Lab 14

Traffic Light Controller

Description

Imagine that you are a controls engineer tasked with creating the controller for your company's new traffic light. Using the FPGA development board here in the lab, and circuit components from your Arduino kit (from ENGR 1041), you will develop a prototype implementation of the traffic light controller with a few different lighting patterns.

Procedure

Background: Light Patterns

- The traffic light you will be prototyping is designed for a standard four-way intersection.
  - As you would expect, the lights on the north and south sides will match each other, and the lights on the east and west sides will also match each other.
    - As a result, you will only need six LEDs to represent the traffic light. Correspondingly, we will be using six different signals, one for each LED and state (red, yellow, and green for both north-south and east-west).
- There will be four different patterns, controlled by three switches on the FPGA development board using a priority scheme.
  - The three rightmost switches, i.e., `switches(2 downto 0)`, will be used.
  - `switches(2)` will take priority over `switches(1)` which has priority over `switches(0)`.
- The highest priority pattern, activated with `switches(2)`, is the "four-way stop" pattern. All sides will flash red, with a period of 2 seconds and duty cycle of 50% (i.e., 1 second on, 1 second off).
- The second highest priority pattern, activated with `switches(1)`, is the "north-south stop" pattern. The north-south sides will flash red as before, but the east-west sides will flash yellow (with the same period and duty cycle).
- The third highest priority pattern, activated with `switches(0)`, is the "east-west stop" pattern. It is the converse of the previous pattern. The north-south sides will flash yellow, and the east-west sides will flash red.

- The default pattern, activated when all switches are off, is the standard traffic light pattern defined by the table below.
  - This pattern has a period of 100 seconds, and repeats as long as it is activates.
  - The three-letter abbreviations in this table use NS to represent north-south and EW to represent east-west. The third letter is either R (for red), Y (for yellow), or G (for green).

Time	NSR	NSY	NSG	EWR	EWY	EWG
40 sec	`1`	`0`	`0`	`0`	`0`	`1`
10 sec	`1`	`0`	`0`	`0`	`1`	`0`
40 sec	`0`	`0`	`1`	`1`	`0`	`0`
10 sec	`0`	`1`	`0`	`1`	`0`	`0`

Project Creation

1. Download `lab14.vhd` and `lab14.xdc` from Canvas and place them in a new folder titled `lab14`.
2. Open Vivado and create a new project titled appropriately.
   - For your convenience, the board identifier is `xc7a35wtcpg236-1`.
     - You can also find the board with the same options as before: `General Purpose`, `Artix-7`, `cpg236`, `-1`.

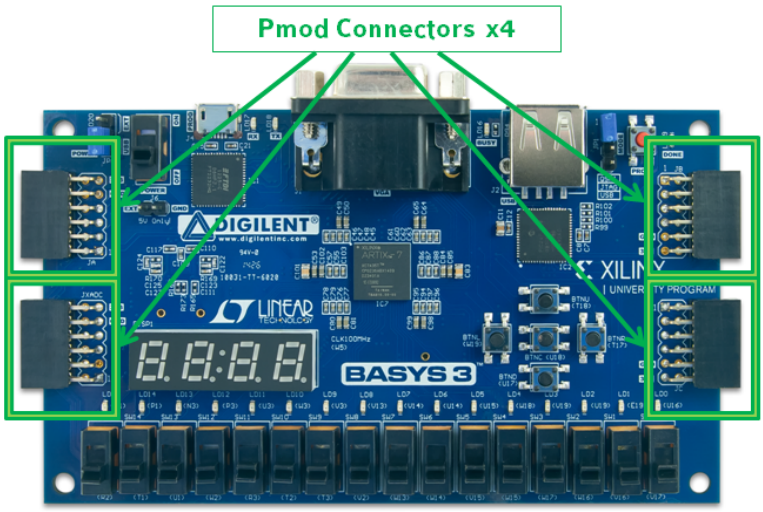
Hardware Review and Upload

3. Open `lab14.vhd` in the editor, and follow along with the instructor to review the hardware description.
4. Next, open `lab14.xdc` and review it alongside the instructor.
   - For this lab, understanding the pinout is critical since you will be connecting wires directly to the FPGA's PMod headers. If done incorrectly, this could damage the FPGA.
5. `Open Elaborated Design` and review the schematic alongside the instructor.
6. Run synthesis and implementation and then generate the bitstream to program the device.
   - While Vivado is running, you should get started building the circuit with Step X in the next section.
7. Once the bitstream has been generated, open the Hardware Manager.
8. Plug the FPGA development board into your computer using the provided Micro-USB to USB-A cable.

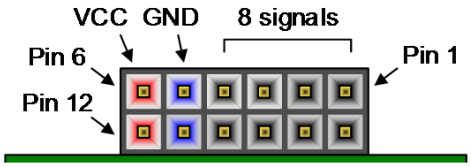
9. At the top, in the green banner (or under the `Open Hardware Manager` dropdown) click `Open Target`, then `Auto Connect`.
10. Once the device is connected, select `Program Device` (in either of the locations where `Open Target` was previously).
    - See 06B-handout for troubleshooting steps, or ask the instructor or TA.
11. Click `Program` to upload the bitstream to the FPGA.

Physical Circuit Construction

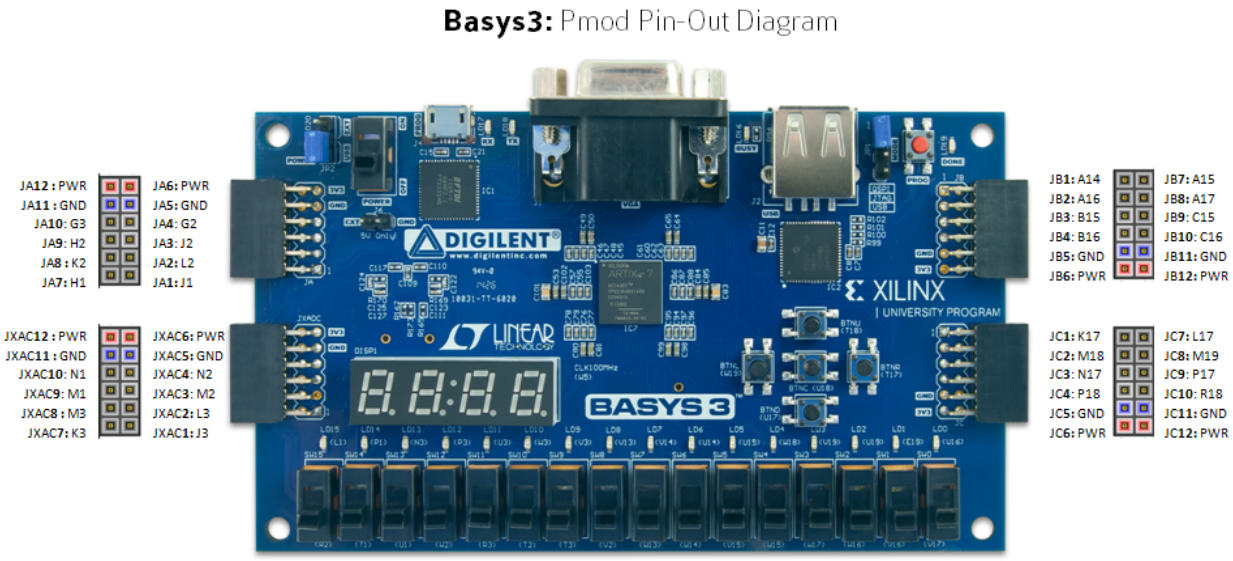
12. Before constructing the circuit, you must learn how the FPGA's PMod (Peripheral Module) connectors work.
    - The Digilent Basys 3 FPGA Development Board has four sets of PMod connectors:



    - We will be using the `JC` set, on the bottom right side of the board.
    - Each set has the following pins:

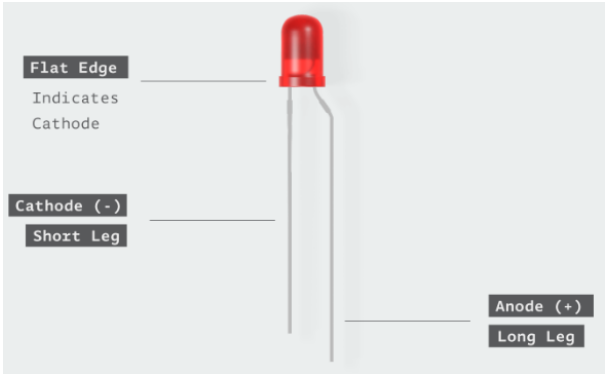


- In the constraint file, you will see the pin labels below:



- Under no circumstances are you to connect either of the VCC pins directly to either of the GND pins.
- In this section, you will use six of the eight signal pins, along with two of the ground pins connected to the ground rails on your breadboard, to control six LEDs representing the traffic light.

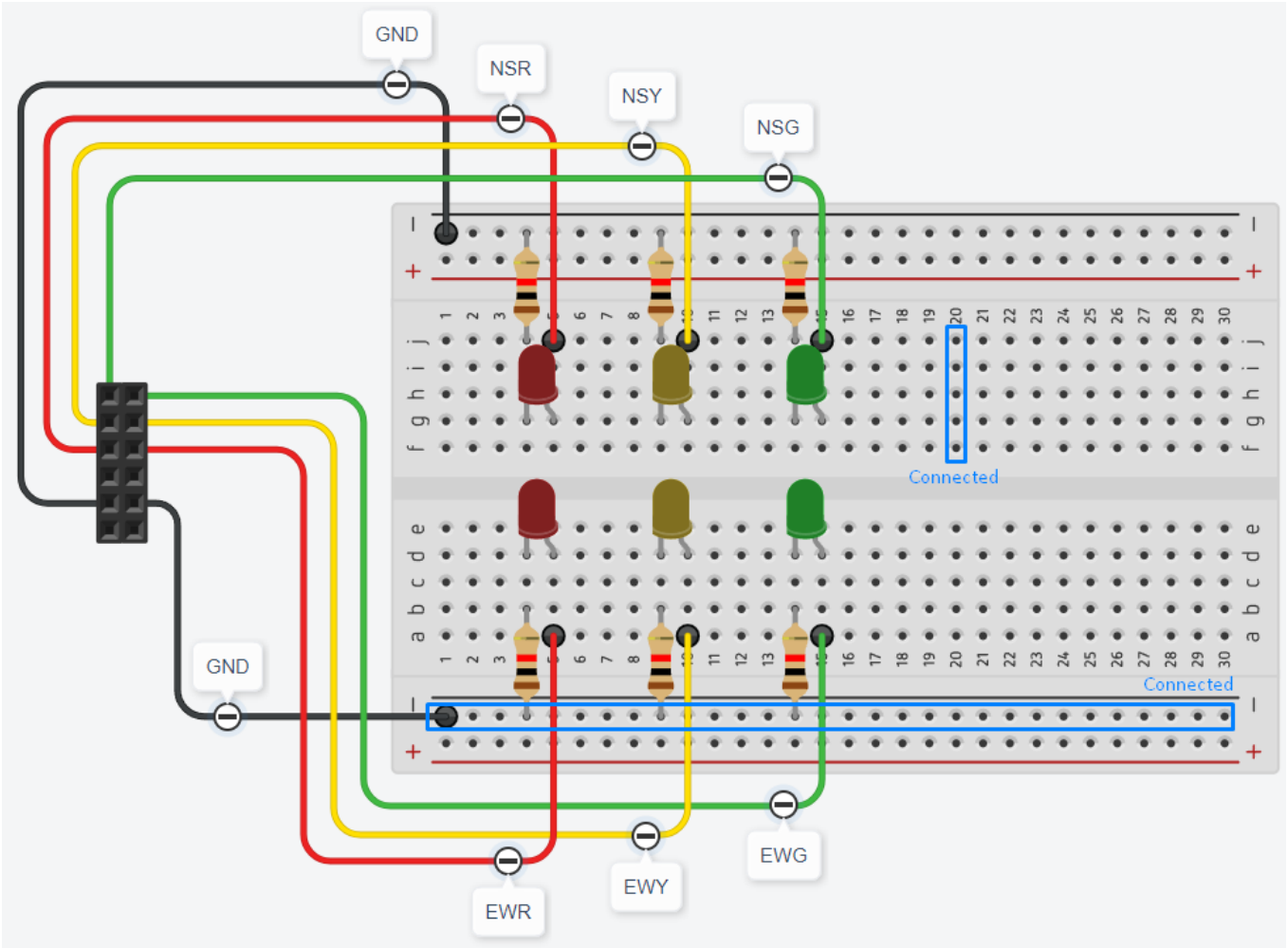
13. Before constructing the circuit, you also need to know how LEDs must be configured, to avoid immediately breaking them.
    - LEDs have one short leg, which is the cathode (negative), and one long leg, which is the anode (positive), as shown below (source):



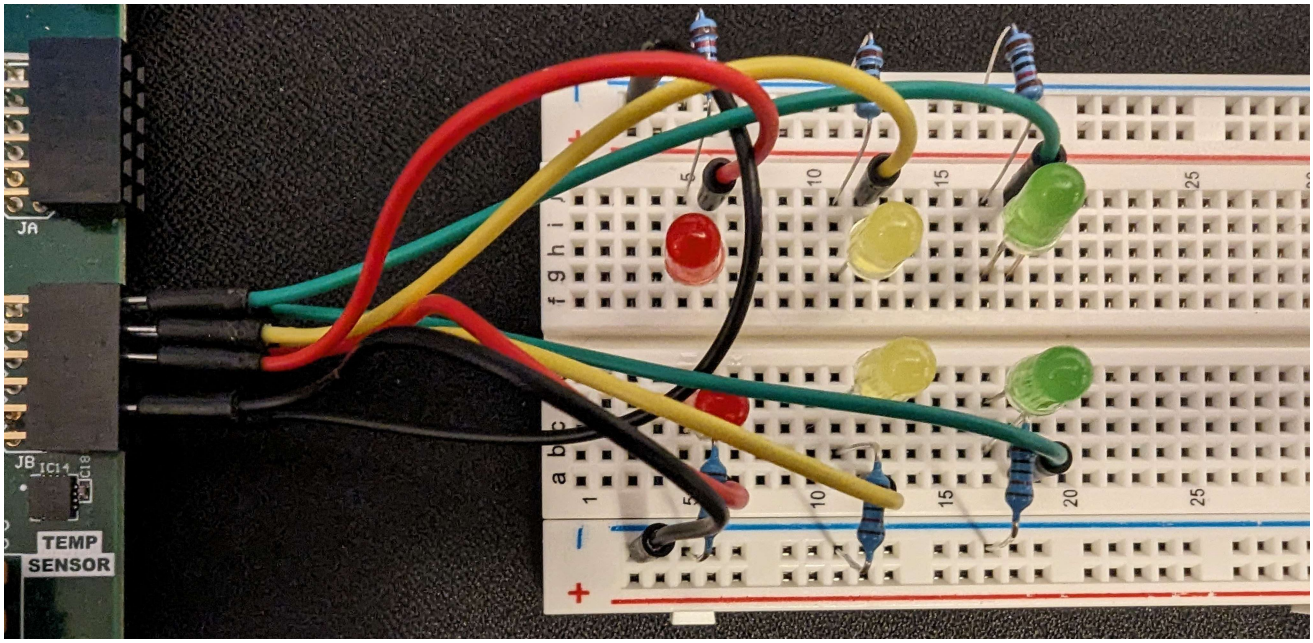
    - On some LEDs, the cathode has a flat edge (this may not be the case for yours).
    - Current must enter the LED through the anode and exit via the cathode.
    - LEDs also must be placed in series with a resistor to avoid shorting the LED and microcontroller (in this case, the FPGA).
    - Thus, we will place a resistor from the cathode to the ground rail of the breadboard, and connect the PMod header pin to the anode.

14. To construct the circuit, you will need to retrieve the following from your Arduino kit:
    - A breadboard
    - (8) Male-to-Male Jumper Wires – which look like this:

    - (6) LEDs, ideally 2 each of red, yellow, and green
    - (6) Resistors (any that you have in your Arduino kit should suffice)
15. Construct the circuit according to the diagram below. In the diagram, the "left column" of the black connector corresponds to the top row of the PMod header on the FPGA. Similarly, the "right column" corresponds to the bottom row.



- In short, connect the two ground rails of the breadboard to the two ground pins on the PMod header. Then, connect the north-south LEDs to the top row and the east-west LEDs to the bottom row, starting from the right side of the PMod header with green, then yellow, then red.
- Note that the ground rail (the entire line on the breadboard marked with the negative sign and black line) is connected, as shown above. Each five-pin-socket row is also connected as shown.
- Your final product should look roughly like this (ignore the usage of a different board and PMod header):



Hardware Testing and Analysis

16. If you have not already done so, connect the jumper cables to the appropriate pins on the PMod header, and program the FPGA.
17. Check that the controller is going through the different patterns of lights and giving priority to the switches as described previously.
18. Discuss why there are 35 flip-flops used in this design – what is each used for?

Course and Lab Survey

19. Please complete the course and lab survey available here: https://forms.gle/1yp8iEL374DB16dZ8

Deliverables

- You are required to have fun for this lab.

Outcomes

- Practice working with VHDL.
- Practice using Vivado for hardware synthesis and implementation.
- Practice programming and testing a hardware description on an FPGA development board.
- Understand how to use an FPGA development board to control external devices.
- Understand how a state machine can be implemented on an FPGA using VHDL.