Lab 1 – Introduction to Verilog HDL & the Lab Design Flow

In this lab, you will be introduced to the Verilog Hardware Description Language and the design flow we will be using in the labs throughout the course. More specifically, the designs you produce and test will be written in the SystemVerilog language, a superset of Verilog that is both a Hardware Description Language (HDL) and Hardware Verification Language (HVL). Verilog files typically use a ".v" extension, while SystemVerilog files are named with a ".sv" extension. All of the design files in this course will have a ".sv" extension. For our purposes, Verilog and SystemVerilog have essentially the same functionality, with SystemVerilog introducing the useful **logic** data type. You can read more about the differences between these two languages here.

Setup (Lab 0): AWS Connection and VS Code

Please ensure that you have completed the steps from Lab 0. You should be able to log in (via VS Code) to the class AWS server (ece-000) using the ssh keys you generated during Lab 0. All of the content for this lab can be found in the following directory: **/home/<username>/ece2613/lab1**

Additionally, you should have set up the Oracle VM VirtualBox software during Lab 0. This VirtualBox will be used to upload your designs to your hardware.

Please consult with a lab instructor if the following steps have not been completed:

DE10-Lite Hardware

The hardware you will be using is DE10-Lite, with which you have been supplied. This board contains an FPGA device manufactured by Altera. You have also been supplied with a board-compatible USB cable, which will be used to load your designs into the board. You can read more about the DE10-Lite hardware in the **intel_altera_v2.pdf** document posted on Canvas.

Basic Design Flow for Lab Exercises

The Design Flow for the labs in this course consists of the following steps:

- Enter your design using **SystemVerilog HDVL** onto the class AWS server (**ece-000**) using **VS Code**
- Simulate your design and verify that it is functionally correct (Intel Quartus simulation software)

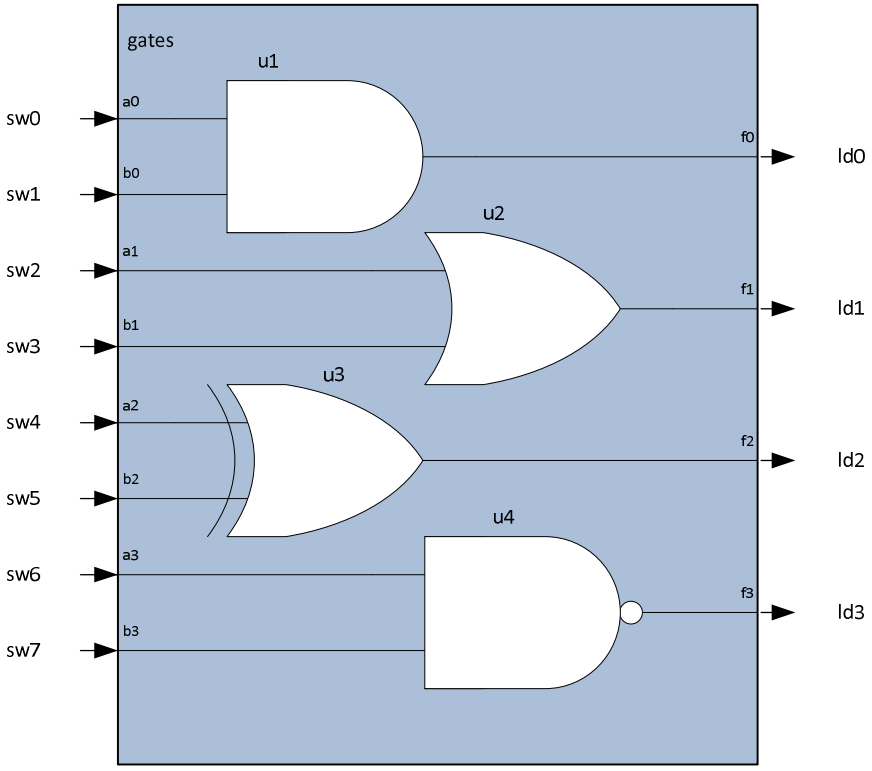
- Create a binary file (*.sof) that can be loaded into the FPGA hardware
- Download the binary file to your local machine and place it in the VirtualBox shared folder (created during Lab 0)
- Open VirtualBox and load the binary file into the DE10-Lite hardware
- Test your design in hardware and compare with simulation results

The Design

For this lab, you need to design four different gates by **instantiating** the appropriate **primitives**. You will use the Verilog gate primitives as discussed in class, videos, and the textbook (examples of gate primitives). You may also use **assign** statements with the proper gate **operators** (examples of gate operators). The four gates for this design are:

- AND, OR, XOR, NAND

The inputs of the gates will be connected to switches and outputs to LEDs. A block diagram/schematic of the gates design (module) is shown below.



Complete the truth table for the gates in this design.

a0, 1, 2 or 3	b0, 1, 2 or 3	f0	f1	f2	f3
0	0				
0	1				
1	0				
1	1				

Design Entry

Start by opening the VS Code IDE tool and connect to the class AWS server. Open the **lab1** folder and double click on the gates.sv file. This file contains a skeleton of your design module. You are now ready to enter your design.

Within gates.sv, instantiate each of the four primitive gates, connecting them as shown in the above diagram. The instance names are u1, u2, u3 and u4 as shown above. If you chose to implement the gates using continuous assign statements, the instance names are not needed. For example:

Instantiation implementation: `and u1 (f0, a0, b0);`

Assignment implementation: `assign f0 = a0 & b0;`

Save your edited result when you are finished editing your design (Ctrl+S keystroke).
Remember to save frequently!

Simulation

This is an operation where the machine checks your design's syntax and if ok, it simulates the design – to make sure it matches your design objective or specifications. The design specification that is tested for this lab is contained in the truth table above.

In your hardware implementation, you will be testing this by toggling switches and observing the outputs of the LEDs. One way to see if your design meets your specifications before committing it to hardware is to provide all of the possible input combinations to the design module in a wrapper module, run the simulator, and check by viewing a timing diagram to check that each output is correct. Such a wrapper is called a verification **testbench**.

A better way is to create a **self-checking testbench**. This module also knows the expected values and compares the simulated outputs to the expected outputs. Then any mismatches are printed to the console display.

In your directory, you are provided a self-checking testbench, called **tb_gates.sv**. The details of this module will be covered in future lectures and labs. For now, it is important that you know that it reads a text file called **tb_gates.txt**. You can open this file and compare it to your truth table. These data are read by the self-checking testbench. When you run the simulation (tb_gates.sv), each of these inputs will be applied to your design every 20 nsec. If your design has errors, there will be mismatches. **You must correct these mismatches before continuing to the hardware implementation.**

To simulate your design, find the file **gates.m_sim** in your environment, right-click on it and select **Run**. Observe the terminal at the bottom of the VS Code environment. This will contain messages from your simulation. The messages are also copied to a file: **tb_gates.log**. Open this file by double-clicking on it (you may need to refresh the directory navigation pane in VS code).

You are looking for a message: Simulation complete – no mismatches!!! You need to take a screenshot of this window to put into your lab report.

If you have mismatches, the inputs, expected outputs, and your design outputs will be displayed. You should be able to find your design error using this information to direct you.

Synthesis

This step creates a binary file that must be copied to your local PC, then to VirtualBox, and finally loaded into your DE10-Lite board through VirtualBox. The design can then be tested in hardware.

A top-level input/output wrapper is now substituted for the testbench in your design. For all labs, it will be called: **DE10_LITE_Temple_Top.sv**. You can view this file if you like, but do not edit it. There is a related file called **lab1_top.qsf**. To synthesize the design, right-click on this qsf file and select Run. This will take a little time for the tool to optimize your design and map it to gates for your hardware.

When it completes without errors, it should have built a binary file called: **lab1_top.sof**. In order to find this file, you must navigate to the "output_files" directory (may need to refresh again). This is the file that you need to load into your hardware. **If you try to click on this file, VS Code will produce a warning since it is a large binary file that should not be viewed in a text editor.** To download it, right-click on the filename and select Download. The file will be in your Downloads path on your local PC.

Next, copy or move the binary file to the shared folder that you set up in Lab 0. Connect the DE10-Lite board to your PC using the provided USB cable. Now, open VirtualBox and select the green "Start" button. Sign in using the password "user" in all lowercase. Select "Activities" at the top left of the window to open the navigation pane. Select "Files" from the navigation pane. Finally, double-click on the lab1_top.sof file to load it into the DE10-Lite board.

Verify your design by toggling the switches and observing the LEDs. Is the behavior what you expect? Does it meet the design specifications?

Shutting Down/Logging Off

When you are finished, close all of the open tabs inside the VS Code IDE. Closing VS Code will disconnect you from the AWS server. Exit VirtualBox and ensure that the "Power off the machine" option is selected.

Troubleshooting

Sometimes, loading designs on the DE10-Lite hardware can be a bit cumbersome. The loading process may freeze or an error may be produced that does not allow loading to begin. This is usually due to VirtualBox not recognizing the board as plugged into your PC. The following steps can help to solve these issues:

- Select "Devices" from the top navigational panel in VirtualBox and observe "USB" connections. Ensure that "Altera USB-Blaster [0400]" has a check mark next to it. If it does not, click on it to select it.
- Ensure that the board is plugged into a USB 3.0 port on your PC. If the board is plugged into a USB 2.0 port, the design may not load. On Windows, you can determine which USB ports are version 3.0 by navigating to Control Panel -> Device Manager -> Universal Serial Bus controllers.
- Restart VirtualBox. Connect the DE10-Lite to your PC before starting VirtualBox as this has sometimes corrected the issues.
- Ensure that the USB bit blaster driver is properly installed. Follow the steps here.