ECE255 Lab 1 FALL 2019 100pts

MGT Room MK315

August 27, 2019

This is the handout for your first lab. Due to Labor Day M 2 Sept, the due-date for demo/report of Lab1 to GTA in MK224 is T-F 3-6 Sept & M 9 Sept 2019 in the time slot you signed up on the Canvas Page; however, this is the deadline, so start planning now and demo/rept the lab as soon as you're finished.

Lab1 is about logic design of a small schematic by the two methods we use in ECE255:

- VHDL code to describe the design
- block diagram to implement the design

Demo your work to TA and turn in a concise written report at the same time. See the handout **LabRept.pdf** for the format of your *short but organized* written reports.

The handout **Intro to ECE255 Labs Using Xilinx Vivado** in Canvas Module *Labs Info and Material* has info about the Digilent BASYS3 board, the Xilinx Artix-7 xc7a35t FPGA, and the Vivado software.

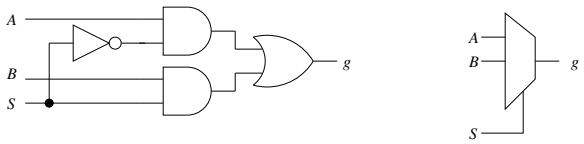

Figure 1: MUX: logic gates and icon

Lab1 gets you started with Vivado and the BASYS3 board by implementing the simple 2×1 `MUX` in fig. 1 four different ways. The last Part of the lab is to run the Vivado Simulator to simulate a MUX logic design and display the simulation waveforms.

For your user constraints in I/O Planning to implement the MUX, use

switch SW0 (pin `V17`)) for input S to the `MUX`

switch SW1 (pin `V16`) for input A to the `MUX`

switch SW2 (pin `W16`) for input B to the `MUX`

LED LD1 (pin `E19`) for output g from the `MUX`

Here're the five required parts in this lab.

Part 1: Use an existing BIT file to program the FPGA

The file `lab1mux.bit` on Canvas is a BIT file for the MUX using the I/O pins V17, V16, W16, E19 listed above. This first part of the lab is simply to connect your BASYS3 board, upload the BIT file to the board, and verify correct implementation of the MUX by applying all input combos on the switches.

Warning: A BIT file is not plain text ASCII for easy reading by a human. It's an encoded file that programs the FPGA, in this case, it programs the Artix-7 xc7a35t to implement the MUX using the I/O pins V17, V16, W16, E19.

1. Copy the file `lab1mux.bit` from Canvas to a convenient local folder (say to your local Desktop).

2. Connect your BASYS3 board to your computer with the USB-microUSB cable[1].

3. Since the BIT file already exists, you skip the logic design steps and simply upload the BIT file.

   Start up Vivado.

   Look for the `Task` menu on the leftside of the start-up window and select `Open Hardware Manager`.

4. Near the top of the new window, find `Open target` in the green bar. Click on `Open target`, then in turn

   Auto Connect
   `Program device` in green bar

   Fill in the path to your copy of BIT file `lab1mux.bit`. You can Browse by clicking the little box on the rightside of the line. **We will not use options for advanced debugging in ECE255, so leave `Debug probe` blank.**

   Finally, select `Program` to upload to your BASYS3.

   This programs the FPGA by transmitting the BIT file on the USB-microUSB cable.

5. Test the MUX on your BASYS3 by applying all possible combs of the inputs. How many combos are there? ■

Part 2: Write VHDL code for the MUX

1. Open a new project and follow all steps in Appendix A in the **Intro** handout to generate and upload the BIT file to a BASYS3. These steps include creating a new VHDL source file for the MUX.

2. Test the MUX implementation on your BASYS3. ■

Part 3: Block Diagram design using the XUP_LIB IP core for 2_to_1 MUX

Use the IP core called `XUP 2_to_1_mux` in the Xilinx library `XUP_LIB`. This core is one complete MUX, so you only need one instance of it in your block diagram.

1. Open a new project and follow the steps in Appendix B in the **Intro** handout.

2. The IP core `XUP 2_to_1_mux` has input bits `a,b,select` and output bit `y`. Put one instance of this core in your diagram and make each I/O bit an `external I/O port`.

   Your I/O ports must have the correct names for agreement with fig. 1; therefore, make `g` the name of the Output port and `S` the name of the select Input port.

   Explanation about VHDL keyword `select`

   The labels `a,b,select,y` on the `MUX` block are used in the Xilinx documentation for that block logic. But you must still give names to the external I/O ports you attach to these bits, and you cannot use `select` as the name of an I/O port.

   Why not? Because `select` is a keyword in VHDL. Every keyword in VHDL already has a specific usage in the language and cannot be reused for anything else.

3. Test the MUX implementation on your BASYS3. ■

Part 4: Block Diagram design using XUP_LIB IP cores for individual gates

Use `XUP_LIB` again, but this time implement the MUX with IP cores for individual AND, OR, INV (NOT) gates instead of `XUP 2_to_1_mux`.

1. Open a new project and implement a block diagram for the MUX in fig. 1 with one INV, two 2-input ANDs, and one 2-input OR.

[1]The BASYS3 has a 32Mbit nonvolatile serial Flash memory chip attached to the FPGA using a dedicated SPI (Serial Peripheral Interface) bus. The board-maker Digilent loads a Self-Test program for the FPGA into the Flash device during manufacture of the board. The Self-Test runs by default as soon as power is applied unless the user changes the default mode. Among other things, the Self-Test cycles the 7-segment displays, lights an LED above a switch when the switch is ON, and blanks a digit in the display when a pushbutton is pressed.

2. Make your external I/O port names `A,B,S,g` to agree with the figure.

3. Test the MUX implementation on your BASYS3. ■

Part 5: Use the Vivado Simulator

The Vivado Simulator is software that simulates a logic design without using the real FPGA. The simulator will show you waveforms of logic signals over the time period you specify; however, you must have a special VHDL program that defines the input waveforms to drive the simulation. This special VDHL code is called a *Test Bench file*.

Usually you have to write the Test Bench code for your logic design yourself, but a Test Bench file for a `MUX` has been put on Canvas for downloading. We'll discuss the format of Test Bench code in the future. For now, just note that its VHDL structure is the usual LIBRARY followed by ENTITY followed by ARCHITECTURE.

1. Reopen your project from Part 2 in which you implemented VHDL code for the MUX.

2. Download the VHDL file `tblab1.vhd` from Canvas to a convenient location (say to your local Desktop).

3. In `Project Manager`, add the Test Bench file to your project as a `Simulation` source.

4. Test Bench code has to contain the name of the entity to be simulated.

   The Test Bench file `tblab1.vhd` is written for an entity called `lab1mux`. If your entity has a different name, edit your copy of `tblab1.vhd` and replace `lab1mux` by your entity's actual name in two places:

   - after the keyword `component`
   - after `uut:`

5. This is a good time to double-check several things:

   - The Test Bench code must be a `Simulation` source, not a `Design` or `Contraint` source.
     It's your job to tell Vivado what kind of source the code is.
   - Be sure your `Project Settings` design language is `VHDL`.
     In most cases, Vivado will not automatically switch from Verilog to VHDL for you.
   - Be sure to change the *target entity* `lab1mux` in `tblab1.vhd` to *your actual entity name.*

     Warning:
     A successful simulation will input the Test Bench waveforms to your entity, simulate the entity at the hardware level, and plot the relevent input and output waveforms in the Vivado window for you to see.
     If there's an error in the set up of your simulation (for instance, not having your actual entity name in the Test Bench code), in many cases *the simulator will run anyway but simply plot waveforms that never change.*
     Then it's your job to find and correct the problem(s) and rerun the simulator.

6. Find `Simulation` in the Flow Manager and open Simulator Settings. In the lower pane, open Simulation and change the `runtime` to 60ns. This is the length of time the simulator will run before it starts a new simulation.

7. Select `Run Simulation` and then `Run Behavorial`.

   After a few seconds, you'll see the simulation waveforms in the Workspace pane. The hot icon `Zoom Fit` to the left of the pane will fit the complete waveforms to the window.

8. Look at the waveforms in detail and confirm correct functioning of the MUX. Print the waveforms and include with your written report. ■