ECPE 174 – Lab 1
Moore Finite State Machines

I. Objectives

Design and implement a synchronous, Moore-based finite state machine that mimics the drive circuitry for the taillights of an old car.

II. Problem Statement

Three input signals control the operation of the taillights. The input signals are LEFT, RIGHT, and HAZARD. The output signals LA, LB, and LC drive the left taillights, and the signals RA, RB, and RC drive the right taillights. All lights should be off (idle state) when none of the inputs is HIGH. When HAZARD=1, all six lights should flash on and off in unison. When LEFT=1, the left side taillights should flash according to the repeating pattern LB on, LA and LC on, all lights on, all lights off, and then back to LB on. When RIGHT=1, the right side tail-lights should flash according to the repeating pattern RB on, RA and RC on, all lights on, all lights off, and then back to RB on.

Both LEFT and RIGHT cannot be 1 at the same time. If a switch from LEFT to RIGHT (or vice-versa) occurs in the middle of a flashing sequence, the network should go to the idle state (all lights off) and then begin the new sequence. HAZARD takes precedence over LEFT or RIGHT if either of them is asserted at the same time as HAZARD.

The state machine should be driven by a clock signal that runs at a frequency equal to the desired flashing rate (which will be given in class).

Create the controller for this lock using a Moore FSM. Connect the input keys to SW0, SW1, and SW2. Output the LEFT lights using LEDR0-LEDR2; output the RIGHT lights using LEDG0-LEDG2.

III. Pre-Lab

Complete the following steps before lab and turn in by 2pm:

1. Generate a state table and state machine diagram.
2. Generate a state assignment table
3. Design a circuit based on D flip-flops that implements the state machine using minimal hardware:
   a. If your design cannot be easily hand-implemented, please provide the following:
      i. Explain why: what number of variables do you need, how does this limit your K-map implementation, etc.
      ii. How many FFs will you need to implement the design?
   b. If you figure out a design that does allow you to solve the K-maps and implement it, provide that solution for extra credit (2 points, or 10% of your lab grade). Provide the equations; there is no need to draw the circuit. Also specify the number of gates required.
4. Implement the design using behavioral VHDL (see 8.29 on pg 509 as a guideline).

5. Simulate your code using the waveform editor and functional simulation. Turn in image of simulation waveform. Please use buses to group related signals instead of individual wires.

Submit electronically via Sakai using your personal Drop Box or the assignment if possible. Otherwise submit a clean, handwritten copy by 2pm in class.

IV. In-Lab

Complete the following steps in lab:

1. Examine the State Machine Viewer output and save the image.
2. Ensure Quartus is using a minimal bit representation for the states:
    a. Go to Assignments -> Settings
    b. Click on Analysis & Synthesis
    c. Click on More Settings
    d. Choose "State Machine Processing"
    e. Change it from "Auto" to "Minimal Bits" (and note the other options available)
3. Recompile the design. Re-examine the State Machine Viewer output and save the image.
4. Implement the VHDL design on the Cyclone II.
5. Demonstrate it for check-off of the lab.
6. Record information necessary for lab report.

V. Post-Lab

Discuss the following your lab report:

- What, if any, difference is there between the two State Machine Viewer outputs? How do they compare to the one you designed in pre-lab step 1?
- What equations did the synthesis tool use to implement the design? How does this compare with your results and ideas outlined in pre-lab step 3?
- Outline the results of the fitter resource usage summary: how much space on the FPGA does the design require? What types of elements were used?
- How fast could you clock the design according to Quartus? What happens if you attempt to clock it at that speed? Faster than that speed?