**Joshua Pollock**

**EE 310 – Lab 1 Report**

**NAU, 9 February 2020**

**Problem Description**

In this lab, we have been asked to design a state machine to display 1st, 3rd, 5th and 7th digits of my student ID on 7-segment displays. On the reset state, all displays will be off. Then at every clock cycle, one of the displays will be turned on to display the correct digit of my student ID. After 4 clock cycles/states, all displays will be turned on to display all 4 digits. In the next clock cycle, the state machine will return to its initial state where all displays are off.

|  |
| --- |
|  |

Figure 1. Expected behavior of the circuit

**Solution Plan**

In order to solve the problem explained above, I have created a program that will keep track of the 5 different states used to switch the correct displays on and off. Using the “Moving digits state machine” code that is provided on BBLearn, I was able to repurpose much of this code to create my very own. The code will keep track of 5 different states, where the first four enable only one display and number and the last state enabling all displays. There is another state that is not clock based in this lab. This is the reset state. When the reset state is active, all of the displays are disabled until the reset state is disabled. To know which display to turn off and which bits to toggle in the seven segment displays, I used the “7-segment decoder for moving digits state machine” file, again on BBLearn. This file contained all the different bit combinations for the display from 0 to F.

|  |
| --- |
|  |

Figure 2. State diagram for the proposed solution

**Implementation and Test Plan**

I have implemented the solution plan explained above, by first completing the prelim of the lab, creating the main program and a testbench to simulate the program. This was done using the resources on BBLearn and repurposing them for the lab deliverables. Using this code, I was able to get the simulation seen in Figure 4, completing the prelim of the lab. Next the Baseline\_C5GX.V file needed to be completed to have the Lab1 file work with the FPGA. This was as simple as adding a line provided to us in the lab description. Some edits had to be made to this line, such as switching the reset input to SW[0]. The other change that had to be made was adding HEX2 to the line instead of HEX. This is likely a typo that occurred in the lab description. Once the Baseline file was created, I plugged in the FPGA and was able to compile and run it on the board. It worked quite flawlessly and made the lab extremely short for me. I showed it to the GTA of the lab and was checked off as completed. Overall, I am extremely satisfied with how this lab turned out and how quickly it was completed. The prelim portion of the lab made the actual in class portion of the lab lightning fast.

|  |
| --- |
| **Lab1.v**  module lab1 ( rst , clk , ss3 , ss2 , ss1 , ss0 ) ;  input rst, clk;  output reg [6:0] ss3, ss2, ss1, ss0;  localparam [2:0] state\_\_\_\_\_ = 0 ,  state1\_\_\_\_ = 1 ,  state\_2\_\_\_ = 2 ,  state\_\_3\_\_ = 3 ,  state\_\_\_4\_ = 4 ,  state\_\_\_\_5 = 5 ;  reg [2:0] state ;  always @(posedge clk) begin  if ( rst ) begin  state = state\_\_\_\_\_ ;  end  else begin  case ( state )  state\_\_\_\_\_ : begin  state = state1\_\_\_\_ ;  end    state1\_\_\_\_ : begin  state = state\_2\_\_\_ ;  end    state\_2\_\_\_ : begin  state = state\_\_3\_\_ ;  end    state\_\_3\_\_ : begin  state = state\_\_\_4\_ ;  end    state\_\_\_4\_ : begin  state = state\_\_\_\_5 ;  end    state\_\_\_\_5 : begin  state = state\_\_\_\_\_;  end    default : begin  state = state\_\_\_\_\_ ;  end  endcase  end    //5072    case ( state )  state\_\_\_\_\_ : begin  ss3 = 7'b1111111 ;  ss2 = 7'b1111111 ;  ss1 = 7'b1111111 ;  ss0 = 7'b1111111 ;  end  state1\_\_\_\_ : begin  ss3 = 7'b0010010 ;  ss2 = 7'b1111111 ;  ss1 = 7'b1111111 ;  ss0 = 7'b1111111 ;  end  state\_2\_\_\_ : begin  ss3 = 7'b1111111 ;  ss2 = 7'b1000000 ;  ss1 = 7'b1111111 ;  ss0 = 7'b1111111 ;  end  state\_\_3\_\_ : begin  ss3 = 7'b1111111 ;  ss2 = 7'b1111111 ;  ss1 = 7'b1111000 ;  ss0 = 7'b1111111 ;  end    state\_\_\_4\_ : begin  ss3 = 7'b1111111 ;  ss2 = 7'b1111111 ;  ss1 = 7'b1111111 ;  ss0 = 7'b0100100 ;  end  // All  state\_\_\_\_5 : begin  ss3 = 7'b0010010 ;  ss2 = 7'b1000000 ;  ss1 = 7'b1111000 ;  ss0 = 7'b0100100 ;  end    default : begin  ss3 = 7'b1111111 ;  ss2 = 7'b1111111 ;  ss1 = 7'b1111111 ;  ss0 = 7'b1111111 ;  end  endcase  end  endmodule  **Lab1\_tb.v**  module lab1\_tb ;  reg clk\_tb , rst\_tb ;  wire [6:0] ss3\_tb , ss2\_tb , ss1\_tb , ss0\_tb ;  localparam PER = 20 ;  lab1 dut (rst\_tb , clk\_tb, ss3\_tb , ss2\_tb , ss1\_tb , ss0\_tb ) ;  always begin  clk\_tb = 0;  #(PER/2) ;  clk\_tb = 1;  #(PER/2) ;  end  initial begin  rst\_tb = 1 ;  #PER ;  rst\_tb = 0 ;  #PER ;  #PER ;  #PER ;  #PER ;  #PER ;  #PER ;  $stop ;  end  endmodule  **baseline\_c5gx.v (Only one line added)**  lab1 dut ( SW[0] , KEY[0] , HEX3 , HEX2 , HEX1 , HEX0 ) ; |

Figure 3. Verilog code for the proposed solution

|  |
| --- |
|  |

Figure 4. Lab pictures of the running solution