LAB 6: Sequential Circuit Design: Traffic Light Controller

Objective: In this lab, you will design a traffic light controller that takes input from switches on your DE10-Lite Board and displays traffic signals on a VGA screen.

Prelab

1. Draw a very-well-thought-out detailed state diagram for the traffic light controller finite-state machine specified below. Bugs are far easier to find and fix in a state diagram than verilog code so a carefully-designed state diagram will make your lab go far more smoothly and quickly.
2. Write pseudo code to implement the various timers that will be needed for your design. These timers must be able to measure accurate time delays of 1, 3 and 6 seconds.

Introduction

In this lab, you will display the traffic lights for an intersection on your VGA monitor. To do this lab, you will need the Quartus Project in lab6.zip found on the course website.

Traffic Light Controller Specifications

The traffic light controller for this lab is based on a very simplified version of the system that controls the intersection at La Rue Rd and Orchard Rd, near the campus entrance by the ARC. We have removed the left turn arrows and the count-down pedestrian timer to simplify the design. The inputs to your system are as follows:

- KEY[0] – system reset pushbutton
- KEY[1] – pedestrian pushbutton
- SW[0] – car sensor for Green light on La Rue Rd.
- SW[1] – car sensor for Green light on Orchard Rd.

Your traffic light controller must meet the following specifications:

1. At system reset, the system will start with a Green light on La Rue, a Red light on Orchard, and Don’t Walk for all pedestrians.

2. After any Green light, there must always be a Yellow light on for 3 seconds before turning on the Red light.
3. While a light is green or yellow, the perpendicular street’s light must be red, and the pedestrian Don’t Walk signal must remain on.

4. When the Green light on La Rue is on:
   a. If no other sensor (Orchard car sensor or pedestrian request) is active:
      • Green light on La Rue stays green indefinitely
   b. If another sensor is active
      • [Light has been green 0-3 seconds] Stay green
      • [Light has been green 3-6 seconds]
        If La Rue car sensor is active Stay green
        else Immediately begin to take care of request
      • [Light has been green 6+ seconds] Immediately begin to take care of request

5. The cycle for servicing pending requests is:
   a. La Rue Green (reset state)
   b. Pedestrian Walk (all directions)
   c. Orchard Green
   Your system should follow this cycle, but it should be demand-driven; it should respond to requests in the order of this cycle, but it should skip a state if the corresponding request is not active. Thus, if there is a car waiting on Orchard, but no pedestrian request, your system should go directly from Yellow on La Rue and to Green on Orchard, skipping the pedestrian walk state.
   When there are no pending requests from any sensors, your system should return to the reset state of Green on La Rue, Red on Orchard, and Don’t Walk for pedestrians.

6. When the Green light on Orchard is on:
   a. If no other sensor (La Rue car sensor or pedestrian request) is active:
      i. [Light has been green 0-3 seconds] Stay green
      ii. [Light has been green 3+ seconds]
        If Orchard car sensor is active Stay green
        else Begin changing to La Rue Green
   b. If another sensor is active
      • [Light has been green 0-3 seconds] Stay green
      • [Light has been green 3-6 seconds]
        If Orchard car sensor is active Stay green
        else Immediately begin to take care of request
      • [Light has been green 6+ seconds] Immediately begin to take care of request

7. For the pedestrian crossing, the lights on La Rue and Orchard should both be Red and:
   a. the pedestrian Walk signal should be On for 6 seconds followed by
   b. the Don’t Walk signal should cycle On (1 second) and Off (1 second) for 6 seconds total.
   After the 6+6 seconds, the Don’t Walk signal should remain constantly On and the controller can service the next request.
8. Above all, your system must create a safe intersection. At most, one signal should be non-Red at any time. For each set of lights, one, and only one, light must be on at any time (with the exception of when the Don’t Walk light blinks off). A light should never “glitch” on and off – the minimum duration for any light, other than the 1 second blinking of the Don’t Walk signal, is 3 seconds.

Design Guidelines and Hints

1. All flip-flops, registers and counters must be clocked directly using the 25 MHz system clock. You should generate appropriate enable signals in order to update your registers or counters only on certain clock pulses.

2. The reset pushbutton should generate a synchronous reset to your finite-state machine, flip-flops, registers and counters.

3. The pedestrian pushbutton switch must be clocked into a flip-flop so that the system “remembers” that the pedestrian button has been pressed. The pedestrian should only need to press the pushbutton once for his or her request to be recorded – he or she should not need to hold the pushbutton until the request is serviced. Your controller should clear the pedestrian flip-flop at reset and after the pedestrian request has been serviced.

4. The switches serving as car sensors (SW[0] – SW[1]) should also be clocked into flip-flops to synchronize the requests to the system clock. These slide switches correspond to sensors under the highway. If a car sensor is activated, but is later de-activated before the request is serviced, your system can ignore the request. For example, a car may briefly activate a sensor but then make a legal right turn on Red. The sensor will then be deactivated and that request can be ignored even in the case of a light transitioning from green to yellow to green. However, if a Green light is turned on, it must stay on for the minimum 3 seconds even when the corresponding car sensor is no longer active.

Design Requirements

You must write a testbench for your module. Simulate the behavior of your traffic light controller for the various input conditions. The simulation must print the current state to the console using the $display command whenever the state changes.

Once you are satisfied that your design is working in simulation, compile the project in Quartus and program the DE10-LITE board. Once the board is programmed, press the reset pushbutton (KEY[0]). Connect the DE10-LITE board to your monitor using the VGA cable. The traffic signals should be displayed in the reset state. Verify the operation of your traffic light controller using the switches and pushbuttons on the DE10-Lite board.

Checkoff: Show your working testbench and implementation on the DE10-LITE board to your TA and have them sign off.
Submitted Work

[20 pts] **Prelab** – Solutions to prelab exercises.

[70 pts] **Lab Checkoffs**

A) Demonstrate your testbench to your TA and show your design successfully simulates.

B) Demonstrate the final design on the DE10-LITE to your TA. The final result should implement the traffic signals on the VGA screen according to the design specifications.

[10 pts] **Lab Report**

Submit *all Verilog hardware and testbench code that you wrote. Do not include any code that you did not write.* Uploading your verilog is essential to receive credit for the entire lab.

A) [5 points] Print and submit during your lab session.

B) [5 points] Upload to Smartsite using the following steps by the end of your lab session:

1. Make a folder on your computer
2. Copy all verilog files you wrote into the folder—only the ones you wrote
3. “zip” the folder into a single .zip file
4. Log onto Smartsite, click *Assignments* on the left side of the page, click on the correct lab number
5. Upload the .zip file as instructed

Updates:

2017/05/18  Added small clarification to Design Guideline and Hint #4.