Project A

Thunderbird Taillight Finite State Machine

TCES 330 Digital

Spring 2020

Authors:

Allysen Arntsen Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings
Ink Drawings


Riley Ruckman 

Travis Shields *Travis Shields*

Submitted 5/22/2020

# 

# **Table of Contents**

[**Table of Contents**](#_lgaqp4cn22nv) **2**

[**Project A Logistics**](#_37f2nguz9z7t) **3**

[**Design**](#_2d6oego5kd5h) **3**

[Design Requirements](#_lxxob5g9o91f) 3

[Design Process](#_aswbtia0lix4) 4

[**Test Procedures**](#_62vdmf7umhgi) **6**

[SystemVerilog Testing](#_2ak3h0kjhbr9) 6

[DEII Board Testing](#_xmi4g6gxo6us) 7

[**Test Results**](#_isky9r4xk2v1) **8**

[**Conclusion**](#_j5v55qslog0r) **9**

[**Appendix**](#_75qmgi8xy5bj) **11**

[Link to video of DEII board test](#_fvcpzc7mdfo5) 11

# Project A Logistics

The purpose of this project is to create a Finite State Machine that models the iconic Thunderbird taillight system.

Our team members are Allysen Arntsen, Riley Ruckman, and Travis Shields

We divided the workload in the following way:

* Allysen determined the state logic and output logic
* Riley programmed the Taillight FSM module
* Travis dealt with testing the design

# Design

## Design Requirements

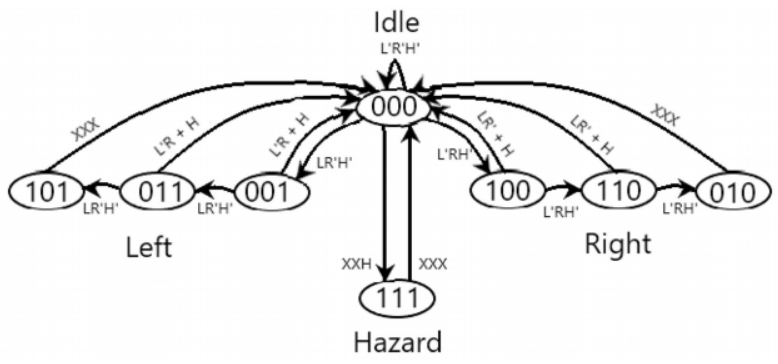
The goal of this project is to build a circuit that models the taillights of an older model Ford Thunderbird. Each side of the car has 3 lights to make up the turn signals and hazard.

Position of tail lights: **LC LB LA RA RB RC**

To control the lights, the driver needs three inputs: right (R), left (L), and hazard (H). When the driver inputs L, the lights should flash in the sequence: LA on; LA and LB on; LA, LB and LC on; all off. Then the sequence should repeat until the driver turns off the signal. The same sequence should happen on the right side when the driver inputs R. The inputs L and R also cannot be on at the same time, which makes sense in reality. If any input is changed in the middle of any sequence, the lights should all switch off and then go to the new sequence based on the current set of inputs. If H is turned on at any time, the lights should all switch off, then every light should blink on and off. All changes should take place on a 1 second interval.

## Design Process

Since the output needs to be continuously updated given a constant input signal, this design clearly needs to utilize a finite state machine; more specifically, a Moore machine. From the logic defined under the “Design Requirements” section, we can determine what states we need: three for each turn input, an idle state with all lights off, and a hazard state with all lights flashing. A state transition diagram depicting this FSM is shown in Figure 1.

Figure 1. State Machine Diagram.

To produce this circuit in SystemVerilog, we used a case statement, with each case corresponding to one of the eight states. Each case must account for all input possibilities. The output is determined based on the current state of the finite state machine, and matches up to what is shown in Figure 1. The most important segment of this finite state machine can be seen in Figure 2 below.

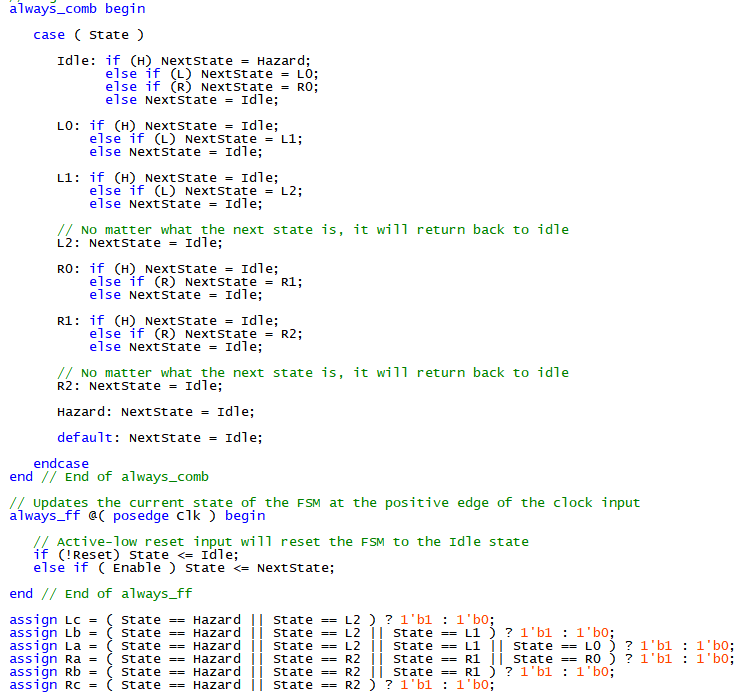
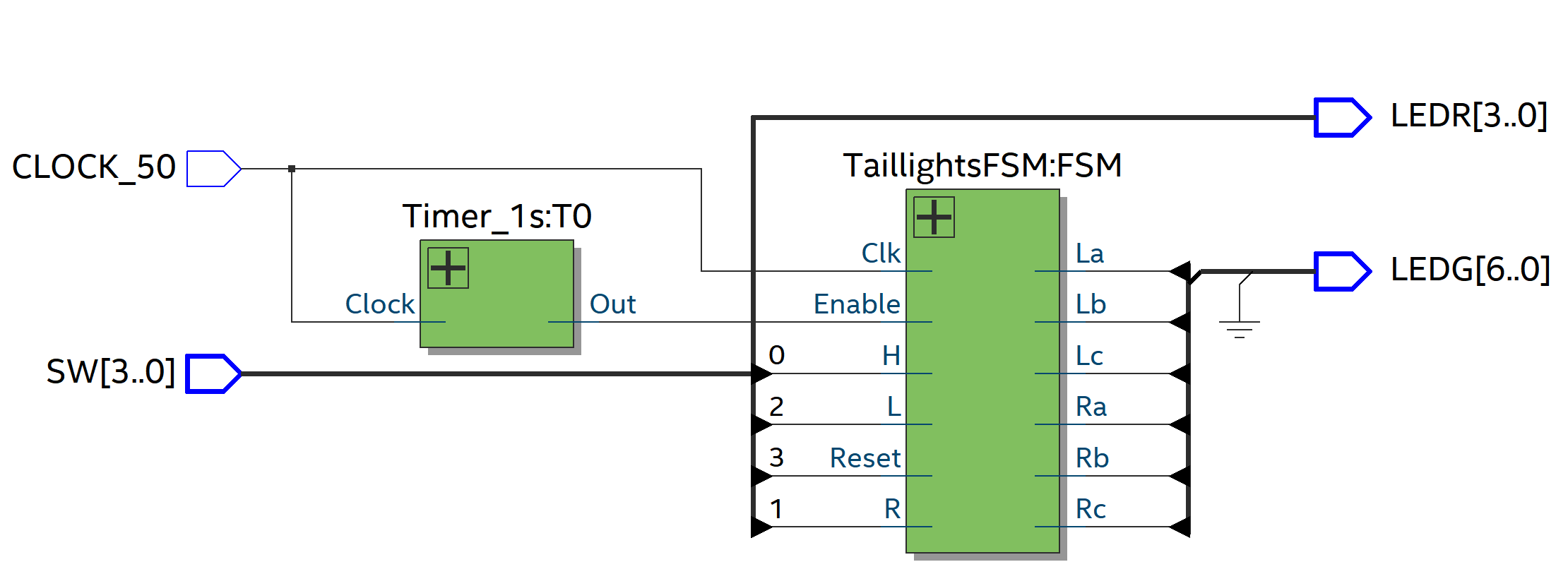


Figure 2. Finite State Machine Instructions.

To get the FSM to update every one second, we will be repurposing a module we created for Homework 3. The module *Timer\_1s* counts up to 50,000,000 and then outputs a positive signal. Using an onboard 50MHz clock input, this module will output a 1 Hz clock signal. The FSM is used with the 50 Mhz clock as its clock input and the 1 Hz clock as its enable input.

Figure 3. RTL Viewer diagram of full circuit.

# Test Procedures

## SystemVerilog Testing

The first testing method we used for this circuit was a ModelSim simulation using a SystemVerilog test bench (Figure 3). To implement this, we created a clock signal with a period of 20 ps. Our simulation would run for 750 ps and test cases for no signal, left turn, right turn, and hazard override for both signals. To ensure hazard would function in any case, we first tested its ability to function when not having to override any signals. We then proceed to test the right signal, then hazard override, then left signal, and hazard override again. Each of these signals would run long enough for about two cycles to complete so we could be certain if they were functioning properly. Results were to be displayed in both a table and a waveform model in the ModelSim software.

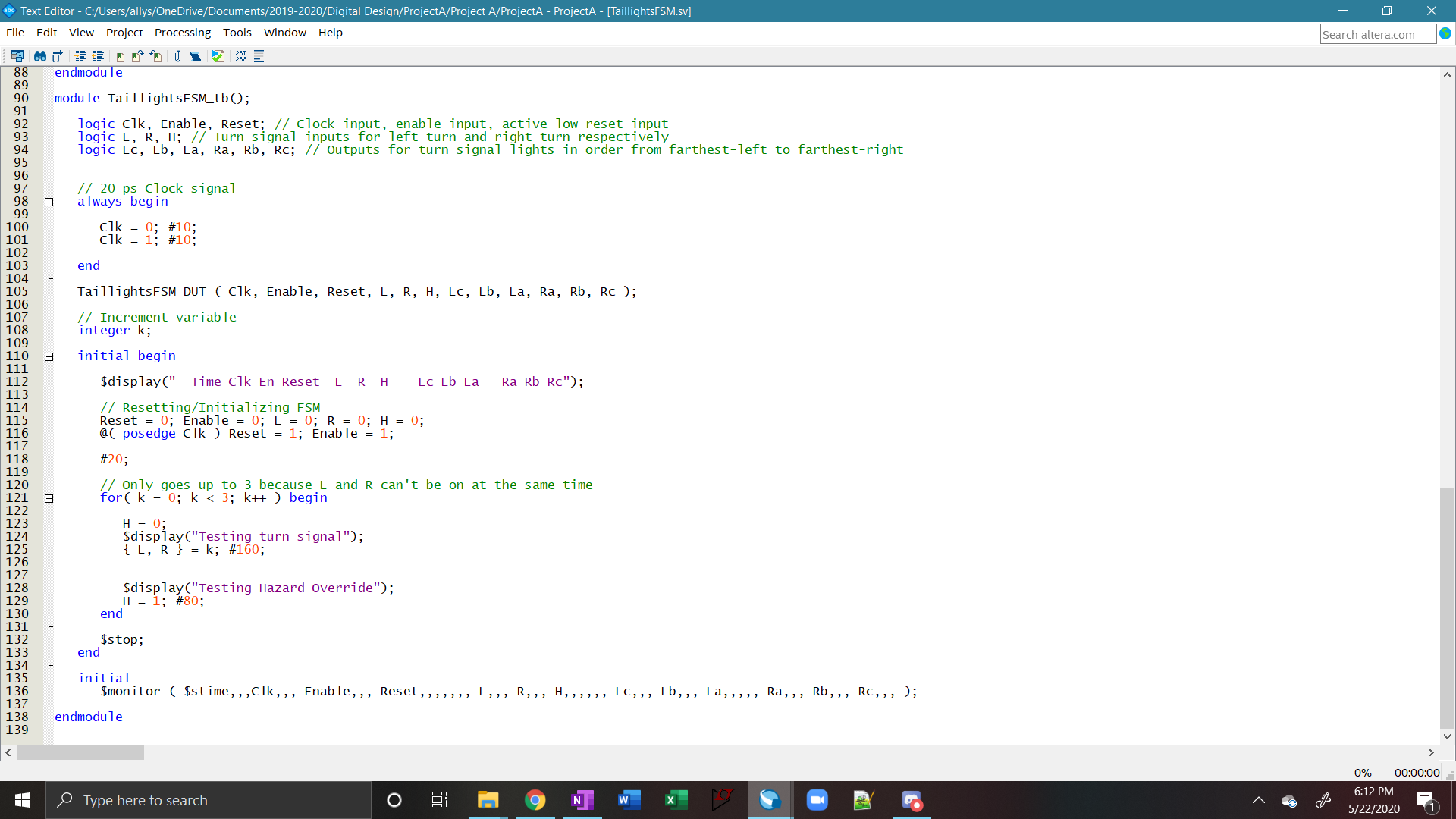


Figure 4. SystemVerilog Code for FSM test bench.

## DEII Board Testing

To implement this circuit on our DEII board, we wrote the top-level module ProjectA as seen in Figure 4. This utilized a 50MHz clock which we tied in to our *Timer\_1s* module to implement changes once per second. We used SW3 as our Reset bit for the FSM. To test this, we observed that the circuit would not function unless this switch was enabled. With SW3 at a high value, we could then test the other functions of our circuit.

SW2 was used to control the left turn function. This switch would make LEDGs 3-5 operate in sequence as the left turn signal. Similarly, SW1 was wired to operate the right turn function which utilized LEDGs 0-2. SW0 was wired to the hazard signal for the circuit. This was tested and observed on the board to override both the left and right signals and ultimately result in 1 second flashing of green LEDs 0-5 simultaneously. We took a video recording to demonstrate the functionality of our working circuit (Appendix A).

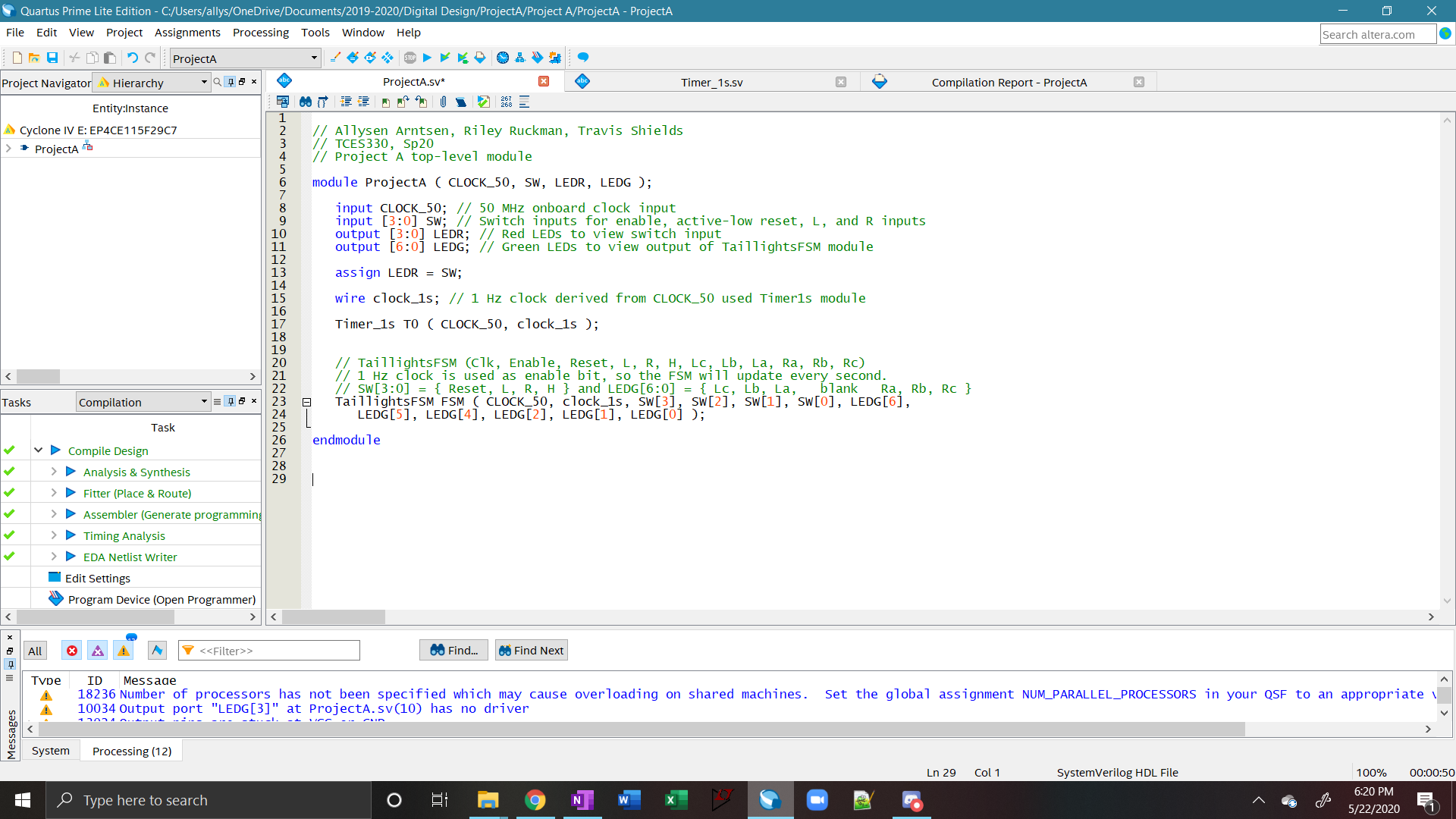
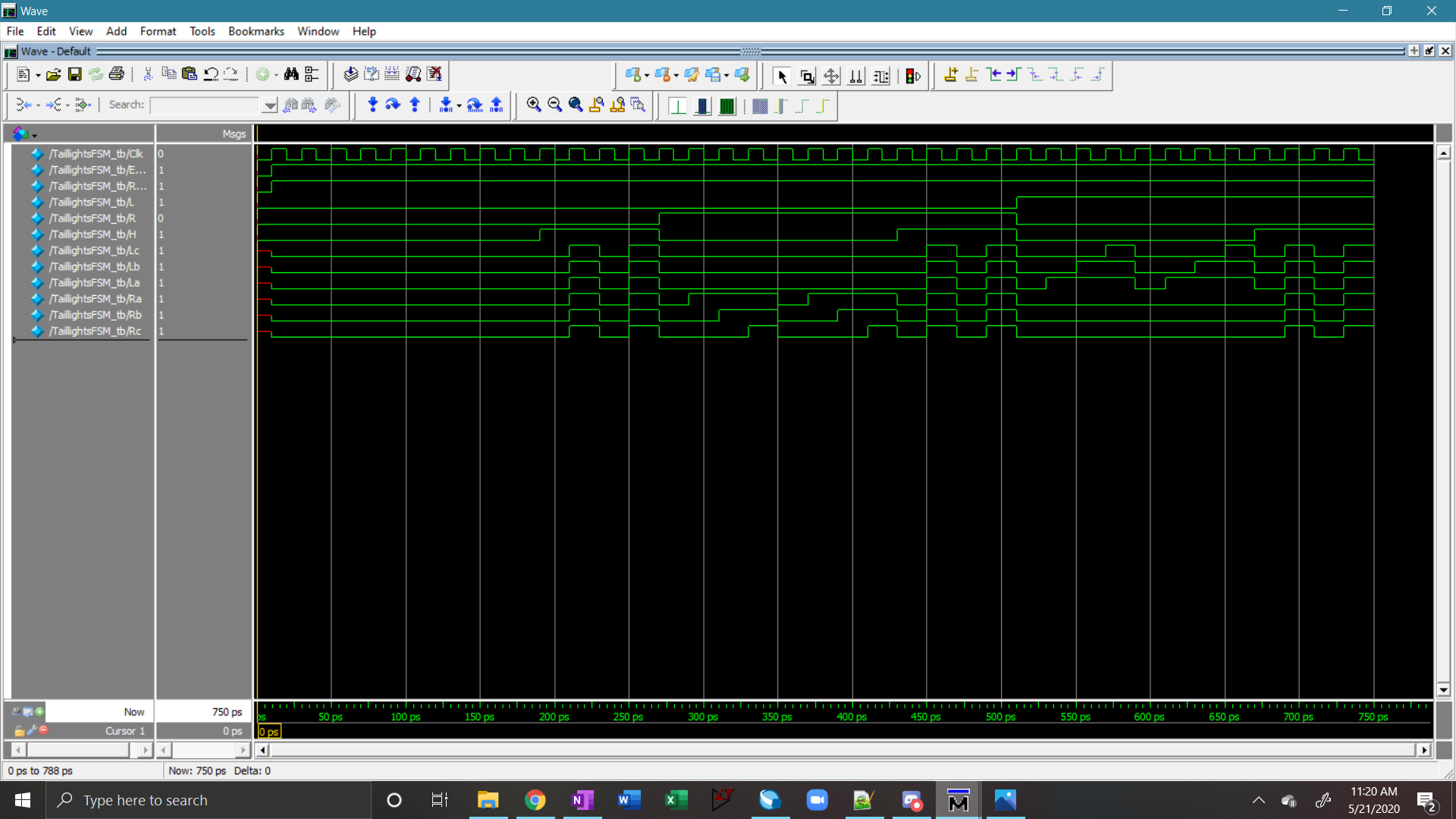


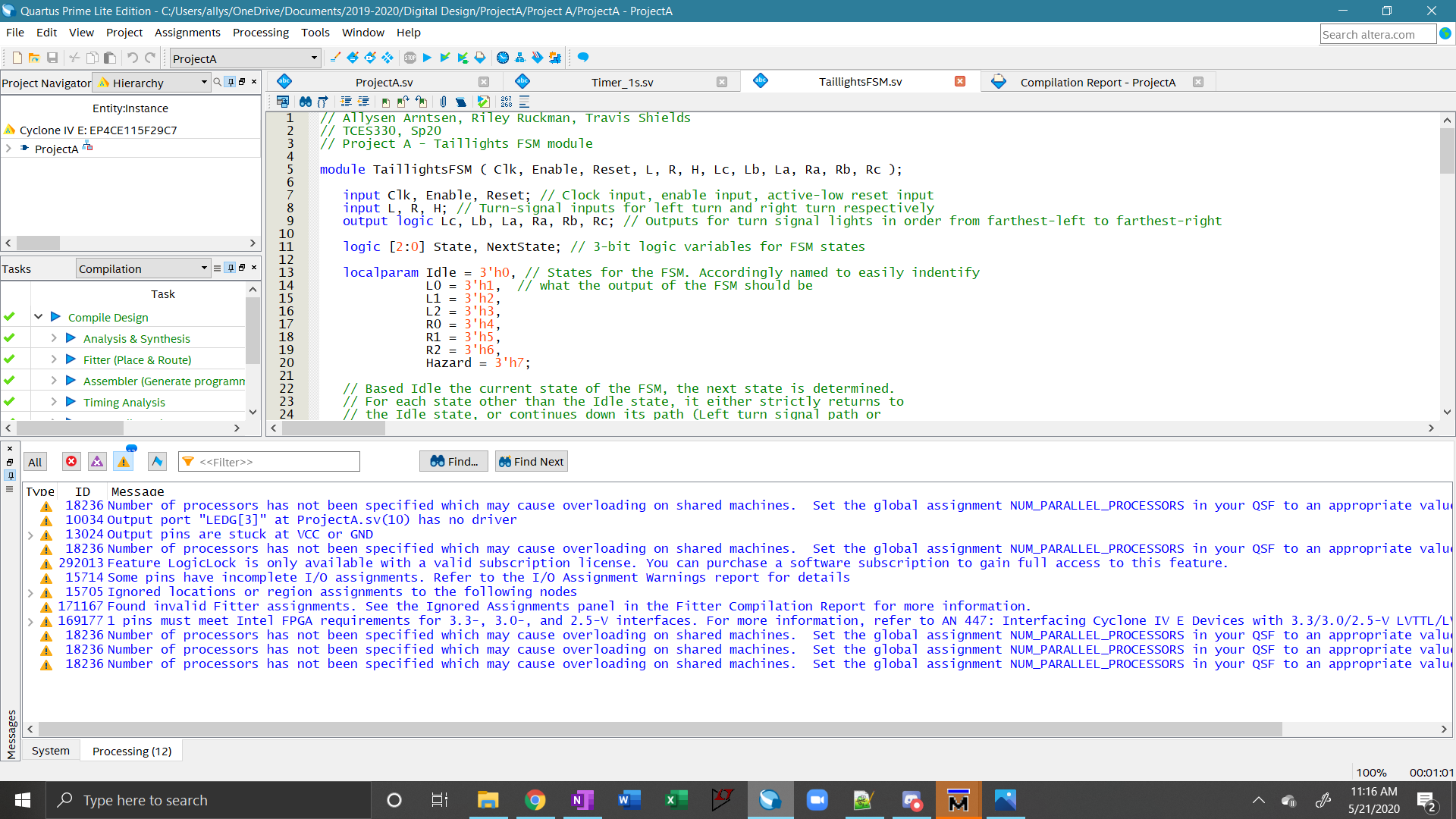
Figure 5. SystemVerilog code for DEII Board testing.

# Test Results

Through the testing process using a ModelSim simulation and by performing a test on our DEII board, we found that our circuit did indeed function as we intended it to. The output of the ModelSim simulated wave, Figure 6, showed the proper functioning of all cases that we tested. The output LEDs remained off while no input was activated, flashed simultaneously when the hazard indicator had a high value, and they responded as expected to create the staggered lighting of the left and right turn signals. On the DEII board, all input switches and output LEDs behaved as intended. The FSM updated every second, and followed the design requirements specified previously.

Figure 6. ModelSim Test Results.

As seen in Figure 7, Quartus warnings were properly handled to ensure our circuit test was accurate. There were no timing errors or crucial runtime errors of any kind that would interfere with the operation of our circuit.

****Figure 7. Final Quartus Warnings.

# Conclusion

Through the designing, testing, and implementation of this circuit design using a finite state machine model, we were able to successfully complete the task of using SystemVerilog code to design a circuit to mimic the patterns of the old Ford Thunderbird tail lights. This design proved to be far simpler than the version we had previously built in TCES 230 which was implemented through combinational logic. However, by creating this same result using two different methods, we can see the importance of utilizing proper and efficient means of accomplishing a task in SystemVerilog. We were also able to see multiple ways of implementing the same finite state machine in Verilog code.

This project also allowed us to have the unique opportunity to test our design though multiple methods. Not only could we run a ModelSim simulation of the circuit we had designed, but we also were able to run a physical test on our DEII board. This simplified the process immensely from what we had to do in TCES 230 where we had to manually wire everything together on a breadboard to build the circuit. Because we already have this knowledge, this task was simplified so that the Quartus software would handle the physical building of the circuit. This streamlined our testing process and made it much easier to come to a conclusion that everything worked as we intended it to.

In part because we had previously designed a version of this circuit, our group was not met with many major challenges. We instead found that this circuit was fairly similar to other state machines that we had designed in past homework assignments. This experience combined with our prior knowledge of the general design and functionality of this circuit made it a simple and smooth process to create a functioning model of the Ford Thunderbird tail light circuit.

# Appendix

## Link to video of DEII board test

<https://drive.google.com/file/d/1BVH0aQPEIoxDEkg_aHXRRZXoLFKFb4Pi/view?usp=sharing>