ECE3829 Lab 1: Combinational Logic Design

ECE3829-A22

Getting Started and Decoder Tutorial:

Before starting this lab, you will need to complete the steps in the Getting Started Lab Guide and Decoder Tutorial to verify that your Basys3 hardware is operational, your Vivado Software is installed and running correctly and to learn how to run and create projects in Vivado. The current version is 2020.2. You will need a decoder lab 1, so the Decoder Tutorial feeds nicely into the start of your Lab 1 activities.

Important Material:

This lab will use the LEDs, slide switches, push button switches and seven-segment displays on the Basys3. Read these sections of the Diligent Basys3 reference manual to identify these components and understand how they work.
https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual

Reminder:

Remember to update your file header comment in each file and follow recommended coding guide lines presented in class, the specific implementation details defined in the lab and commenting guidelines listed in the grading rubric at the end of the lab.

All code should be written in Verilog for this lab.

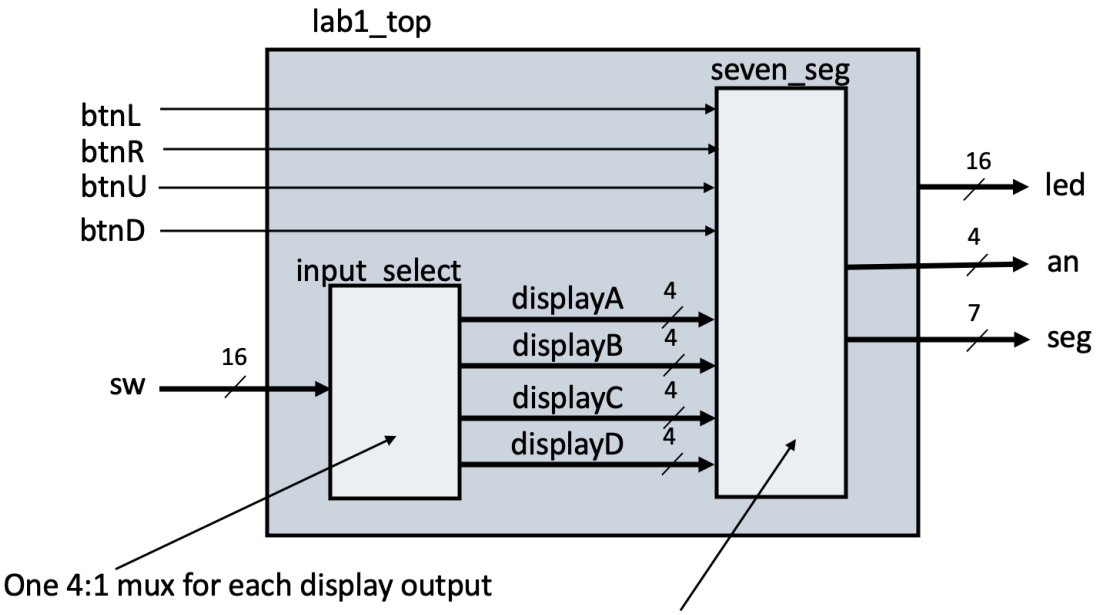
Late Submissions:

Please check Canvas for the day and time of each lab submission. Any assignment submitted after the deadline in canvas, will receive a 20% late deduction and can be submitted up to a week late. Unless special arrangements have been made in advance, no submissions after one week late will receive credit. Labs must be signed-off to receive credit for any part of the lab.

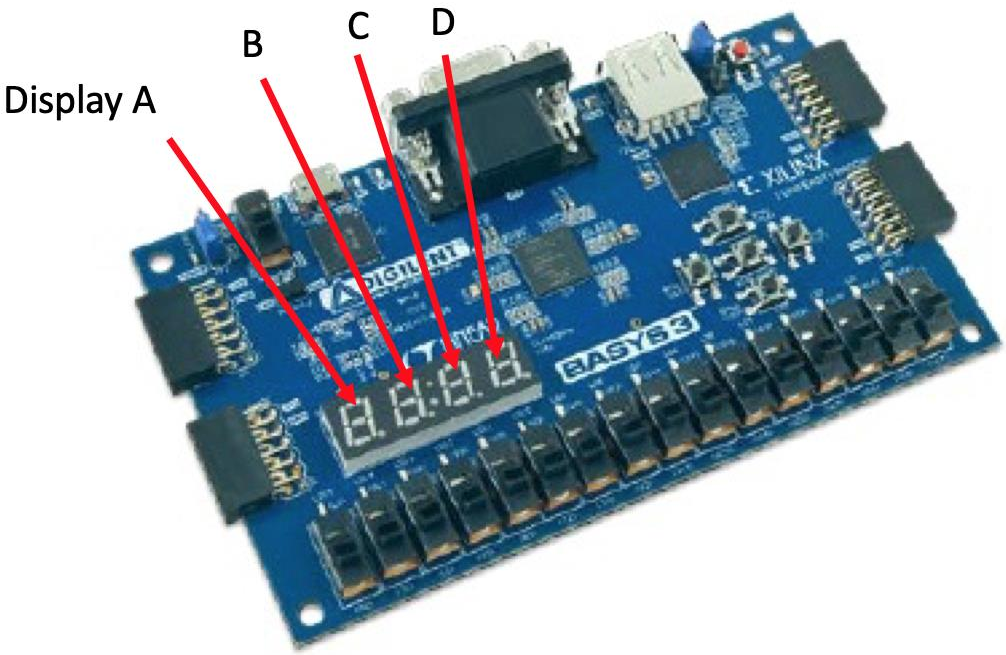
Overview

This lab uses the slider switches, push buttons, seven segment displays and LEDs on the Basys3 board. The Slider switch settings determine what value should be displayed on each of the four seven segment displays. Since all the displays share common seg signals, only one display value can be shown at a time. The push buttons will determine which one of the four displays to light. This lab uses only combination logic, there are no clocks. It requires the use of multiplexers and decoders. Everything you need to know is in Module 1.

The problem is broken down into two modules input_select and seven_seg. The input_select module calculates the data to display for each of the four displays. The seven_seg module determines which display is on and uses a decoder to convert the 4-bit input into the anode and segment controls for the 7-segment display.



The displays have been labeled A, B, C, D as shown below, with display A being on the left most of the board and display D being on the right most of the board.



Step 1: Create a submodule called input_select

Create a new module called input_select. It is common practice that your module and file name are the same.

The module should have the following ports

- One 2-bit mode select input
- One 14-bit slider value input
- Four 4-bit display value outputs.

This module uses sw[15:14] to sets mode of operation. sw[15] is the most significant bit and sw[14] is the least significant bit.

- When in mode 0, the last four digits of the ID on your WPI badge are displayed.
- When in mode 1, the displays show the values of the slider switches in hexadecimal; sw[13:12] on display A, sw[11:8] on display B, sw[7:4] on display C and sw[3:0] on display D. Example: SW[11:8] = 4'b1100 = 4'hC.
- When in mode 2, display A shows the value sw[13:12], display B shows the value of sw [11:8] and display C and D shows the value of sw[13:8] multiplied by two. This multiplication should be implemented by shifting the value to the left by 1, not by using the multiplication operator.
- When in mode 3, display A shows the value of sw[7:4] in hexadecimal, display B shows the value of sw[3:0] in hexadecimal and display C and D show the sum of sw[7:4] and sw[3:0]. When adding two 4-bit values, the sum will be a 5-bit value.

	A	B	C	D
Mode 0	WPI ID	WPI ID	WPI ID next to last	WPI ID last digit
Mode 1	sw[13:12]	sw[11:8]	sw[7:4]	sw[3:0]
Mode 2	sw[13:12]	sw[11:8]	times2 [6:4]	times2 [3:0]
Mode 3	sw[7:4]	sw[3:0]	sum[4]	sum[3:0]

Step 2: Create a submodule called seven_seg

Create a new module called 'seven_seg' in a file called seven_seg.v.

The module should have the following ports:

- Four 4-bit display inputs
  - The outputs of the input_select module should connect to these inputs.
- One 4-bit select input
  - Inputs from 4 push buttons
- One 7-bit segment output
  - Drives the seg signals on the 7-segment display
- One 4-bit anode output
  - Drives the anode signals on the 7-segment display

Only one display can be on at a time as they share the same seg pins. The module uses a 4-bit select input to determine which one of the displays is on and selects the corresponding display input value to be displayed. Four of the push buttons are used to select which of the four displays is illuminated. One button illuminates the display A, another button illuminates display B, and so forth. When no buttons are pushed or more than one button is pushed all segments of the display should be off. This is accomplished by setting all of the anodes to their off-logic level.

The module has a 7-bit output to drive the cathodes /seg signals of the display and a 4-bit output to drive the four anodes / an signals of the display. Parameter statements should be used to define the display segment values for each hexadecimal input value.

The structure of the module should have the following

- A mux to select which value to display
- A mux to select which display is lit using the anode signals
- A decoder to convert the four-bit input hexadecimal value to display segment values.

Step 3: Create a top-level module called lab1_top and create your constraints file

Create a top-level module called lab1_top. Use it to connect the pins named in the Master XDC file to the signals you need to access in your modules.

Instantiate your input_select and seven_seg modules with port connections made by name association.

The following connections should be made

- Assign the sw[15:0] inputs to their corresponding led[15:0] output pin, such that when the sw bit is logic high the corresponding led is turned on.
- Connect sw[15:14] to the mode select input of input_select module
- Connect sw [13:0] to the slider inputs of the input_select module
- Connect the four 4-bit display outputs of the input_select module to the display inputs of the seven_seg module. You will need wire statements to declare these internal wires.
- Connect four push buttons to the select inputs of the seven_seg module.
- Connect the segment output of the seven_seg module to the seg signals on the 7-segment display.
- Connect the anode outputs of the seven_seg module to the an signals on the 7-segment display.

The top-level port names should match the port names in the Master XDC file provided by Digilent. Ports with common string names like the sw ports should be consolidated into a single statement. For example, ports sw[15] through sw[0] in the XDC should be implemented in a single input statement 'input [15:0] sw,'.

- XDC File Link: https://github.com/Digilent/digilent-xdc/blob/master/Basys-3-Master.xdc

Using the master XDC as a starting point only uncomment or include statements for signals that are in use in your design and also the set_property statements for CFGBVS and CONFIG_VOLTAGE at the bottom of the file which are needed for all design.

Sign-Off Procedure

Every lab MUST be signed-off to receive credit for any part of the lab.

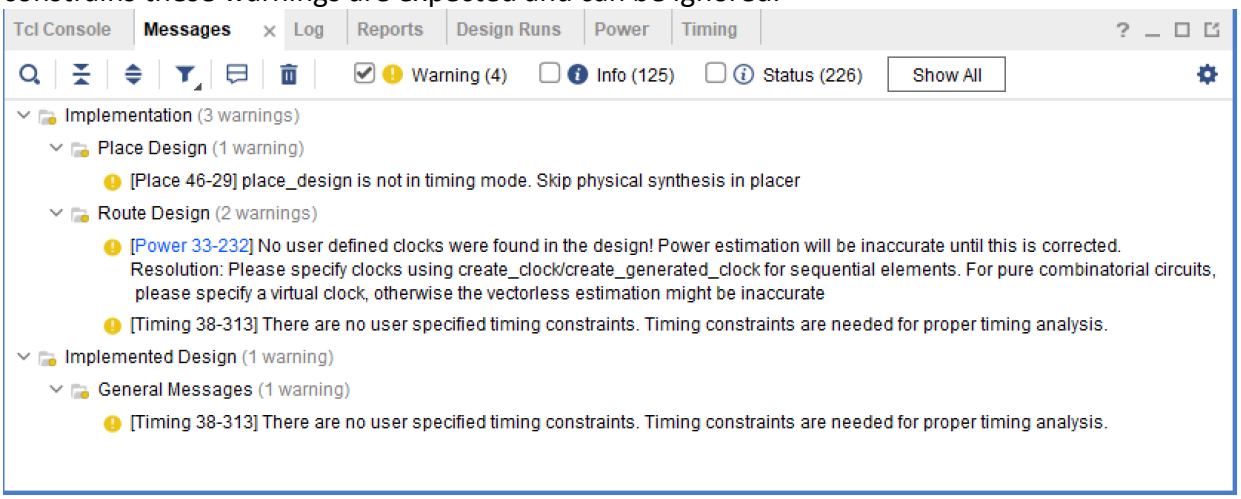
Signoff can be done in-person in AK113 or via zoom during any of the lab times or office hours shown on the Weekly calendar. If signing off remotely, you will need to have the ability to share your screen with the TA and also have video capability to display the Basys-3 board functions clearly. Please test the clarity of your video before signing-off and verify the LED locations and segments of the 7-segment display can be seen. This is often best done using the zoom app on your cell-phone.

You may repeat steps 1-4 as many times as needed. But you may only submit your archived project and pdf of your Verilog code once during your final demonstration. No changes to these submissions are allowed after your final demonstration.

1. Have your project fully Generated with Bitstream and be ready to your project to the TA.
2. Have your project archived and all your Verilog code in a single pdf file ready to submit.
3. In Vivado show your messages window with Warnings and Errors displayed to the TA. TA will verify that only expected warnings occurred. Some warnings are expected. However, points will be deducted if warnings or errors are generated by your code that are not expected.
4. Download your bit stream on to your board with TA present.
5. Demonstrate the functionality in the Functionality Check List. The TA will record results and assign points for all functionality that is correct.
    a. You can demo as many times as needed.
    b. You will always be rewarded points for the functions that are fully working.
6. Once you are happy with your demonstration results.
    a. You will submit your archived project in canvas immediately.
    b. You will submit the single pdf version in canvas immediately.
    c. TA will verify they are submitted and complete the sign-off process.

Expected Warnings

The first lab is a purely combinational design and has not clocks and therefore no timing constrains these warnings are expected and can be ignored.



Sign-Off Check List

Project Status (10 points)

1. Expected files present in project (3 points)
   - top_lab.v, seven_seg.v, input_select.v and constraints file present in project.
2. Vivado warnings (5 points)
   - Full credit no unexpected warnings.
3. Project programs part successfully(2 points)

Functionality Check List (35 points) / 5 points each item

1. LED[15:0] Operation:
   - When switch is pushed up, correct LED turns on. When switch is pushed down correct LED turns off.
2. Seven Segment Decode Operation:
   - In mode 1 verify each LED displays 0 through F with appropriate input (except 0 through 3 on display A).
3. Display Badge ID number in mode 0:
   - Check against badge value.
4. Verify Multiply by 2 in mode 2:
   - Left bits showing inputs and right bits showing x2 result.
5. Verify Addition in mode 3:
   - Left bits showing inputs and right bits showing sum.
6. Verify when no buttons pushed all displays off
7. Verify multiple buttons pushed all displays off

Canvas Submissions Completed (5 points)

1. Archived project submitted in canvas at time of sign-off
2. PDF of Verilog code submitted in canvas at time of sign-off

Rubric for Grading Verilog Code in PDF

1. Code complies with specific instructions detailed in the lab description. (10 points)
    - Modules have the ports specified in the lab description.
    - Seven_seg has 4 inputs and anode select muxes and a cathode decoder
    - Parameters are used to define the digit to segment mapping.
    - Seven_seg is organized with two muxs and decoder. .
    - Top level signal names should match names in Digilent constraints file.
    - Code has a top-level module which includes seven_seg and input_select modules, wire statements for internal signals and LED assign statements.
    - Modules and file have same name.

2. Code is well commented (5 points)
    - Each file should start with a commented header. Files generated with Vivado will create a template for you, fill in the values for your lab information. If not generated by Vivado refer to the decoder project example for format.
    - Each Input and Output should have an in-line comment as to what it does.
    - Similar to input and outputs, wire and reg statements should be commented
    - If using Parameters, a comment describing how the parameters are used and their values should be added.
    - Each set of concurrent statements should have a comment describing what the block of code does.

3. Code is well formatted, organized and written (10 points)
    - Code is well organized with separate assignments and always blocks for each logical function.
    - Signal names are descriptive but not to long.
    - Code is appropriately indented with only one statement per line.
    - Best Practices outlined in class were followed.