COSC175: FPGA Lab

Work through the following instructions and exercises. Whatever you do not finish as part of the in-lab time should be completed, along with your collaboration document, as part of your Lab 1 report.

These instructions are quite detailed and important to follow carefully in Parts 0-2, so you need to **read every word**, and I highly recommend **reading each step all the way through before you act on it**, to make sure you don't skip anything.

Vivado can look intimidating and has many features, but we will discover them just a little bit at a time. So, we'll spend today getting used to describing hardware using SystemVerilog for the first time, and then testing your circuit designs by programming your FPGAs (more on that later)!

You have keycard access to the lab room to continue your work at any time, but the lab laptops and FPGA boards should not leave room C107 at any time. In future labs, you will have the ability to "sign out" an FPGA from the lab, but they will stay put while we are still learning the basics. Laptops will always remain in C107 and should be returned to the cart for charging any time you leave the room.

Part 0: Your First Vivado Project

1. Start Vivado
   - Open Vivado on your desktop. If you don't see it, navigate to the start menu and search Vivado to open it.

2. Create a new project
   - Once Vivado is open, click "Create Project" to make a new project.
   - Call your project "Lab1YourName".
   - For the project type, select RTL Project and click Next.
   - For the "Add Source," "Add Existing IP," and "Add Constraints" steps, just click next. These are not important (for now -- if, when you revisit this lab or in a future lab, you have a starting design to incorporate, you will use this step to connect to existing SystemVerilog files).
   - When you reach "Default Part," you must go to Boards and search using the search bar for "Basys3."
   - You might need to select "Refresh" if the Basys 3 does not appear right away.
   - You might need to install the board files, but hopefully, they're already downloaded. If you don't see anything pop up in the search, call over a TA, and we'll help you out.
   - Next, you'll be shown the "New Project Summary," summarizing everything you've done up to this point.
   - Click Finish to create your very first project!

3. Add a source
   - The screen you'll see next can be intimidating, as there's a lot of information to take in! Again, we will unveil the mysteries of Vivado step-by-step, so focus on the instructions below for now. If you ever accidentally close a sub-window or have trouble following along, use the top menu bar to select "Layout" --> "Default Layout", and that should take you back to something that looks like this starting screen.

- On the left side of the screen, you'll see a sidebar called the "Flow Navigator." This is your primary way to get between the different utilities Vivado offers and is the main UI piece we will interact with.

- Our first step is to add a source file so that we can implement our first hardware design!

- Find the section in the Flow Navigator called "Project Manager." Underneath, there will be a button that says "Add Sources." Click that button.

- Select "Add or create design sources" and click next.

- Click "Create File." This will cause a popup to appear. We will be using SystemVerilog for the file type, so select that option from the drop-down menu. You can call your file gates.sv (.sv means that this is a SystemVerilog file). Then, click OK.

- You should now see "gates" in the list of design sources. Click Finish to create our file!

- This will cause a new popup to appear, asking you to define the module. This isn't important for now, so just click OK and ignore the warning message. That's just Vivado testing your willpower. Don't let it scare you!

4. Implementing an AND gate
   - The file gates.sv should be open in the top right portion of the main window. If it isn't open, you can navigate inside the sources subwindow to find gates.sv and then double-click it to open it.

   - Once you have it open, retype our implementation of a structural and gate.

```
// This is a structural implementation of an AND gate

module gates(input logic a, b, output logic y);

&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;and(y, a, b);

endmodule
```

Congrats! You've created a module that has the function of an and gate. **Save** your SystemVerilog file (the floppy disk icon), and optionally skip ahead to the "Lab Save and Clean-Up" section below to back up a copy of your progress so far.

Part 1: Your First Deployed Hardware Design

Now that you've defined a circuit (your individual AND gate!), it's time to test it! To test our hardware designs in this course, we will utilize an FPGA, which stands for Field Programmable Gate Array. It is (mostly) what it sounds like, and it is really cool – it provides you with an **array** of small programmable logic blocks (lookup tables and flip-flops, rather than literal **gates**) plus programmable wiring between them, all of which you can **program** out in the **field** (i.e., at your desk) to behave according to your own functional and timing specifications!

These reconfigurable circuits are used in hardware prototyping and in real industry settings to test new or experimental hardware designs. In this course, the Basys 3 will provide us a playground to deploy, test, and observe results of an actual implementation of your own designs.

- **Open the box for your Basys 3 board and take a look!**. Refer back to the picture from the pre-lab, and appreciate the various features and ports IRL!

- For our first module, we will connect two switches to our inputs and one LED to our output. This will allow us to set the value of each input to 0 or 1 (LOW or HIGH) and observe whether the output

LED is off (0) or on (1).

- If you look closely, you'll notice that next to each component on the board are pin numbers (Usually a letter followed by a number). This will be important so we can connect our variables to the board properly. We will construct a **constraints file** that tells Vivado how to translate between the inputs and outputs defined in our HDL (a, b, and y), and the physical resources available on the board (e.g., switches and lights).

1. Creating your Constraints file (connect your design to real inputs and outputs!)
   - Find the section in the Flow Navigator called "Project Manager." Underneath, there will be a button that says "Add Sources." Click that button, just like before.
   - However, this time select "Add or create constraints" and click next.
   - Click "Create File." This will cause a popup to appear. We will be using XDC for the file type, so select that option from the drop-down menu. You can call your file Constraints.xdc. Then, click OK.
   - You should now see Constraints in the list of constraints. Click Finish to create our file.
   - Double-clicking on our file under sources will open it, and you should retype this text exactly (**the syntax for constraints files is unforgiving, and spacing and capitalization matter**). Notice that in the last line, there are no commas between a, b and y.

```
set_property PACKAGE_PIN V17 [get_ports {a}]

set_property PACKAGE_PIN V16 [get_ports {b}]

set_property PACKAGE_PIN U16 [get_ports {y}]

set_property IOSTANDARD LVCMOS33 [get_ports {a b y}]
```

2. Synthesize and test your design!
   - Once your gates.sv and Constraints file are ready to go (make sure both are saved!), you have everything you need to test the operation of your design! The steps below will walk you through how to synthesize your HDL into a gate schematic, implement the design using the specific gates and resources available on the Basys 3, generate the bitstream – a special file format that programs the FPGA, and then test your circuit!
   - At this point, make sure your board is plugged into your computer -- the USB into the lab laptop, and the micro-USB into the matching "PROG" port on the board, and turned on (flip the power switch)! If it's on, you'll see the seven-segment display slowly count up.
   - Select "Run Synthesis"
   - What the heck is going on during? Check the status under the "Design Runs" tab, and you can get more information than you ever wanted from the log and reports
   - When Synthesis completes, you can expand the "Open Synthesized Design" header and then open the "Schematic" – *Take notes for your report:* does the schematic look like what you expected? Why or why not? What is familiar, and what is new? If you see anything unfamiliar, do some investigation on what it is and what role it may be playing, and include what you find in your report and any relevant sources in your collaboration document.
   - Next, select "Run Implementation"

- Next, select "Generate Bitstream". You can see it is still working if the "Generate Bitstream" option is greyed out and/or if you watch the spinning wheel on the "Design Runs" tab and/or if you look for updates in the top-right corner of the GUI. Generating the bitstream can take a long time, especially as we progress to more complex designs, so the instructor team suggests passing the time by singing a song in your head, asking a neighbor how their day is going, or reading ahead to the next steps for this lab if you are ever waiting. The step is doing the important translation from the generic schematic to the specific file format that "programs" the design onto the FPGA!

- If there are any errors, try double-checking your source files and checking out the console, but don't feel afraid to call over one of your wonderful TAs! We're here to help. If there are any errors, you'll have to generate the bitstream again! As you descend slowly into madness, eventually, everything will compile without errors, and we can get to the fun part.

- When you get a pop up that "Bitstream Generation successfully completed", do a little cheer, and get excited, and also select "Open Hardware Manager" – we will now send the bitstream to your plugged-in board.

- Under "Program and Debug" in the hardware manager, click on "Open Target." This will bring up a pop-up. Select "Auto-connect" to automatically connect to the board.

- This will open up the hardware manager and may change what windows and logs are visible on your screen. Focus back on the menu in the bottom left, where you should now be able to "Program Device." Click on that, and select your board (it may appear by its part number, which is "xc7a35t_0")—press program.

- When programming completes, your module has been configured onto the board! The input switches (V16, V17) now encode your two inputs, and the rightmost LED (U16) now encodes your output.

- Switch V16 and V17 both to 1 to see the LED light up. ***WOW!!!***

- When you are done marveling, close the Hardware Manager pane to return to editing your project sources and try out some other designs.

Part 2: Exploring & Testing Boolean Equations

Now, we'll go back into Vivado and edit your HDL code to use different logic gates. Test out an OR gate, a NAND gate, and an XOR gate. After configuring the SystemVerilog module and corresponding Constraints for each of the following examples, **repeat the steps in Part 1 to deploy and test that the design works as expected**.

1. Modify `gates.sv` to implement an OR gate instead of an AND gate. After you test that your circuit is operational, save a copy of your files (see "Lab 1 Save and Clean-Up")!

2. Take a look at the examples of `assign` statements in your textbook, and modify `gates.sv` to implement, in terms of AND, OR, and NOT, the behavior Y = A XOR B. After you test that your circuit is operational, save a copy of your SystemVerilog and Constraints files!

3. Modify `gates.sv` and `Constraints.xdc` to incorporate **three** inputs and **two** outputs, then implement the functions specified in DDCA Exercise 2.2.b (as output X) and 2.2.c (as output Y). Feel free to try out using SoP or PoS form, and to simplify your expression before implementing, or examine whether different gate schematics arise for different SystemVerilog definitions! After you test that your circuit is operational, save a copy of your SystemVerilog and Constraints files!

Lab 1 Save and Clean-Up!!

SAVE AT LEAST 5, up to 10 MINUTES TO SAVE AND SHARE YOUR WORK AND CLEAN UP

1. Save copies of your lab files
   - This will allow you to quickly recreate your Vivado project on another machine, and allows for easy grading for your instructor team!
   - Navigate in the File Explorer window to find your SystemVerilog file(s) (*.sv), likely within Users --> yourUsername --> yourProjectName --> yourProjectName.srcs --> sources_1
   - Copy them somewhere you can get to after you leave the lab laptop, along with your Constraints file (in the neighboring constrs_1 folder) and any other source files you developed or modified in this lab. These files are part of your submission.
   - ***As you make changes and additional designs,*** continue in the habit of regularly saving a copy of each improved version, with a file name that tells future-you what that version is (e.g., "gates_part1_and.sv" or "gates_part2_final.sv").
2. Clean up before you leave!
   - **Close the Hardware Manager in Vivado** by X-ing out the sub-window to safely disconnect from the FPGA board
   - Make sure up-to-date copies of your source files are saved off the laptop
   - Carefully unplug your board and re-pack into box
   - Exit Vivado and log out of the laptop
   - Leave your station neat
   - Return your laptop to cart for charging

Lab 1 Submission

Submit the following the same way you have for past labs:

- your SystemVerilog source file(s)
- your Constraints file(s)
- Written responses from Parts 0-2 **as a single PDF**, including any notes or visuals you'd like to include (e.g., drawings, schematic screenshots).
- Collaboration Document
- And a video, as described in class and on the slack channel
- (no notes from Pre-Lab this week)