University of California, Davis
Department of Electrical and Computer Engineering

EEC180B DIGITAL SYSTEMS II Spring 1999

Lab 5: State Machine Design in VHDL

Objective: In this lab you will design a simple state machine in VHDL. After verifying the design by simulation, you will synthesize your code and implement your design in a Xilinx FPGA.

Pre-lab: Draw the state graph (i.e. state transition diagram) for the state machine based on the design specifications.

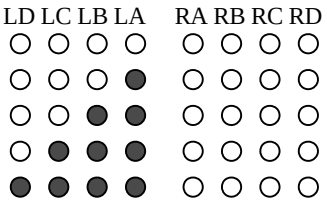
Note: this lab and pre-lab are to be done individually. That is, you may not work with anyone else or use anyone else's work.

Problem Specification

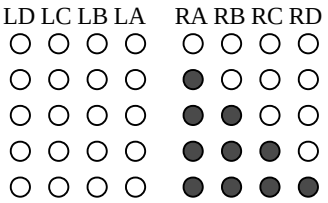
This exercise is based on Problem 3.2 in your textbook. However, the number of lights has been increased to 8 in order to correspond to the Xilinx demo board and several other details have been modified.

An older-model Thunderbird car has *four* left and *four* right tail lights, which flash in unique patterns to indicate left and right turns.

Left-turn pattern:



Right-turn pattern:



The network has three inputs, LEFT, RIGHT, and HAZ. When HAZ=LEFT=RIGHT=0, the network should be in the IDLE state (all lights off). As indicated above, when LEFT=1 the lights flash in a pattern starting with all lights off, then LA on, LA and LB on, LA, LB and LC on, and finally LA, LB, LC and LD on. When RIGHT=1, the light sequence is a similar pattern with lights RA, RB, RC and RD. If LEFT changes to 0 during a left-turn sequence or RIGHT changes to 0 during a right-turn sequence, the network should go to the IDLE state on the next clock. If HAZ=1 *or* if both LEFT=RIGHT=1, then all eight lights should flash on and off in unison. HAZ takes precedence if LEFT or RIGHT is also on. If the network is in the midst of a left or right turn sequence and the switches are changed so that HAZ=1 or LEFT=RIGHT=1, the network should go to the IDLE state on the next clock and then begin flashing all eight

lights. Assume that a clock signal is available with a frequency equal to the desired flashing rate.

In addition to the inputs already mentioned, the network also has CLOCK and RESET inputs. When RESET=0, the network should *asynchronously* go into the IDLE state.

Design a Moore sequential network to control these lights. (Note: the LED's on the Xilinx board are active-low.) The entity of the VHDL model should be as follows:

```
entity thunderbird is
port (clk, reset, left, right, haz : in std_logic;
        led : out std_logic_vector(7 downto 0));          -- led(7) is LD; led(0) is RD;
end thunderbird;
```

Lab Requirements

1. Draw the state graph. (Pre-lab)
2. Write a *synthesizable* VHDL model of the sequential machine. Use the standard 2-process model, with one process for the state registers and another for the next-state and output combinational logic. (If you desire, you can use more than one process to model the combinational logic.)
3. Write a test bench to verify the operation of the circuit. The test bench should at least generate the clock and reset signals. You can control the logic inputs directly through the debugger if desired. Print the waveforms from your functional simulation. Demonstrate your simulation to your TA for verification.
4. Copy the attribute file, /afs/ece/eec180b/lab5/thunderbird.attr, to your project directory. This file contains the pad assignments for the input and output signals specified in the entity. The file must have the same name as your Xilinx netlist file (.sxnf) produced by the Synopsys Design Compiler.
5. Copy the example synthesis script file, /afs/ece/eec180b/lab5/lab5.scr, to your project directory. You will need to modify the script for your source file, entity, and architecture names. Note: The Synopsys tools *are* case-sensitive so you must give the exact names of the source file, entity and architecture in the script file.
6. Synthesize the sequential machine for the Xilinx library. Make sure you are using the .synopsys_dc.setup file for the Xilinx library. Print the gate-level schematic from the Synopsys Design Analyzer.
7. Examine the synthesis log file and make sure you did not infer latches in your design. The log file should report how many flip-flops were inferred.
8. Run the Xilinx Design Manager and generate the bit file for your design.
9. Download the design and verify its operation. Demonstrate your circuit to your TA for verification.

Lab Report

Submit the following items in your lab report: (Due in your first lab period of the following week - i.e. one week after the start of the lab.)

- All VHDL source code.
- Test bench source code.
- Simulation waveforms from your functional simulation
- Gate-level schematic of your finite state machine printed from the Design Analyzer.
- Signed TA verification for the simulation and final implementation.
- From the various Xilinx report files, record the number of flip-flops and CLBs used in your design.