

Project A

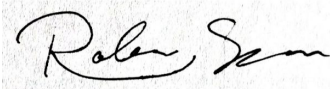
Thunderbird Tail Light

TCES 330 Digital Systems Design Spring 2020

Authors:



TRUNG DO



ROHAN SEAM

Submission Date: May 22, 2020

Table of Contents

Project A

1. Requirements	3
2. Design	4
3. Test Procedures	5
4. Test Results	6
5. Observations	10
6. Conclusions	10

PROJECT A

The purpose of the project is to learn how to write System Verilog code and test circuit using Modelsim, through designing and implementing the thunderbird tail light design on the DE2-115 board.

Our team members are: Rohan Seam and Trung Do

Work load:

- Quartus Taillights.sv module: Rohan (FSM), Trung (testbench)
- Quartus Top-level module: Trung
- Quartus Timer module: Rohan
- DE2-115 Board testing: Rohan
- Project report: Rohan (requirements, design, test procedures), Trung (test results, observations, conclusions)

The work load divided equally among both members.

1. Requirements

We are required to design a finite state machine using System verilog that will control the lighting sequence of the tail light that is made of 6 leds, a set of three on the left and three on the right. The lighting sequences indicate left turn, right turn and emergency flash (blink) based on the input signals Left (L), Right (R) and Hazard (H).

Position of tail lights: Lc Lb La Ra Rb Rc

When the input L is on, the flashing sequence must be La on, then La and Lb on, then La, Lb, and Lc on, then all lights off again, and then the sequence repeats. The same operation works for input R. L and R can not be on at the same time, for this case, all lights go off. During a left or right turn flashing sequence, if L switches to R or vice versa, all lights will go off and then start the new sequence. When the input H is on, all six lights flash on and off together. The input H takes precedence over L and R, so if H is on when L or R is also on, all lights go off and then flash on and off together. Finally the light must flash at a rate of 1 second.

2. Design

The next state will base on the current state and 3-bit input L (Left), R (Right), and H (Hazard). L is the MSB and H is the LSB respectively. When the input is X, it means that no matter what the inputs are, the next state is as indicated. Z is the output of the state. The 6-bit output represents 6 leds of the tail light Lc, Lb, La, Ra, Rb, Rc respectively as Lc is the MSB and Rc is the LSB.

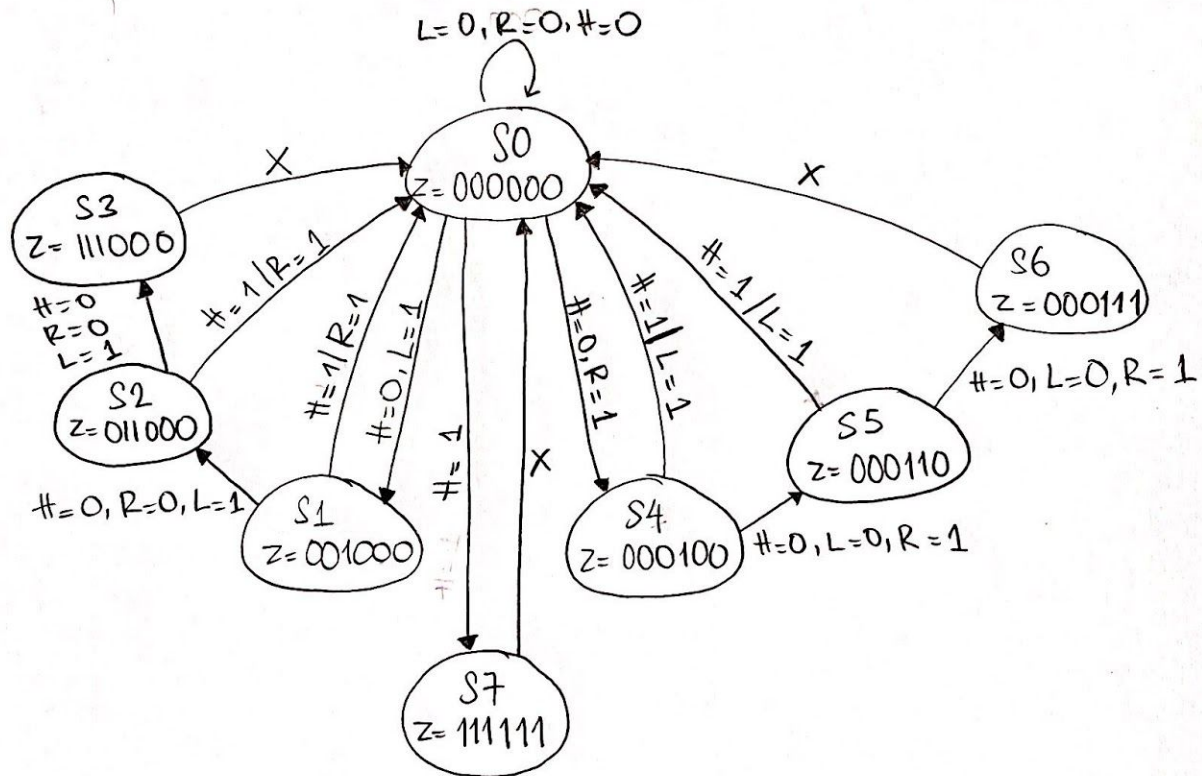


Figure 1. Thunderbird tail light design state diagram

A truth table was also created to help visualize the state transitions based on the current state and the input signals. The table also fills in the input combinations that did not appear on the state diagrams. We named each state with a 3-bit binary number. The X bit of the input means that it can be either 0 or 1.

Tabel 1. The state transitions and output table

Current state		Next state					Output
State	State name in binary	000	100	010	XX1	11X	Lc Lb, La, Ra, Rb, Rc
S0	000	S0	S1	S4	S7	S0	000000
S1	001	S0	S2	S0	S0	S0	001000
S2	010	S0	S3	S0	S0	S0	011000
S3	011	S0	S0	S0	S0	S0	111000
S4	100	S0	S0	S5	S0	S0	000100
S5	101	S0	S0	S6	S0	S0	000110
S6	110	S0	S0	S0	S0	S0	000111
s7	111	S0	S0	S0	S0	S0	111111

3. Test Procedures

In our design, our finite state machine operated off of a clock input, an enable bit, a left turn signal, a right turn signal, and a hazard signal. The output represents a 6-bit vector of binary values that specifies which lights would be on for both of our tail lights. We used the “case” statements in an “always” block to set the states following the state diagram we sketched in 1. An “always_ff” block is used to implement the flip-flop that propagates the state from one state to the next at the positive edge of the clock cycle.

The Timer module was implemented to take the 50 MHz clock supplied by the DE2-112 board and output a 1Hz clock signal to propagate our state machine once per second. Both modules were tested extensively and the timing diagrams of the finite state machine is illustrated in Figure 5, 6, and 7.

For the tail lights FSM testbench, testing all cases and comparing results from ModelSim and the DE2-115 board was important.

Finally we created a top level module that instantiated the thunderbird tail light FSM module and the Timer module. The bits [5:3] represent the left tail light, while the bits [2:0] represent the right tail light. The right turn input is controlled by SW[1], left turn by SW[2], and hazard by SW[0]. The output LEDRs were used to display the state of the switches. They were connected together as specified by Figure 3.

Table 2. Modelsim testing results expectation table

L	R	H	Expectation result
0	0	0	All lights off
0	0	1	All lights off then Emergency flash
0	1	0	Right turn
0	1	1	All lights off then Emergency flash
1	0	0	Left Turn
1	0	1	All lights off then Emergency flash
1	1	0	All lights off since not allowed
1	1	1	All lights off since not allowed

4. Test Results

When we completed our design, we were able to test its operation in ModelSim through the testbench module we created.

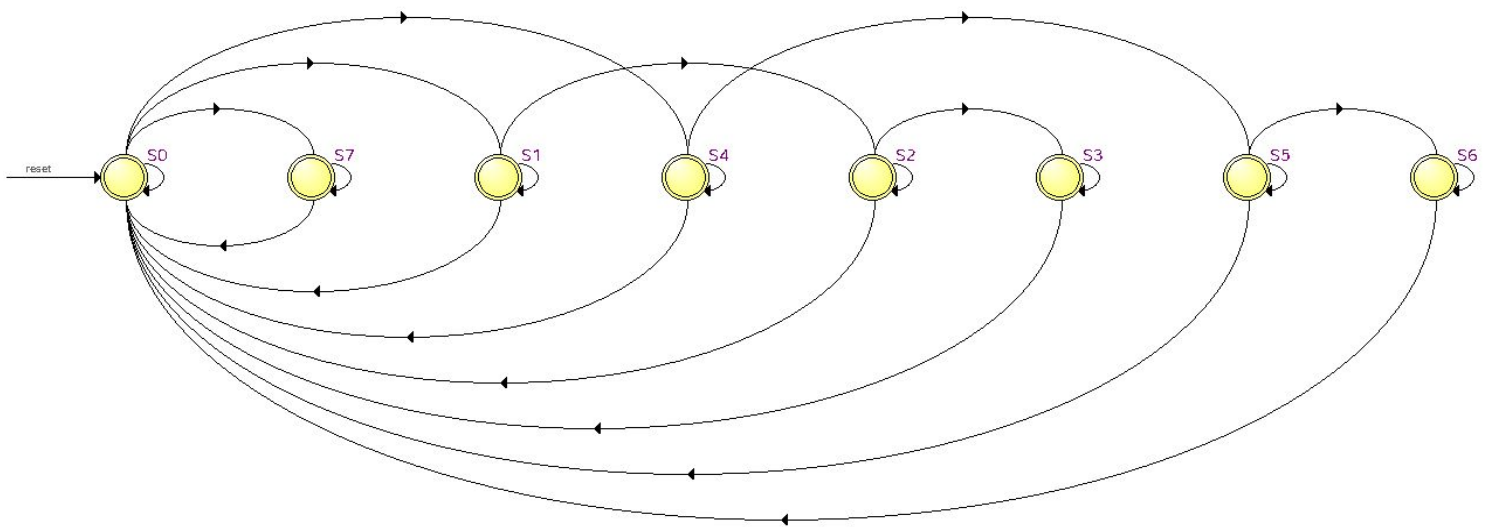
**Figure 2.** FSM viewer under Quartus

Figure 2 displays all the states of the finite state machine, as well as the transitions from each state to the next. S0 represents the default state where all lights are off, while S7 represents the state where all six of the lights are on. We can confirm that this matches our sketched state transition diagram as shown in Figure 1. The only difference is that

each state can transition to itself as we have implemented an enable bit which will not allow the state machine to propagate while the enable input is zero.

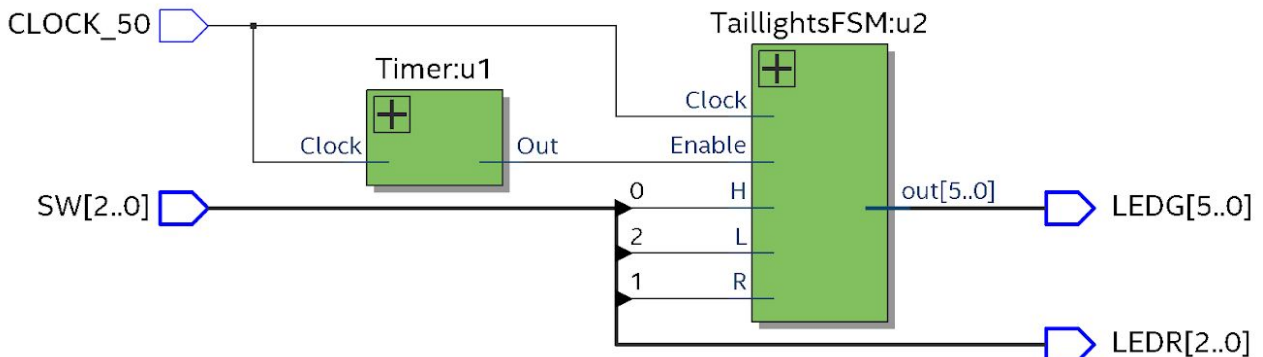


Figure 3. RTL Viewer of the FSM to Illustrate the Project Hierarchy

We can see that both the Timer module as well as our FSM module have synchronized clocks. The output of the Timer module, a 1Hz clock signal, is applied to the enable input of the FSM which controls when the states are propagated. The green LEDs display the state of the tail lights while the red ones display the state of our switch inputs.

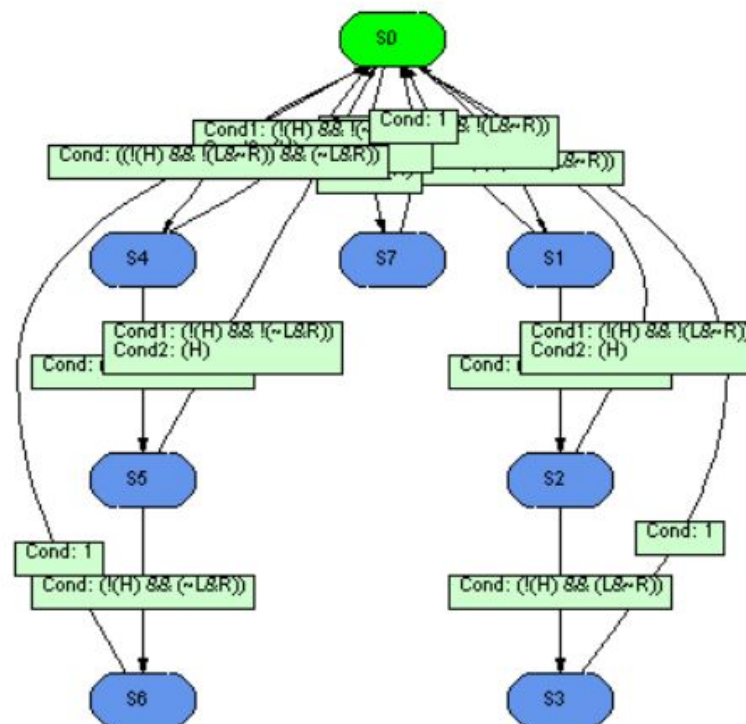


Figure 4. State Transition Diagram (ModelSim)

We can see from Figure 4 that this state transition diagram matches the one that was generated in Quartus (Figure 2). The conditions match what we have implemented in the code which makes sense to what we are seeing.

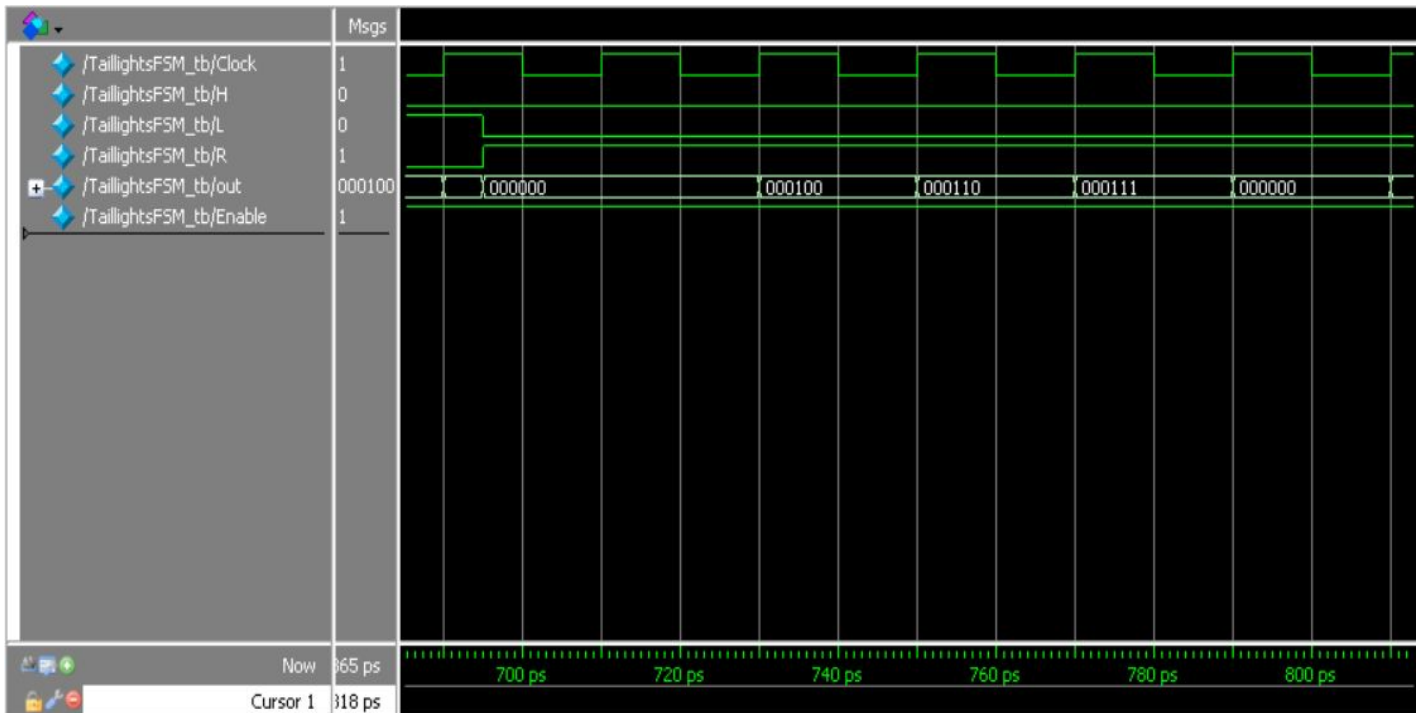


Figure 5. Right turn Timing Diagram (ModelSim)

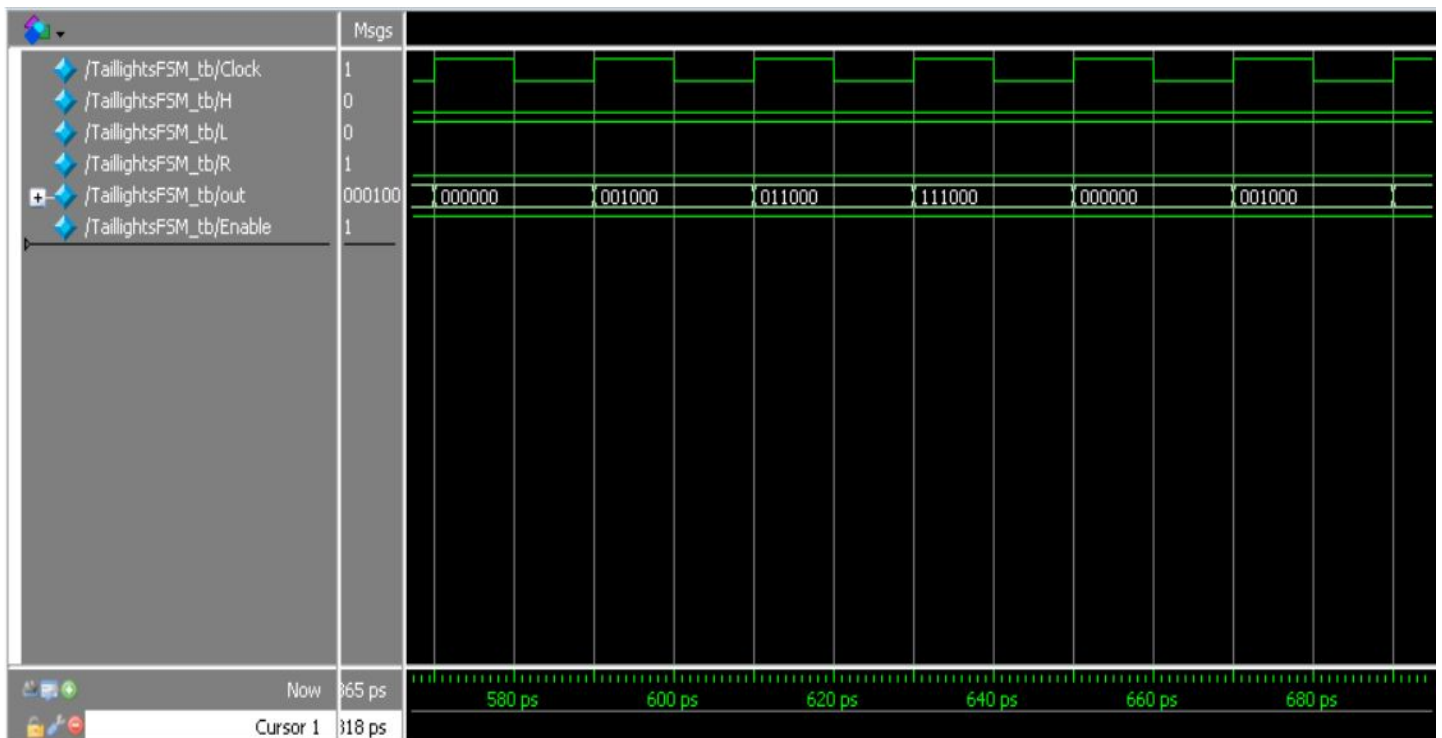


Figure 6. Left Turn Timing Diagram (ModelSim)

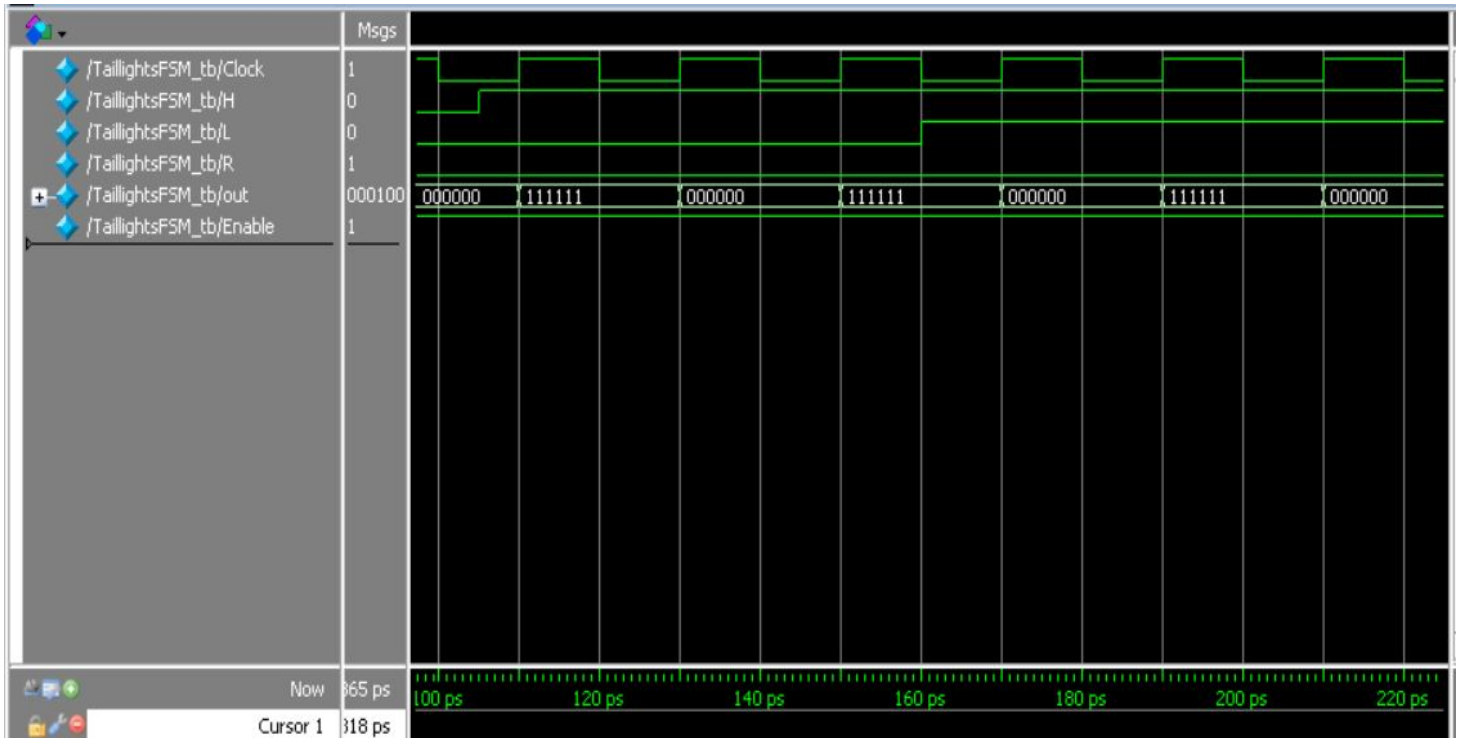


Figure 7. Emergency flash, blinking Timing Diagram (ModelSim)

18236 Number of processors has not been specified which may cause
 18236 Number of processors has not been specified which may cause
 292013 Feature LogicLock is only available with a valid subscriptic
 15714 Some pins have incomplete I/O assignments. Refer to the I/O
 171167 Found invalid Fitter assignments. See the Ignored Assignment
 169177 1 pins must meet Intel FPGA requirements for 3.3-, 3.0-, and
 18236 Number of processors has not been specified which may cause
 18236 Number of processors has not been specified which may cause
 18236 Number of processors has not been specified which may cause

Figure 8. Final warning messages from Quartus

Figure 8 shows all the warning messages from our completed project. We removed all the warnings related to timing as well as other miscellaneous warnings that were easily resolvable. The only remaining warnings are ones that are expected to be present.

The design is downloaded to the DE2-115 board and tested in hardware. All input combinations are tested and the lights display the correct patterns. The following link provides a physical demonstration of our implementation of the thunderbird tail light,

<https://www.youtube.com/watch?v=cBjLhK8NK04>

5. Observations

One issue we had with implementing this circuit was recognizing the importance of priority in our states. For example, we experienced the issue of our design not returning to state S0 (default) state when either of the tail lights were in mid cycle and, for example, the hazard signal was enabled. To resolve this, we checked for a hazard signal before continuing our cycle. Another point to mention was the importance of understanding illegal inputs (L and R on). This led to us implementing an output of all the lights off when both the left turn and right turn signals were on.

6. Conclusions

The project was completed successfully. We were able to design and implement the thunderbird tail light on the DE2 Board. The hardest part of this project is when we were sketching the state diagram, because we have to understand the problem, like what is the sequence, which inputs are directing the state transitions and what is the output at each state and make sure that the state diagram satisfies the requirements. Writing the code is very straightforward since we can just follow the sketched diagram.

Through the project, we learned to cooperate and work together as a team. The importance of communication and team chemistry played an important role in our success to the project completion. We learned the basic idea of how a team project is designed, implemented, and tested while working together to achieve the common goal.