CS456 FPGA Lab
Lab 0: Getting Set Up and Familiarizing Yourself with Tools

(using EECS151/251 Lab 0, Berkeley)

1 Setting Up Accounts

1.1 Getting a Github Account

If you haven't done so previously, sign up for a Github account at `https://github.com/` with your jmu.edu email address.

If you already have a Github account that's registered with your personal email address, don't create a new account. Instead, login to Github, go here `https://github.com/settings/emails`, and add your jmu.edu email address to your Github account.

Once you have an account, make sure Dr. Weikle has the information about it. You can email her at weikleda@jmu.edu if needed.

2 Getting Familiar with our Development Environment

2.1 Linux Basics

In this class, as in previous classes such as 261, we will be using a Linux development environment on stu. If you are unfamiliar or uncomfortable with Linux, and in particular, using the terminal and manipulating files on stu, you should definitely check out this tutorial:

`https://www.digitalocean.com/community/tutorial_series/getting-started-with-linux`

It is highly recommended to go through all four parts of the tutorial above, even if you already are familiar with the content. To complete the labs and projects for this course, you will find it helpful to have good command line skills.

One of the best ways to expand your working knowledge of bash is to watch others who are more experienced. Pay attention when you are watching someone else's screen and ask questions when you see something you don't understand. You will quickly learn many new commands and shortcuts.

2.2 Git Basics

Version control systems help track how files change over time and make it easier for collaborators to work on the same files and share their changes. For projects of any reasonable complexity, some sort of version control is an absolute necessity. There are tons of version control systems out there, each with some pros and cons. In this class, we will be using Git, one of the most popular version control systems. It is highly recommended that you make the effort to really understand how Git works, as it will make understanding how to actually use it much easier. Please check out the links on Canvas for a high level overview.

Git is a very powerful tool, but it can be a bit overwhelming at first. If you don't know what you are doing, you can really cause lots of headaches for yourself and those around you, so please be careful. If you are ever doubtful about how to do something with Git ask the professor or an experienced classmate.

For the purposes of this class you will probably only need to be proficient with the following commands:

- `git status`
- `git add`
- `git commit`
- `git pull`
- `git push`
- `git clone`

Git has a huge feature set which is well documented on the internet. If there is something you think Git should be able to do, chances are the command already exists, but you will likely be able to do much in this class by simply using the single repository workflow.

2.3 Acquiring Lab Files

The lab files, and eventually the project files, will be made available through a git repository. Today though we are keeping it simple. You can download the files from a repository on my github account, https://github.com/weikleda/cs456lab0. You can simply download the files today. Once you have downloaded the file make sure and move it to a place you will recognize and extract it.

3 Your First FPGA Design

Throughout the semester, you will build increasingly complex designs using Verilog, a widely used hardware description language (HDL). For this lab, you will use basic Verilog to describe a simple digital circuit.

Now that you have downloaded this lab's skeleton files, cd to where they are located. You will note that there is a `lab0.srcs` directory and a `lab0.xpr` file.

This course will use the Xilinx Vivado Design Suite ("Vivado" for short). We will initially use Vivado's Integrated Development Environment (IDE) instead of any command-line tools, though you will eventually see that the framework (especially with Pynq) is ripe for automation with TCL and Python scripting.

The lab skeleton files include the project meta data file, `.xpr`, a Verilog source file for a simple top-level module, `ml505top.v`, and a constraints file, `ml505top.xdc`.

HDL source files like `ml505top.v` (where the HDL is Verilog) describe the circuit that you want to create on the FPGA. `ml505top.v` describes a circuit that is the *top-level* of your circuit: it has access to the signals that come into and out of the FPGA chip. Constraints files, such as `ml505top.xdc`, allow the engineer (you!) to tailor specific properties of the synthesized design to how they wish to use their specific chip. This includes the crucial mapping between FPGA input/output pins and signal names used in circuit descriptions. We'll cover more on constraints files later.

3.1 Set up your Pynq-Z1

1. You should already have the imaged SD card installed to provide a boot OS for the onboard ARM (mostly for use later).
2. Connect the USB interface to a spare USB port on your workstation. When the PYNQ-Z1 is connected to the laptop through USB and powered on, it will automatically assign the IP address of 192.168.2.1 to the ethernet port, so you can work with the PYNQ-Z1 which assigns itself 192.168.2.99
3. Turn the board on with the switch on the board. Note you may need to wait for a response. If you have waited a minute or two and it still doesn't come on (indicated by power lights) switch it off and then on again.

3.2 Open the Lab 0 project in the Vivado Design Suite

Open up the Vivado software by finding it using the menu and double clicking.

Once in Vivado, open up the `lab0/lab0.xpr` project file by selecting Open Project from the Quick Start menu. Look around the environment to try and get a feel for the GUI. Open up the `lab0/lab0.srcs/sources_1/new/ml505top.v` source file. Again, this file contains a Verilog module description which specifies which signals are inputs into the module and which signals are outputs.

The `BUTTONS` input is a signal that is 4 bits wide (as indicated by the [3:0] width descriptor). This input signal represents the logic signals coming from the momentary push-button switches on the bottom right side of your Pynq-Z1 board. You should inspect your board to find these switches and confirm that there are 4 of them. Another basic input signal is `SWITCHES`, which is 2 bits wide (as indicated by the [1:0] descriptor). Each of these two signals represents the slide switches on the Pynq-Z1, located just to the left of the momentary switches (look for SW0 and SW1).

The `LEDS` output is a signal that is 6 bits wide (as indicated by the [5:0] width descriptor). This output signal represents the logic signals coming out of the FPGA and going into the bank of LEDs at the bottom right of the Pynq-Z1, just above the buttons. Almost. There are only 4 LEDs there; 2 more are tri-color LEDs located just above the slide switches in the middle.

In this file, we can describe how the slide switches, push buttons and LEDs are connected through the FPGA. There is one line of code that describes an AND gate that takes the values of one of the buttons and one of the slide switches, ANDs them together, and sends that signal out to the first LED. Let's put this digital circuit on the FPGA!

3.3 Synthesize and Program

1. In Vivado, locate the *Flow Navigator* pane to the left of the screen. Near the bottom, under *Program and Debug*, click *Generate Bitstream*. Accept the default settings and wait. This should invoke the dependent steps in the flow: *Synthesis* and *Implementation* (among other things).
   - Selecting *Project Manager* will give you a nice overview of the progress of various background steps while this happens.
   - So will watching the *Messages* and *Logs* output.
2. When the synthesis and bitstream generation is done, select *Open Hardware Manager* and connect to your FPGA.
   - If you haven't before, or the hardware manager says no devices are connected, select *Menu* → *Open New Target*
   - You should see `xilinx_tcf` listed under Harware Targets in the top pane. In the bottom pane, two entries: `arm_dap_0` and `xc7z020_1`. That's good. *Next* → *Finish*.
3. Back in the *Flow Navigator* on the left, under *Program and Debug*, select *Program Device*. The only option to program will be the FPGA, `xc7z020_1`. The default bitstream file path should work too.
4. See if it worked! What happens when you push the BTN0 button? What about when you change SW0? Both?

Go ahead and extend this example with more AND or other gates to see them in action!

There is no checkoff for this lab.

4 References and Resources

A very handy overview of what your Pynq-Z1 board can do is published by Digilent in the Pynq- Z1 Reference Manual: `https://reference.digilentinc.com/reference/programmable-logic/pynq-z1/start`