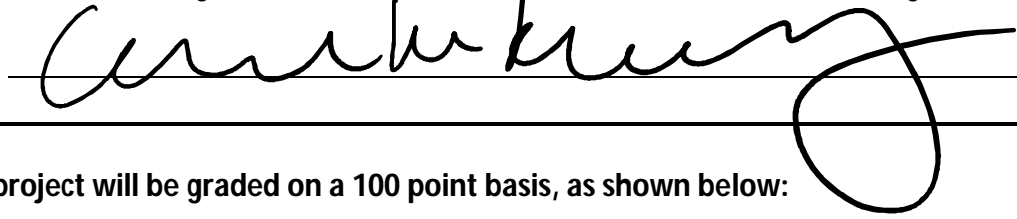


## ECE 3544: Digital Design I

### Project 3 (Part B): Design and Synthesis of a Synchronous Finite State Machine

Student Name: Charlie Kelley

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



---

**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: *Does your state diagram for the keypressed 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**

# Project 3B

## Objective

The aim of this project is to explore the aspects of designing a synchronous finite state machine, an FSM, and how to simulate this design. A module was provided with the outline for the FSM, and our main task is comprehending and modifying the design to specification.

## State Machines

The main objective in this project is to understand the implementation of an FSM, and the best tool to understand an FSM is its state diagram. These diagrams show all states that can be reached by our machine and the flow between states. Figure 1 shows the state diagram for the buttonpressed module that was provided in the project specifications.

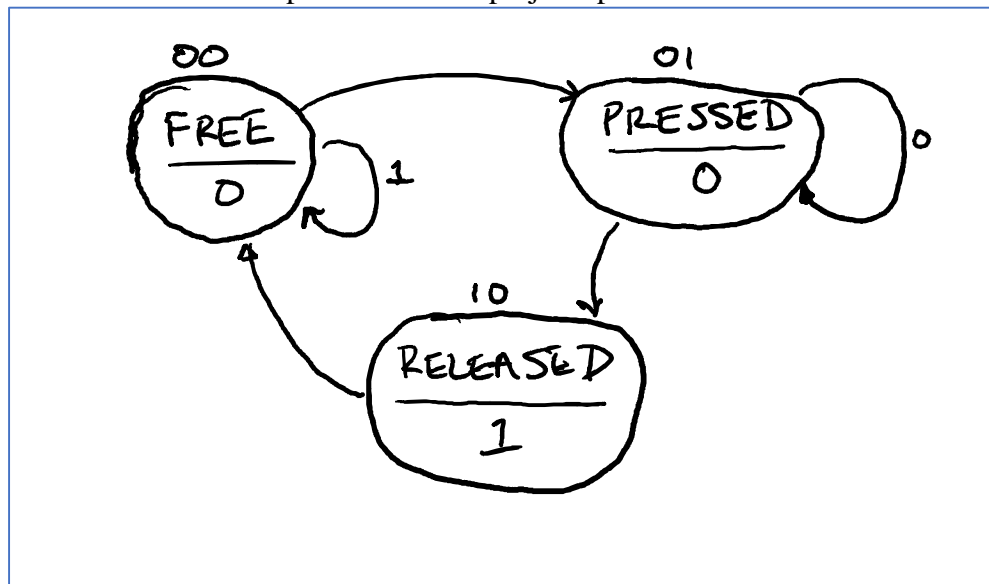


Figure 1. Buttonpressed State Diagram

From observation, each procedural block within the two state machines represents either a set of combinational logic components or a set of sequential logic components. The buttonpressed module uses a procedural block for its handling of states, using sequential logic, and its handling of the outputs, as combinational logic. The FSM for the counter holds similar logic that the states are handled in a block containing sequential logic.

The two state machines differ in a couple aspects. The buttonpressed module uses two procedural blocks. The two blocks handle the register logic and then the output logic separately. On the other hand, the counter uses one procedural block to handle the state of the FSM while it modifies the output directly. The two FSMs differ because of the logic associated with their output. The counter's output is directly related to its previous state, while the buttonpressed FSM has a purely combinational output based on the current state.

In my observation multiple procedural blocks can have several purposes. The main difference observed is sectioning the sequential and combination sections of a state machine. Looking at the components of a Mealy and a Moore state machine, the two modules provided seem to follow these patterns. Within these two macro patterns, the procedural blocks fall into the sections of the two design templates of a state machine.

## Design

The following will discuss my implementation of the modified counter state machine. The counter FSM had three states – reset, enabled, disabled. For a given clock cycle the counter module was always within one of these three states. The modifications for the state machine came within the enabled state of the counter. The counter had to account for four inputs that altered the functionality of the state machine when enabled. Each input combination and its output were independent of the previous input states, so therefore a combinational approach was used within the enable state to determine its output. A simple if-else if-else structure could be used to determine the function of the counter for a given enable state. From this point it was a matter of taking the previous output and modifying its value based on the branch of the enable cycle that was determined by the inputs.

## Testing

The testbench and the validation sheet provided in the project specification provided a template for testing the two FSMs that were used in this project. In order to test a synthesizable mode a clock was simulated using an always block instead of a module with a specify block. With a running clock, the test began by cycling through an array of inputs similar to the validation sheet but expanding on each case and adding an additional case. Within the count up and down feature, the value to increment by was modified through. The last test case that was included tested the counters ability to increment past the max value that could be stored. The waveforms below showcase the inputs and outputs asserted by the two state machines.

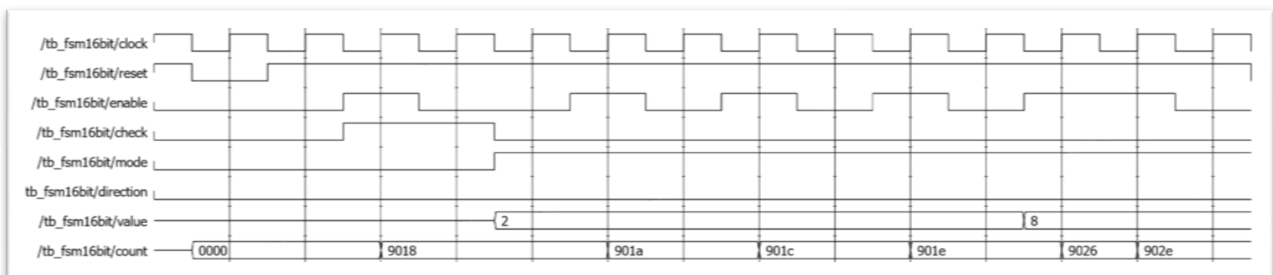


Figure 2. Testing PID display and count feature

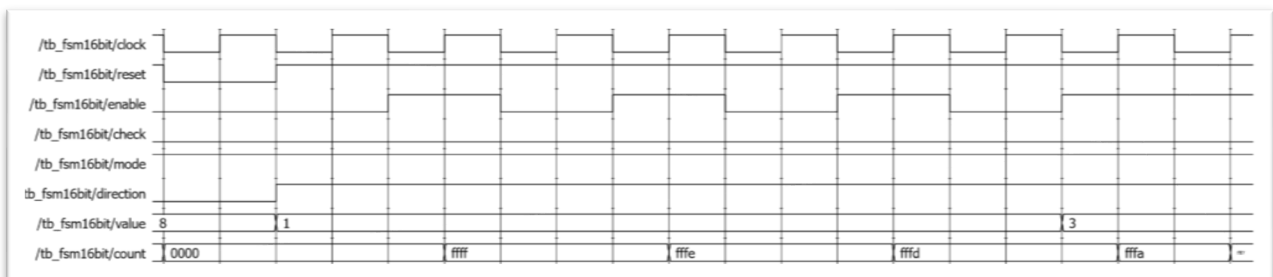


Figure 3. Testing count down feature by different values

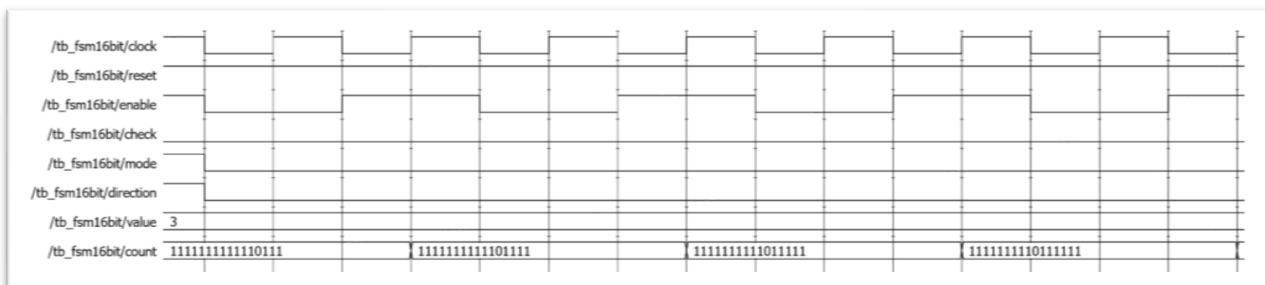


Figure 4. Testing shift left register feature

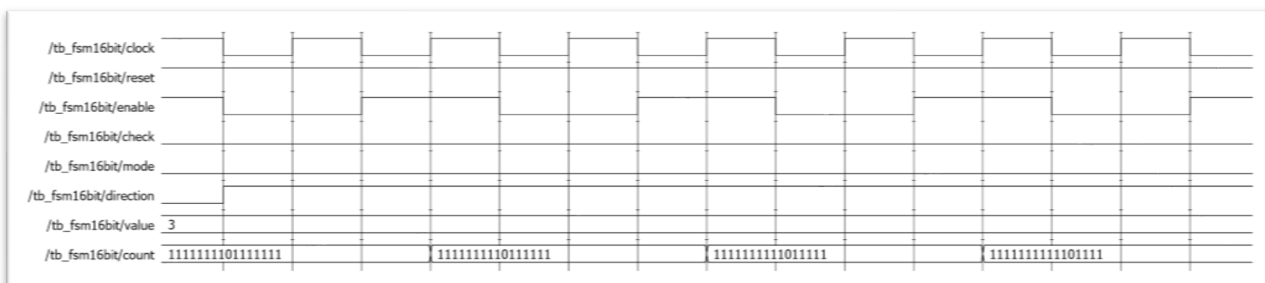


Figure 5. Testing shift right register feature

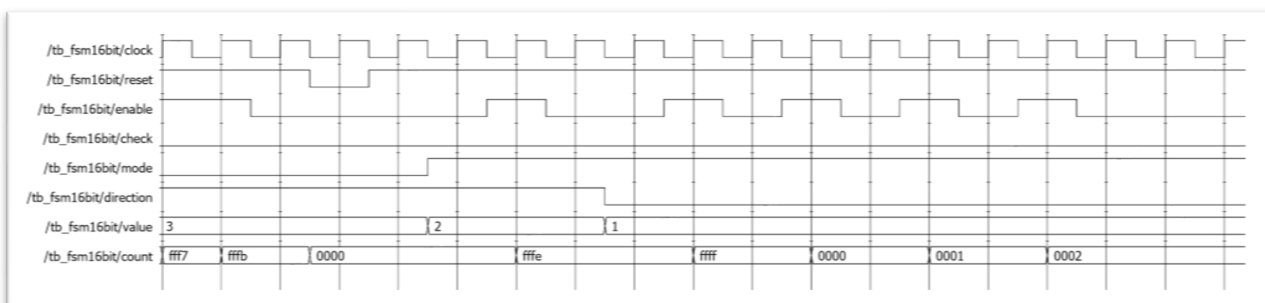


Figure 6. Testing under and overflow cases

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 1. (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] = 1**

\_\_\_\_\_  
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 010. 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] = 010**

\_\_\_\_\_  
1<sup>st</sup> Press

\_\_\_\_\_  
2<sup>nd</sup> Press

\_\_\_\_\_  
3<sup>rd</sup> Press

\_\_\_\_\_  
4<sup>th</sup> Press

\_\_\_\_\_  
5<sup>th</sup> Press

*Reset the design using KEY1.* Set SW[6:4] to 011. 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] = 011**

\_\_\_\_\_  
1<sup>st</sup> Press

\_\_\_\_\_  
2<sup>nd</sup> Press

\_\_\_\_\_  
3<sup>rd</sup> Press

\_\_\_\_\_  
4<sup>th</sup> Press

\_\_\_\_\_  
5<sup>th</sup> Press

*DO NOT RESET THE DESIGN.* Set SW[6:4] to 000. Press and release KEY0 five times. After each release, record the value of the seven-segment displays.

**SW[6:4] = 000**

\_\_\_\_\_  
1<sup>st</sup> Press

\_\_\_\_\_  
2<sup>nd</sup> Press

\_\_\_\_\_  
3<sup>rd</sup> Press

\_\_\_\_\_  
4<sup>th</sup> Press

\_\_\_\_\_  
5<sup>th</sup> Press

*DO NOT RESET THE DESIGN.* Set SW[6:4] to 001. Press and release KEY0 five times. After each release, record the value of the seven-segment displays.

**SW[6:4] = 001**

\_\_\_\_\_  
1<sup>st</sup> Press

\_\_\_\_\_  
2<sup>nd</sup> Press

\_\_\_\_\_  
3<sup>rd</sup> Press

\_\_\_\_\_  
4<sup>th</sup> Press

\_\_\_\_\_  
5<sup>th</sup> Press