

EE 231 Prelab 1

Introduction to Verilog HDL and Quartus

In this lab you will design simple circuits by programming the FPGA. At the end of the lab you should be able to understand the process of programming a PLD, and the advantages of such an approach over using discrete components.

1. Prelab

- 1.1. Read the material provided in the supplementary section 2.
- 1.2. Using the schematics of the board, if the FPGA sends a logic 1 to one of the LEDs, would it turn on or off?

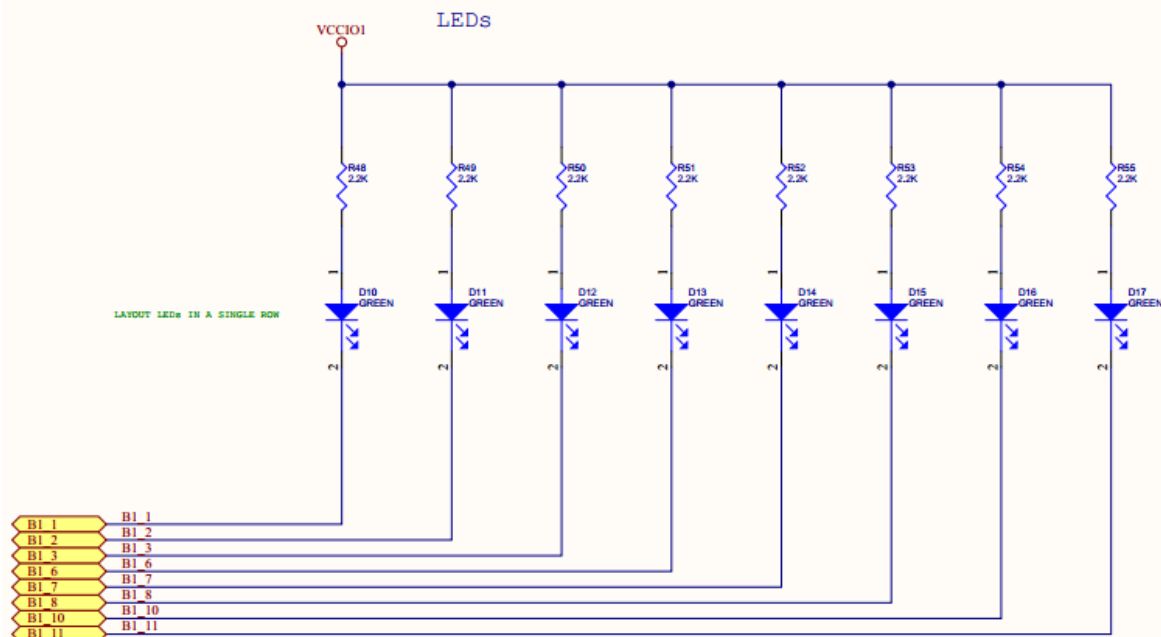


Figure 1: Schematic of Cyclone IV LEDs

2. Supplementary Material

2.1. Introduction to Verilog

- 2.1.1. In order to write a Verilog HDL description of any circuit you will need to write a module, which is the fundamental descriptive unit in Verilog. A module is a set of text describing your circuit and is enclosed by the key words **module** and **endmodule**.
- 2.1.2. As the program describes a physical circuit you will need to specify the inputs, the outputs, the behavior of the circuit and how the gates are wired. To accomplish this, you

need the keywords input, output, and wire to define the inputs, outputs, and the wiring between the gates, respectively.

2.1.3. There are multiple ways to model a circuit:

- Gate level modeling
- Dataflow modeling
- Behavioral modeling
- Or a combination of the above

2.1.4. A simple program modeling a circuit (Figure 2) at the gate level is described below.

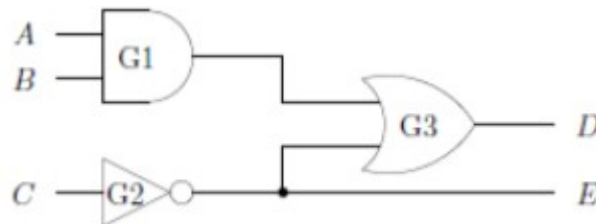


Figure 2: Simple

Circuit

Program 1: Simple Program in Verilog Modeling a Circuit at the Gate Level

```

module simple_circuit(output D, E, input A, B, C);
    wire w1;

    and    G1(w1,A,C);
    not    G2(E,C);
    or     G3(D,w1,E);

endmodule

```

Program 2: Simple Program in Verilog Modeling a Circuit at the Gate Level

```

module simple_circuit(output [1:0] Y, input [0:2] X);

    assign Y[1] = (X[0] & X[1]) | ~X[2];
    assign Y[0] = ~X[2];

endmodule

```

2.1.5. As seen above, the outputs come first in the port list followed by the inputs.

2.1.6. Single line comments begin with //

2.1.7. Multi-line comments are enclosed by /*...*/

2.1.8. Verilog is case sensitive

2.1.9. A simple program modeling a circuit using dataflow is provided below.

2.1.10. You can use identities describing multiple bits known as vectors.

2.1.11. Given an identifier [7:0]X you can assign values by:

```
assign [7:0] X = 8'b00001111;
```

where the 8'b specifies that we are defining an 8 bit binary number and 00001111 is that 8 bit binary number. You can also assign parts of the number as:

```
assign [2:0]X = 3'b101;  
Which assigns only the last three bits of X.
```

2.2. Using Quartus

To implement the circuits that you will design on the FPGA there are a few key steps.

2.2.1. Start a new Project

1. Select File > New Project Wizard.
2. Set the directory name. Make sure that you create a new folder/directory for each project and do not just copy the directory over.
3. Set the name of the project. It will be simple if you name it by the lab name, e.g., Lab 1.
4. Click "Yes" to create a directory if it does not exist.
5. You can add existing files if you already have them, otherwise select "Next."
6. Next you need to specify the device that you are using. The device family we are using is the Cyclone IV E and the device is the EP4CE22F17C6 (This will need to be altered when simulating your circuit. We will discuss this in the simulating section).
7. Press Next.

8. Press Finish.

2.2.2. Writing the Code

1. Select File > New.
2. Choose Verilog HDL File.
3. Click Ok.
4. Select File > Save As. NOTE: The name of the file should be the name of the Project. Also, the name of the module needs to be the name of the file.
5. Choose Save as type, and select Verilog HDL File.
6. Put a check mark in the box “Add file” to current project. Unless the file is part of the project you won’t be able to proceed. If you do not add the file now you can always add it later by selecting Project > Add/Remove Files to Project.
7. Click Save.
8. Now you are ready to type in your program.

2.2.3. Compiling

1. Select Processing > Start Compilation, or click on the play icon on the toolbar. Ctrl + L is the keyboard shortcut for this.
2. You can click on Processing > Compilation Report to see the results of the compilation.
3. If there are no errors, then your program’s syntax is correct. This does not mean that your program will do what you want because you may have some logic errors.

2.2.4. Pin Assignment

You need to specify which outputs. Some pins have already been internally wired to the LEDs and push buttons on the board. A list of those pins is provided in Table 1. **Be sure the selected device is the Cyclone IV E: EP4CE22F17C6.**

1. Select Assignments > Pin Planner.
2. Under Filter select Pin.
3. Double click <<new node>>, a drop-down menu should appear to add an input/output. Select the input/output you want to assign. After compilation, if successful, your input/outputs should already be listed.

4. In the column labeled Location, select the pin you want to assign to that input/output. Table 1 shows the locations of the hardwired pins for your evaluation board. For example: To connect output C to the LED7, you would select Location Pin L3 for Node C.
5. Repeat Steps 3 and 4 to assign all inputs/outputs for your circuit.
6. The Altera default is that all unused pins should be assigned “As outputs driving ground.” This is a good choice for pins not connected to anything (it reduces power and noise), but is not good for pins that may be connected to a clock input – you then have both the clock and the Altera chip trying to drive this input. A safer choice is to define all unused pins “As input tri-stated with weak pull up resistor.”
 - a. Go to Assignments > Device.
 - b. Click on Device and Pin Options.
 - c. Select the Unused Pins tab.
 - d. From the Reserve all unused pins drop-down menu, select As input tri-stated with weak pull-up resistor.

Signal Name	CPLD Pin #	Description
LED0	PIN_A15	LED[0]
LED1	PIN_A13	LED[1]
LED2	PIN_B13	LED[2]
LED3	PIN_A11	LED[3]
LED4	PIN_D1	LED[4]
LED5	PIN_F3	LED[5]
LED6	PIN_B1	LED[6]
LED7	PIN_L3	LED[7]
KEY0	PIN_J15	Push-Button[0]
KEY1	PIN_E1	Push-Button[1]
SW0	PIN_M1	DIP-Switch[0]
SW1	PIN_T8	DIP-Switch[1]
SW2	PIN_B9	DIP-Switch[2]
SW3	PIN_M15	DIP-Switch[3]
CLOCK_50	PIN_R8	50MHz Clock Input

Table 1: Pin Assignments for the LEDs, Push Buttons, DIP-Switches, and Clock inputs

2.2.5. Simulating the Designed Circuit

When simulating the circuit you need to figure out the waveforms for the inputs that will make you confident that your circuit works. If you have a simple circuit, you can easily test all the possibilities. As the circuit gets more and more complicated you will need to figure out a scheme to verify its operation. Due to limitations in the software we are using you must change your device to a MAX II. You don't need to specify a specific device, but just need the MAX II family. Simulation is just a functional check of your code, so the device selection will not affect the simulation results of your circuit. However the device selection does affect your pin assignments, so double check your device before you assign your pins or you will have to repeat those steps for the correct device. To change your device after your project is created:

1. Go to Assignments > Device.
2. From the Family drop down menu, select MAX II.
3. You can then pick any of the specific devices or check Auto device selected by the Fitter.

In simulating your circuit there are four main steps.

1. Create a waveform file.
2. Select your inputs and outputs.
3. Create a waveform for each input.
4. Run the simulation to generate the output for verification with your expected results.

A more detailed explanation of the above steps is described below.

1. Select File > New.
2. Click on Vector Waveform File.
3. Click Ok.
4. Save the file using some meaningful name: filename.vwf.
5. Set the desired simulation time by selecting Edit > End Time.
6. Select View > Fit in Window.
7. Select the inputs and outputs you want to observe by clicking Edit > Insert > Insert Node or Bus.
8. Click on Node Finder.
9. Click on the input/output you want to observe by clicking on the > sign. Repeat this process for all your inputs/outputs.

10. The next step is to specify the logic value of each of the inputs you have selected and duration of that value by right clicking on the input and selecting Value > *your selected value*
11. Save the file, e.g. lab2.vwf.
12. After the waveforms have been defined, we can simulate our circuit. There are two types of modes that we are concerned with: 1) Functional: we are not worried about the delays and we are interested to make sure that logically the circuit is working. 2) Timing: we simulated the circuit and include the delays in all the gates. First perform the functional simulation, and then perform the timing. To select the mode of the simulation:
 - a. Select Assignments > Settings.
 - b. Click on Simulator Settings.
 - c. Choose Functional or Timing. **For now choose Functional unless otherwise instructed.**
1. Create the required netlist that the waveform file will be applied to by selecting Processing > Generate Functional Simulation Netlist.
2. Run the simulation by clicking Processing > Start Simulation, or by clicking on the play icon in the simulation waveform window.

2.2.6. Programing the FPGA

The final step is to program the FPGA with your designed circuit. Be sure the correct device is selected.

1. Select Tools > Programmer.
2. Select JTAG in the Mode box.
3. If USB Blaster is not chosen in the box next to Hardware Setup, select by clicking on the Hardware Setup. Then click on USB-Blaster, the Close.
4. You should see a file listed with extension .sof, if not add it,
5. Finally, press Start. The program will download on your board and once it is finished you can test your circuit in hardware.