ECE 3544: Digital Design I
Project 3 (Part B): Design and Synthesis of a Synchronous Finite State Machine

Student Name: **Jonathan Lemarroy**

Honor Code Pledge: I have neither given nor received unauthorized assistance on this assignment.

Jonathan Lemarroy

Grading: The design project will be graded on a 100 point basis, as shown below:

Manner of Presentation (30 points)

_________ Completed cover sheet included with report (5 points)

_________ Organization: Clear, concise presentation of content; Use of appropriate, well-organized sections (15 points)

_________ Mechanics: Spelling and grammar (10 points)

Technical Merit (70 points)

_________ General discussion: *Did you describe the objectives in your own words? Did you discuss your other conclusions and the lessons you learned from the assignment?* (5 points)

_________ State machines: *Do your state diagram and timing diagram for the* buttonpressed *module correctly represent its behavior? Did you address the questions on structuring state machines in Verilog?* (10 points)

_________ Implementation discussion: *Did you discuss the approach you took to modifying the counter logic?* (5 points)

_________ Testing discussion: *What was your approach to formulating your test benches? How did you verify the correctness of the modules you designed?* (5 points)

_________ Supporting figures: *Waveforms showing correct operation of the top-level module.* (10 points)

_________ Supporting files: *Do the modules pass any tests applied by the grading staff? Modules that do not conform to the requirements of the project specification will receive no credit.* (10 points)

_________ Validation of the final design on the DE1-SOC board (25 points)

_________ **Project Grade**

ECE 3544: Digital Design I
Project 3B Validation Sheet

Student Name: **Jonathan Lemarroy**

Last 4 Digits of Student ID#: **4732**

GTA Validation Instructions:
Program the FPGA on the DE1-SoC board. When the programming has successfully completed, press and release KEY1 to reset the design. Record the value of the seven-segment displays:

Press KEY1

Set SW[6] to 0. (The switches are 0 when they are down.) Press and release KEY0. Record the values of the seven-segment displays as four hexadecimal digits.

SW[6] = 0	
	Press KEY0

If the value does not match the last four digits of the student's Student ID Number, stop the validation.

DO NOT RESET THE DESIGN. Set SW[6:4] to 100. Choose a non-zero value for SW[3:0]. Record the value.

SW[3:0] value

Press and release KEY0 five times. After each release, record the value of the seven-segment displays.

SW[6:4] = 100					
	1st Press	2nd Press	3rd Press	4th Press	5th Press

Reset the design using KEY1. Set SW[6:4] to 101. Choose a new non-zero value for SW[3:0]. Record the value.

SW[3:0] value

Press and release KEY0 five times. After each release, record the value of the seven-segment displays.

SW[6:4] = 101					
	1st Press	2nd Press	3rd Press	4th Press	5th Press

DO NOT RESET THE DESIGN. Set SW[6:4] to 110. Press and release KEY0 five times. After each release, record the value of the seven-segment displays.

SW[6:4] = 110					
	1st Press	2nd Press	3rd Press	4th Press	5th Press

DO NOT RESET THE DESIGN. Set SW[6:4] to 111. Press and release KEY0 five times. After each release, record the value of the seven-segment displays.

SW[6:4] = 111					
	1st Press	2nd Press	3rd Press	4th Press	5th Press

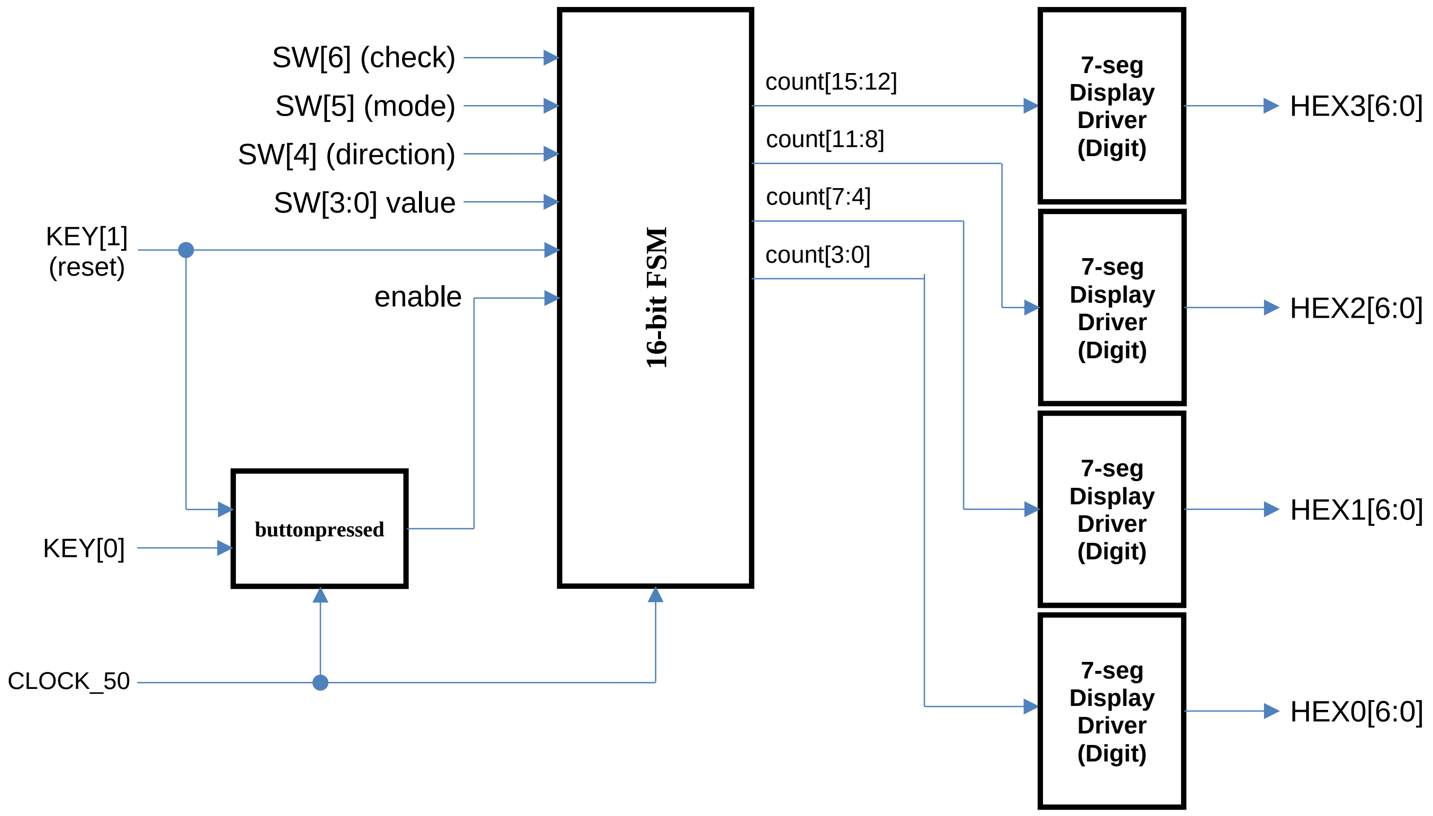
Purpose:

The purpose of this project is to gain experience understanding and designing finite state machines. And to get more practice assigning pins to the correct I/O ports and synthesizing.

Project Description:

The first part of this project is to analyze the current implementation of the "buttonpressed" module provided with this project. This analysis requires a state and timing diagram to be created to represent the module. And secondarily to explain why the reset (KEY[1]) button isn't ran through the "buttonpressed" module as seen in **Figure 1**.

Figure 1: Top-level module block diagram



The second part of this project is to design a 16-bit FSM module that conforms to the requirements below:

```
On a positive clock edge:
    If enable = 1
        Perform the FSM operation determined by the slide switches.
    Else
        Leave the counter state unchanged.
```

- The check (SW6) determines the value of the FSM state:
  - If check = 0, then the 16-bit FSM state should take on the value of the last four digits of your student number, expressed in binary-coded decimal. Your implementation need not calculate these values; you may encode the values as constants.
  - If check = 1, the FSM state should appear as determined by the operations specified below.
- The mode input (SW5) determines whether the FSM acts as a counter or as a shift register:
  - If mode = 0, then the FSM is in "counter" mode.
  - If mode = 1, then the FSM is in "shift register" mode. Implement the shift operations as *circular* shifts.

- The direction input (SW4) specifies the direction of the count and shift operations:
  - If direction = 0, the FSM should increment (count up) if it is in “counter” mode or shift the FSM state to the left by 1 bit if it is in “shift register” mode.
  - If direction = 1, the FSM should increment (count down) if it is in “counter” mode or shift the FSM state to the right by 1 bit if it is “shift register” mode.
- When the FSM is in “counter” mode, the value inputs (SW[3:0]) determine the value by which the FSM state is incremented or decremented. The value inputs have no function when the FSM is in “shift register” mode.

And the last part is to connect the ‘count’ output of the FSM to 7-segment display driver modules that have been created from previous assignments.

Design:

The designing part of this assignment only required using nested if-else non-blocking assignments that followed the FSM spec listed above.

```
module fsm16bit(clock, reset, enable, check, mode, direction, value, count);
    input         clock;
    input         reset;
    input         enable;
    input         check;
    input         mode;
    input         direction;
    input   [3:0] value;
    output [15:0] count;
    reg    [15:0] counter_state;
     always @(posedge clock or negedge reset) begin
        if(reset == 1'b0)
            counter_state <= 16'b0;
        else if(enable) begin
            if(check) begin
                if(mode) begin
                    if(direction)
                        counter_state <= {counter_state[0], counter_state[15:1]};
                    else
                        counter_state <= {counter_state[14:0], counter_state[15]};
                end
                else begin
                    if(direction)
                        counter_state <= counter_state - {12'b0, value[3:0]};
                    else
                        counter_state <= counter_state + {12'b0, value[3:0]};
                end
            end
            else
                counter_state <= 16'h4732;
        end
        else
            counter_state <= counter_state;
    end
    assign count = counter_state;
```

The remaining Verilog work for this assignment was wiring the 7-segment displays as described in **Figure 1**.

Analysis & Conclusion:

The “buttonpressed” module is constructed using a sensitivity list that only updates its current state on the positive edge of a clock cycle or negative edge of reset. Its next states are set the moment the button is pressed, or the current state is changed. See **Figure 2** and **Figure 3** for more information about the module. And based on the “buttonpressed” module’s FSM, the next state is the only thing that should be set immediately for it to maintain is synchronous behavior.

Figure 2: “buttonpressed” module state diagram

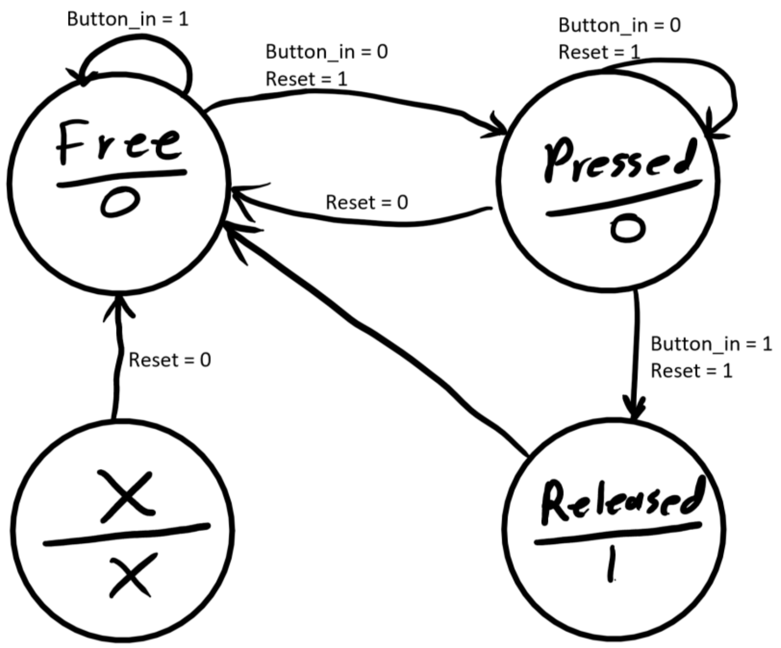
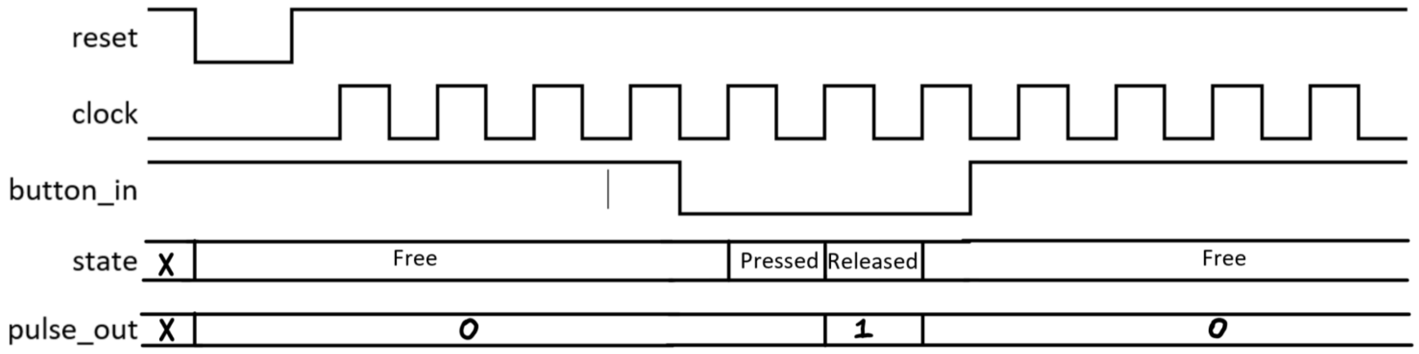


Figure 3: “buttonpressed” module time diagram



I think the primary reason the reset (KEY[1]) isn’t wired through a “buttonpressed” module is likely due to the fact that it couldn’t be asynchronous then.

I tested the top-level module on the DE1-SoC board and found it to function according to the spec requirements. This project has more adequately prepared me for writing and interpreting FSM Verilog modules.

Unfortunately, time ran out for this project while in the process of producing a test bench for the top level module.