Lab 1: Getting Started

Objective

This lab will get you up and running, able to program your chosen FPGA board, with your chosen tools. You will create a trivial input-process-output system to get your toolchain debugged.

Procedures

1. Make a preliminary choice of FPGA board and programming tools. This is merely preliminary -- you can change your mind later. However, the better decision you make early, the better your chances of overall success, as you won't need to backtrack at all.
2. Get your FPGA board setup and your tools installed. This path is to the "current release" of Vivado:

   ```
   /afs/ece/support/xilinx/xilinx.release/Vivado
   ```

   and the you'll need the following environment variable set:

   ```
   LM_LICENSE_FILE=2101@xilinx-lice.ece.cmu.edu
   ```
3. You'll need to get in the habit of searching out documentation and finding ways to get questions answered on your own. To help jumpstart that process, every member of your team needs to get an account on the Xilinx support forums (forums.xilinx.com). Find a non-noob question to ask on the forums and do so.
4. Take a look at the documentation on the course website that pertains to your FPGA board and tools. Spend some time digging around on the Xilinx website to see what other application notes, white papers, etc. might be of use.
5. Program a simple input / output system to show that you can program your board. Your system may be Verilog only, or it may use the EDK and one of the CPU options (ARM, MicroBlaze, etc).
   a. You need to take a multi-bit input from either FPGA board switches, serial or USB ports. If you use serial/USB, you'll need to hook up a laptop or other system to provide controlled inputs.
   b. You need to transform this input in some simple fashion (XOR, Add, etc) to create a different, multi-bit output. The only point of the transformation is to ensure your system programming (either verilog or C) is actually working, actually doing something deliberate. As such, I don't recommend a simple pass-through or inversion, for they are simple enough to accidentally and non-deliberately get working.

   c. The output can be displayed on LEDs on the FPGA or output to serial or USB ports. Once again, serial or USB outputs need to be displayed on some other device.
   d. Use the LCD module to display one or more messages.[1] Dynamic messages based on input data would be really cool. You will find three verilog files on the course website: lcd_control.v, testFSM.v and chipInterface.v. The first is a module with an FSM to manage initialization and interface details to the LCD module. To display a message, you will need to craft a fairly simple FSM to send each character, one at a time, to the lcd_control.v module. An example of how to use it is in the testFSM.v file. chipInterface.v is a top-level module.

6. Make other decisions about your toolchain and get them up and running. Will you use svn, git, cvs or mecurial for your code repository? Where will your project documentation be kept and in what formats. Will someone be the "keeper of the schedule" and who will that person be. Etc.

Report

Document what has occurred in a *simple* lab writeup, one per team. Include discussion about your group's processes. List each member's username on the Xilinx forums and include links to any questions you have already posed. Demo your system to the professor or a TA.

[1] If your board doesn't have an LCD module, negotiate with the Instructor for a replacement task.