with “uw.labsland.com”; Otherwise, you may be looking at a wrong page and therefore working on incorrect lab materials. To avoid this, please do not click the LabsLand logo located on the top-right corner of the page, which will take you back to the generic LabsLand’s main page, not the UW LabsLand portal.

Loading the code to LabsLand FPGA

1. Locate “DE1 IDE SystemVerilog” and click the “Access” button below it. You will be directed to a new page called “SystemVerilog IDE for DE1-Soc”.

The screenshot shows the LabsLand website interface. At the top, there is a navigation bar with links for "Student view", "University of Washington", "EE 371 2021", "Intel DE1-SoC", "Student: heransp21", and "Log out". Below the navigation bar, there is a sidebar on the left with the University of Washington logo and social media links for Seattle, Facebook, and Twitter. The main content area features a section titled "Intel DE1-SoC" with a sub-section "Learn Hardware design with FPGAs using DE1-SoC". Below this are three buttons: "DE1 IDE Verilog" (Program Altera FPGA DE1 in Verilog), "DE1 IDE VHDL" (Program Altera FPGA DE1 in VHDL), and "DE1 IDE SystemVerilog" (Program Altera FPGA DE1-SoC in SystemVerilog). The "DE1 IDE SystemVerilog" button is highlighted with a blue border.

2. In the following page, select the “Add” button to import the top-module “DE1-SoC.sv” and file “full_adder.sv” that you created earlier into this system. You can choose the top-module using the dropdown menu under “Top level entity”. Make sure you select “DE1-SoC.sv” as the top-module. (Alternatively, you may also create new files by clicking “New” and copy the provided full adder example under “Examples” found in the bottom left corner of the interface into the corresponding new files. The top-module is named “main.sv” in this case, so make sure to adjust the settings accordingly.)

[Leave now](#)

SystemVerilog IDE for DE1-SoC Labs Land

The screenshot shows the SystemVerilog IDE interface. At the top, there are buttons for 'New', 'Add', and 'Download'. Below that is a file list with 'fulladder.sv' and 'main.sv'. The 'Information' tab is selected, showing the code for the top-level module 'main'. The code defines a module 'main' with inputs HEX0-HEX5, KEY, and SW, and outputs HEX0-HEX5, LEDR, and SW. It includes assignments for all outputs to '7'b1111111'. The 'Synthesize' button is highlighted with a blue box. A status bar at the bottom says 'All changes saved.'

```

1 // Top-level module for the full adder
2 // Author: Rania Hussein, University of Washington
3
4 module main (HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, LEDR, SW);
5
6   output logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
7   output logic [9:0] LEDR;
8   input logic [3:0] KEY;
9   input logic [9:0] SW;
10
11  fullAdder FA (.A(SW[2]), .B(SW[1]), .cin(SW[0]), .sum(LEDR[0]), .cout(LEDR[1]));
12
13  assign HEX0 = 7'b1111111;
14  assign HEX1 = 7'b1111111;
15  assign HEX2 = 7'b1111111;
16  assign HEX3 = 7'b1111111;
17  assign HEX4 = 7'b1111111;
18  assign HEX5 = 7'b1111111;
19
20 endmodule
21
22 module main_testbench();
23
24  // Testbench code here

```

Top level entity: main.sv

User interface: Standard | Edit

Documentation

- IO signal names

The compilation result will be displayed here

3. You will then be able to synthesize the code using the button “Synthesize”. Once the synthesis is complete and succeeds without errors, you can click on “Upload to FPGA” to load your design onto an FPGA.

[Leave now](#)

SystemVerilog IDE for DE1-SoC Labs Land

The screenshot shows the SystemVerilog IDE interface after synthesis. A green message box at the top says 'The program was successfully verified and compiled (3:21:46 PM)'. The 'Synthesize' button is highlighted with a blue box. The code editor shows the same Verilog code as before. The status bar at the bottom says 'All changes saved.'

The program was successfully verified and compiled
(3:21:46 PM).

```

1 // Top-level module for the full adder
2 // Author: Rania Hussein, University of Washington
3
4 module main (HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, LEDR, SW);
5
6   output logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
7   output logic [9:0] LEDR;
8   input logic [3:0] KEY;
9   input logic [9:0] SW;
10
11  fullAdder FA (.A(SW[2]), .B(SW[1]), .cin(SW[0]), .sum(LEDR[0]), .cout(LEDR[1]));
12
13  assign HEX0 = 7'b1111111;
14  assign HEX1 = 7'b1111111;
15  assign HEX2 = 7'b1111111;
16  assign HEX3 = 7'b1111111;
17  assign HEX4 = 7'b1111111;
18  assign HEX5 = 7'b1111111;
19
20 endmodule
21
22 module main_testbench();
23
24  // Testbench code here

```

Top level entity: main.sv

User interface: Standard | Edit

Documentation

- IO signal names

console main.fit.summary main.map.summary

```

$ quartus_map main --source COMPILATION_DIRECTORY/fulladder.sv --source COMPILATION_DIRECTORY/main.sv --f
Info: ****

```

4. After waiting for the remote FPGA to connect, you will see the webpage shown below. The right part of the page shows the buttons and keys of the FPGA. You can click on the buttons and keys accordingly as inputs. It is important to note that 'KEYS' need to be held down, as they do not function like switches.

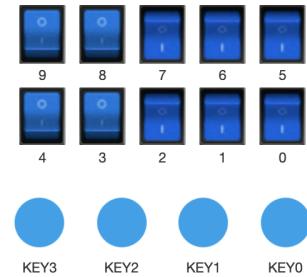
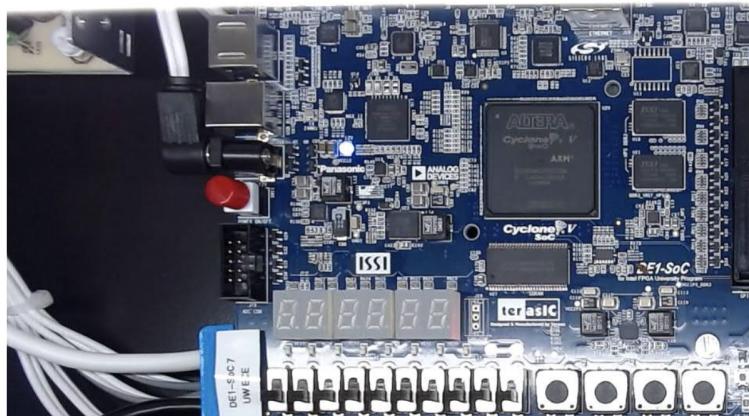


This FPGA is hosted at [University of Washington](#).



[Leave now](#)

Altera FPGA Laboratory



Appendix: Source code used in the videos

1. FullAdder module

```
module fullAdder (A,B, cin, sum, cout);
    input logic A,B, cin;
    output logic sum, cout;
    assign sum = A ^ B ^ cin;
    assign cout = A&B | cin & (A^B);
endmodule

module fullAdder_testbench();
    logic A, B, cin, sum, cout;
    fullAdder dut (A, B, cin, sum, cout);
    initial begin
        A = 0; B= 0; cin =0; #10;
        A = 0; B= 1; cin =1; #10;
        A = 1; B= 0; cin =0; #10;
        A = 1; B= 1; cin =1; #10;
        $stop;
    end //initial
endmodule
```

2. DE1_SoC module

```
module DE1_SoC (HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, KEY, LEDR, SW);
    output logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
    output logic [9:0] LEDR;
    input logic [3:0] KEY;
    input logic [9:0] SW;
    fullAdder FA (.A(SW[2]), .B(SW[1]), .cin(SW[0]), .sum(LEDR[0]), .cout(LEDR[1]));
    assign HEX0 = 7'b1111111;
    assign HEX1 = 7'b1111111;
    assign HEX2 = 7'b1111111;
    assign HEX3 = 7'b1111111;
    assign HEX4 = 7'b1111111;
    assign HEX5 = 7'b1111111;
endmodule

module DE1_SoC_testbench();
    logic [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5;
    logic [9:0] LEDR;
    logic [3:0] KEY;
    logic [9:0] SW;
    DE1_SoC dut (.HEX0, .HEX1, .HEX2, .HEX3, .HEX4, .HEX5, .KEY, .LEDR, .SW);
    integer i;
    initial begin
        SW[9] = 1'b0;
        SW[8] = 1'b0;
        for (i=0; i<2**8; i++) begin
            SW[7:0] = i; #10;
        end
    end
endmodule
```