NYIT

COLLEGE OF ENGINEERING AND COMPUTING SCIENCES

Spring 2020 Semester

EENG 483 – M01

Introduction to VLSI Design

Instructor: Professor Pekcan

Traffic Light Controller

Final Project

Jens Daci

ID# 1135580

Due Date: May 1st, 2020

Date of Submission: May 1st, 2020

Table of Contents

I. Objective

- To design a Traffic Light Controller for use in road intersections
- To implement the design by using the IEEE Standard VHDL Language
- To simulate the system using the Quartus II software
- To be able to understand and explain the output results

II. Introduction

As part of the requirements for the Introduction to VLSI class, a final project in VHDL must be completed in order to fulfill the objectives of the course. In this report, a Traffic Light Controller will be described as a top-level view using a block diagram, as a finite state machine using a state diagram and it will also be constructed in detail in Quartus II software using the IEEE Standard VHDL Language. Finite state machines can be used to model situations and problems in many fields and a traffic light controller is one of them.

This Traffic Light Controller will be able to manage four traffic lights in a road intersection as depicted in Figure 1 below. The design will be implemented with different delays between traffic light states in order to mimic a real-life situation. In other words, the RED and GREEN will stay on for longer periods of time when compared to YELLOW.

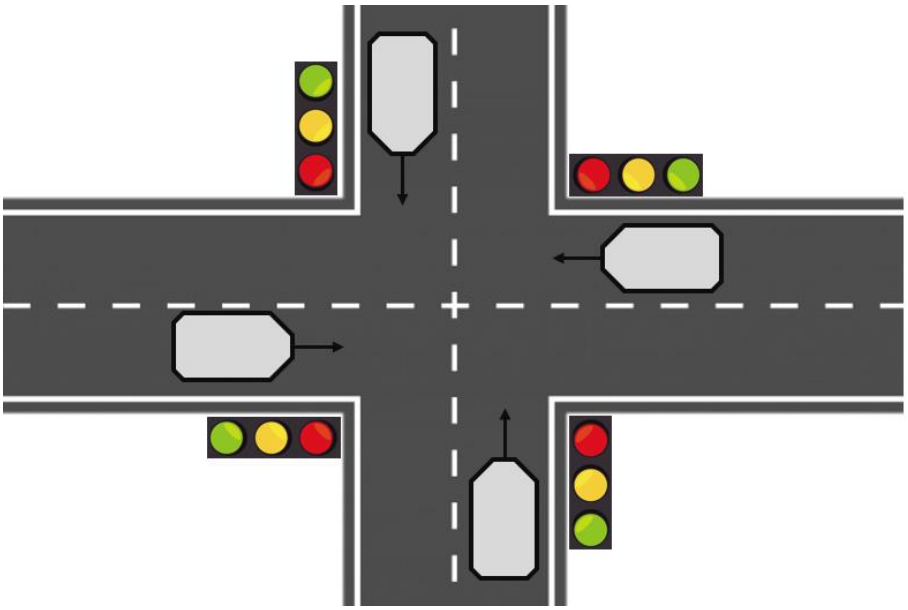


Figure 1. Four Traffic Lights in a Road Intersection

III. State Diagram

Since the Traffic Light Controller will be implemented as a finite state machine, a state diagram must be drawn first before constructing a block diagram and starting the code implementation. A state diagram is used to describe the behavior of a system and must be composed of a finite number of states. In this project, the state diagram for the controller will consist of six states. In contrast to other FSMs, the logic flow in this controller will be based on a counter rather than an input bit. The diagram can be seen in Figure 2 below.

The clock input to the controller will determine the delay in seconds of each state. Depending on the frequency of the clock, the delays will change. In order to calculate the delay, the following formula is used:

$$Delay\ [in\ seconds] = count * clock\ frequency + 1$$

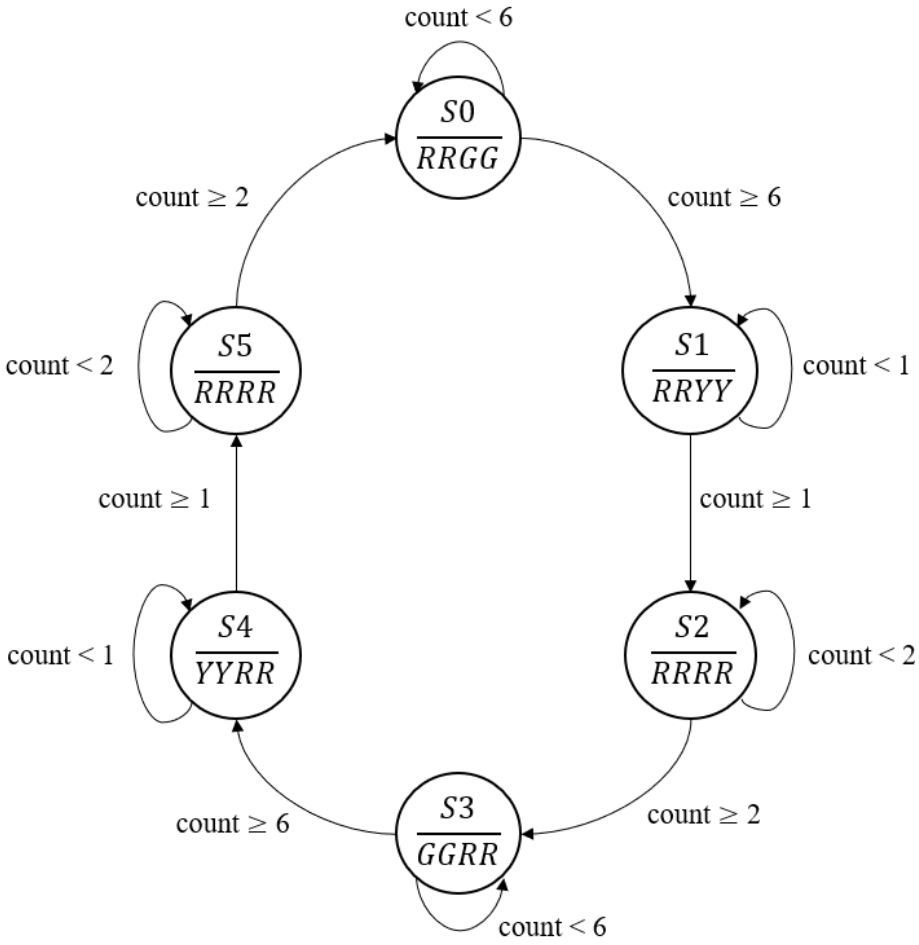
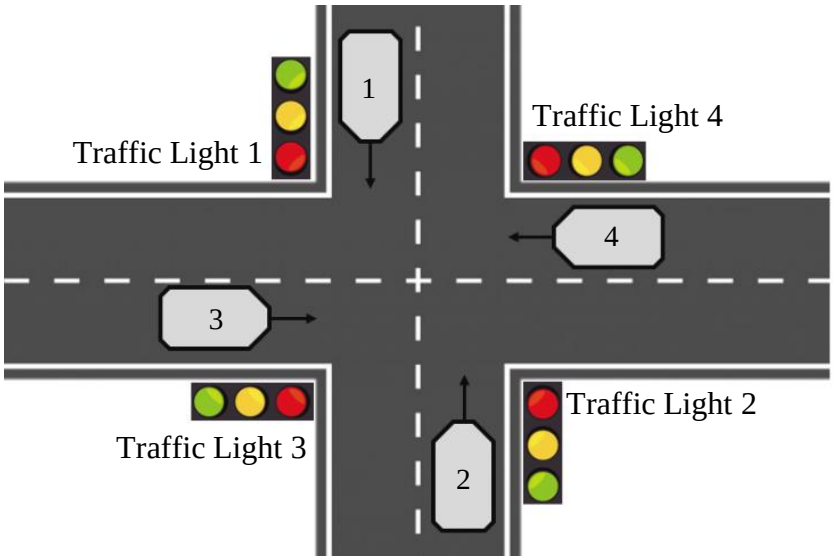


Figure 2. Moore State Diagram of the Traffic Light Controller

All of the six different states shown in the figure above correspond to a situation in the road intersection. For every case, the lights will change their state and manage the traffic efficiently. The cars will stay at RED light for a period of 12 seconds, at YELLOW light for a period of 3 seconds and at GREEN light for a period of 7 seconds. The description of the states can be seen in Table 1 below.



State	Traffic Light 1			Traffic Light 2			Traffic Light 3			Traffic Light 4			Delays
S0	R			R					G			G	7 [s]
S1	R			R				Y			Y		3 [s]
S2	R			R			R			R			2 [s]
S3			G			G	R			R			7 [s]
S4		Y			Y		R			R			3 [s]
S5	R			R			R			R			2 [s]

Table 1. Description of States and their Delays in seconds

IV. Block Diagram

The block diagram of the controller will show the different inputs and outputs of the system in order to better visualize it.

The controller will have three inputs: CLOCK, ENABLE, and RESET
It will also have four outputs: Out_1, Out_2, Out_3, and Out_4

The block diagram of the controller can be seen in Figure 3 down below.

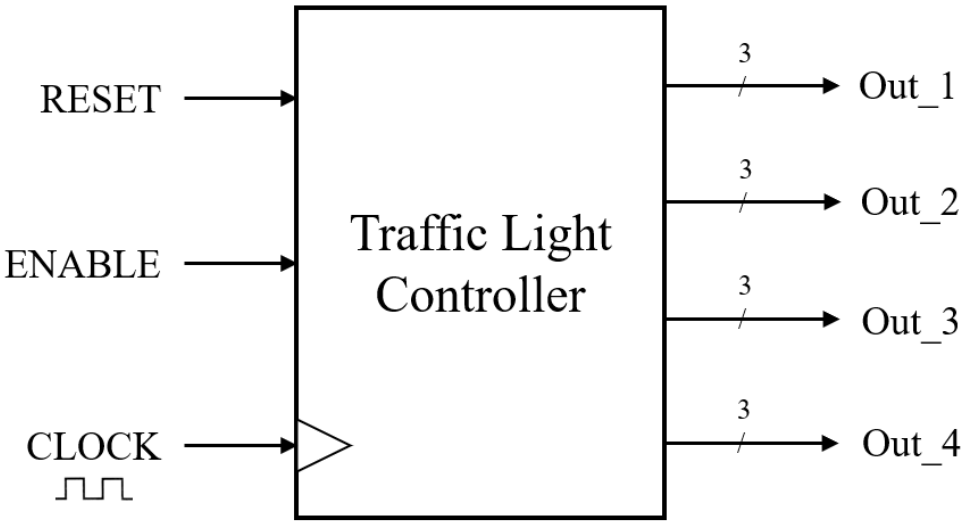


Figure 3. Block Diagram of the Traffic Light Controller

The design will have four different outputs in order to manage the four traffic lights on each road. Each of the outputs is composed of three bits with each bit controlling a single light (RED, YELLOW, and GREEN).

In addition, in order to calculate the delays easier, the clock will have the following frequency:

$$f_{CLOCK} = 1\ [Hz]$$

The diagram on the next page shows how the controller can be connected to the four traffic lights and how the output bits control the individual lights.

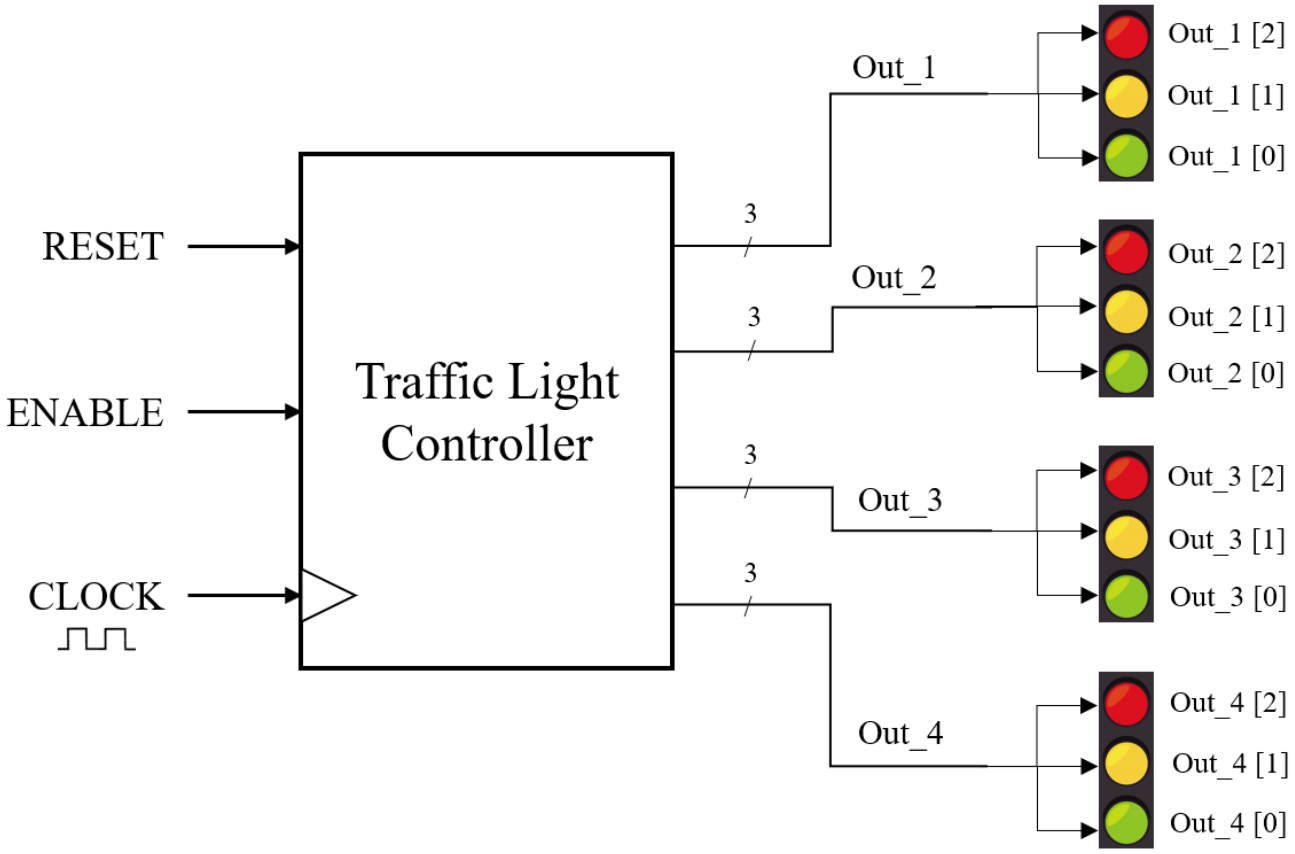


Figure 4. Controller connected to the four Traffic Lights

V. ASIC Design (VHDL Code)

Traffic_Light_Controller.vhd

```
1   Library IEEE;
2   USE IEEE.std_logic_1164.all;
3   USE IEEE.std_logic_unsigned.all;
4
5   ENTITY Traffic_Light_Controller IS
6       PORT(
7               CLOCK, RESET, ENABLE: IN   std_logic;
8               Out_1, Out_2:         OUT  std_logic_vector(2 downto 0);
9               Out_3, Out_4:         OUT  std_logic_vector(2 downto 0));
10  END Traffic_Light_Controller;
11
12  ARCHITECTURE Controller_Arch of Traffic_Light_Controller IS
13      TYPE    state_type is (S0, S1, S2, S3, S4, S5);
14      SIGNAL  current_state: state_type;
15      SIGNAL  count:    std_logic_vector(2 downto 0);  --3 bit counter
16
17      --Initializing constants for each of the lights
18      CONSTANT RED:     std_logic_vector(2 downto 0) := "100"; --RED Light
19      CONSTANT YELLOW:  std_logic_vector(2 downto 0) := "010"; --YELLOW Light
20      CONSTANT GREEN:   std_logic_vector(2 downto 0) := "001"; --GREEN Light
```

```
22  BEGIN
23
24      --Combinational and Sequential Logic
25      PROCESS(CLOCK, RESET, current_state, count)
26      BEGIN
27          IF (RESET = '1') THEN             --Resets the whole system
28              current_state <= S0;
29              count <= "000";
30          ELSIF (CLOCK'Event and CLOCK = '1') THEN
31              IF (ENABLE = '1') THEN
32                  CASE current_state IS      --Actions for each state
33                      WHEN S0 =>
34                          IF (count < "110") THEN   --7 seconds delay
35                              current_state <= S0;
36                              count <= count + 1;
37                          ELSE
38                              current_state <= S1;
39                              count <= "000";
40                          END IF;
41
42                      WHEN S1 =>
43                          IF (count < "010") THEN   --3 seconds delay
44                              current_state <= S1;
45                              count <= count + 1;
46                          ELSE
47                              current_state <= S2;
48                              count <= "000";
49                          END IF;
50
51                      WHEN S2 =>
52                          IF (count < "001") THEN   --2 seconds delay
53                              current_state <= S2;
54                              count <= count + 1;
55                          ELSE
56                              current_state <= S3;
57                              count <= "000";
58                          END IF;
59
60                      WHEN S3 =>
61                          IF (count < "110") THEN   --7 seconds delay
62                              current_state <= S3;
63                              count <= count + 1;
64                          ELSE
65                              current_state <= S4;
66                              count <= "000";
67                          END IF;
68
69                      WHEN S4 =>
70                          IF (count < "010") THEN   --3 seconds delay
71                              current_state <= S4;
72                              count <= count + 1;
73                          ELSE
74                              current_state <= S5;
75                              count <= "000";
76                          END IF;
77
78                      WHEN S5 =>
79                          IF (count < "001") THEN   --2 seconds delay
80                              current_state <= S5;
81                              count <= count + 1;
82                          ELSE
83                              current_state <= S0;
84                              count <= "000";
85                          END IF;
86
87                      WHEN OTHERS =>
88                              current_state <= S0;
89                  END CASE;
90              END IF;
91          END IF;
92      END PROCESS;
93
```

```
 93
 94
 95      --Description of the Output in Each State (Moore)
 96      PROCESS(current_state)
 97      BEGIN
 98          IF (current_state = S0) THEN
 99              Out_1 <= RED;
100              Out_2 <= RED;
101              Out_3 <= GREEN;
102              Out_4 <= GREEN;
103          ELSIF (current_state = S1) THEN
104              Out_1 <= RED;
105              Out_2 <= RED;
106              Out_3 <= YELLOW;
107              Out_4 <= YELLOW;
108          ELSIF (current_state = S2) THEN
109              Out_1 <= RED;
110              Out_2 <= RED;
111              Out_3 <= RED;
112              Out_4 <= RED;
113          ELSIF (current_state = S3) THEN
114              Out_1 <= GREEN;
115              Out_2 <= GREEN;
116              Out_3 <= RED;
117              Out_4 <= RED;
118          ELSIF (current_state = S4) THEN
119              Out_1 <= YELLOW;
120              Out_2 <= YELLOW;
121              Out_3 <= RED;
122              Out_4 <= RED;
123          ELSIF (current_state = S5) THEN
124              Out_1 <= RED;
125              Out_2 <= RED;
126              Out_3 <= RED;
127              Out_4 <= RED;
128          ELSE
129              Out_1 <= RED;
130              Out_2 <= RED;
131              Out_3 <= GREEN;
132              Out_4 <= GREEN;
133          END IF;
134      END PROCESS;
135  END Controller_Arch;
136
```

VI. Simulation Results

Before examining the simulation results for the Traffic Light Controller, the VHDL code was compiled successfully as depicted in Figure 5 below.

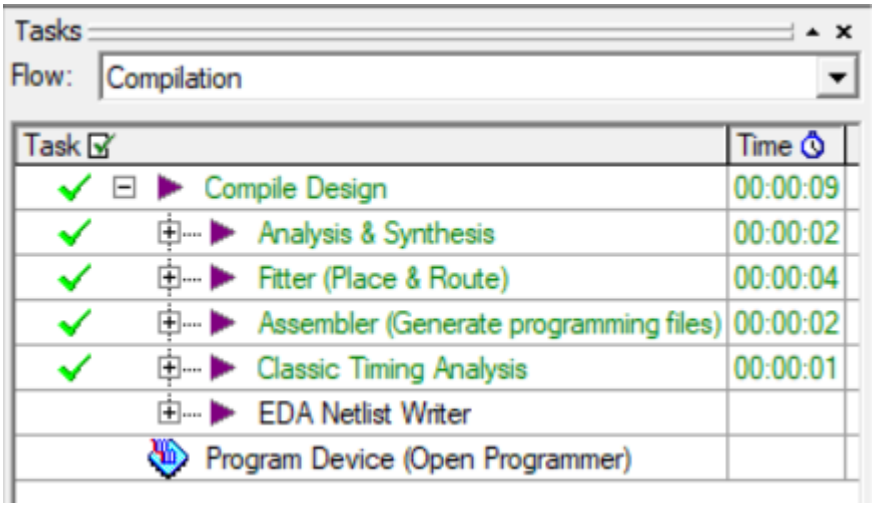


Figure 5. Successful Compilation of the VHDL Code

The RESET and ENABLE inputs were tested first in the simulation in order to inspect if they are functional. As seen in Figure 6 below, from the first second when the clock begins up until the third second, the count does not increase since the ENABLE pin is not set to 1. Moreover, six seconds after the simulation, the counter value goes back to 0 since the RESET pin has been set to 1.

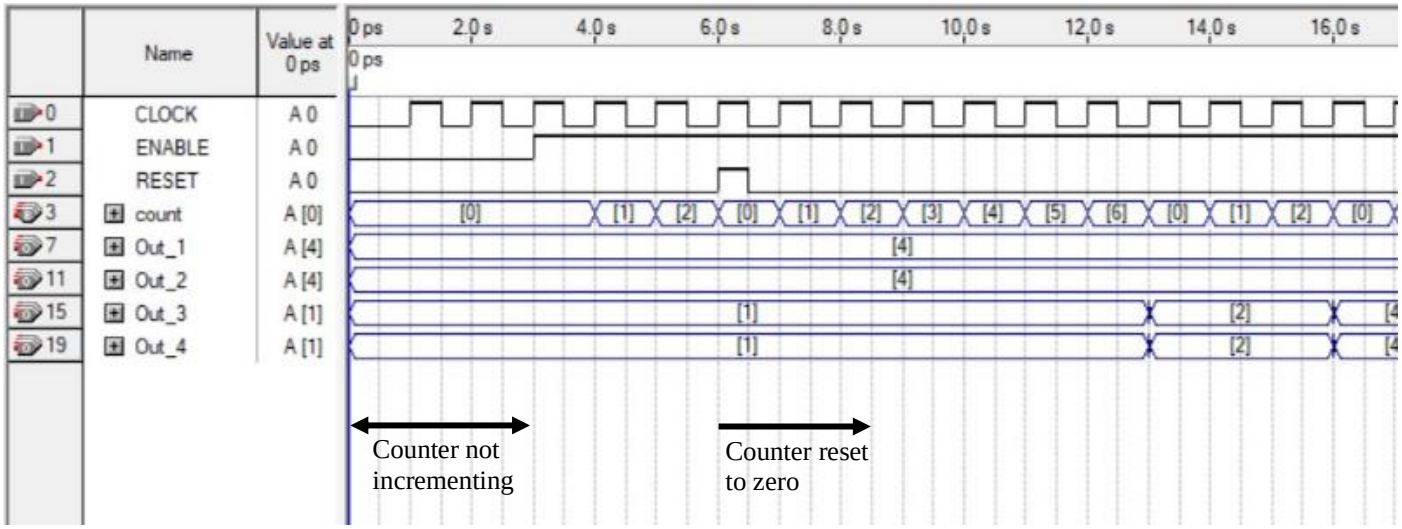


Figure 6. Testing the RESET and ENABLE Input

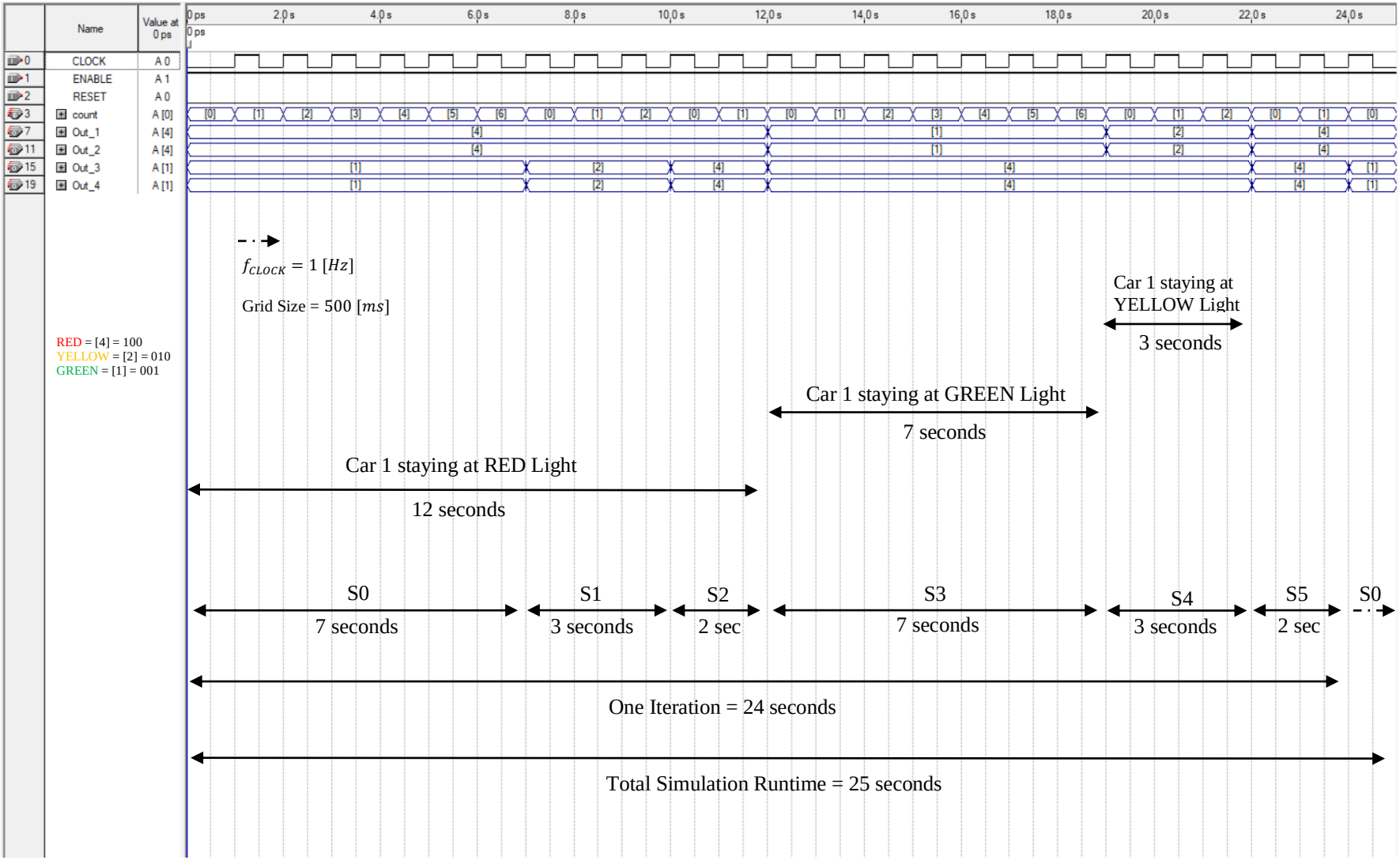


Figure 7. Simulation of the Traffic Light Controller

VII. Discussion on Results

The simulation of the design has been completed successfully. As mentioned in the Block Diagram section of this report, the clock frequency has been set to 1 [Hz]. The Grid Size for the simulator has been set to 500 [ms] or half of the clock period.

As seen in the output results in Figure 7, the total simulation runtime was set to 25 seconds. The figure shows one iteration of the state cycle and ends with the beginning of the second cycle. The first state S0 runs for 7 seconds, state S1 runs for 3 seconds, state S2 runs for 2 seconds, and states S3, S4, and S5 repeat the first three.

As discussed in the State Diagram section of this report, the goal was to make the cars stay at red light for 12 seconds, stay at green light for 7 seconds, and stay at yellow light for 3 seconds. The simulation on Figure 7 shows that the output exactly matches the desired delays set forth before the project was implemented.

Besides testing the counter and the output of the different states, the RESET and ENABLE input pins were also tested. The goal was to make both of these input pins active high and the ENABLE input should be synchronous, whereas the RESET input should be asynchronous.

As seen in Figure 6, in the Simulation Results section of this report, the ENABLE and RESET inputs were able to function efficiently. The simulation shows that the counter does not increment when the ENABLE is set to zero. In addition, it also shows that the counter can be cleared to zero once the RESET input is set to 1.

VIII. Conclusion

The project requirement for the Introduction to VLSI course is necessary because it allows students to get a practical experience of designing their own project from the ground up. The objective of this project was to design a Traffic Light Controller for use in road intersections, to implement the design by using the IEEE Standard VHDL Language, to simulate the system using the Quartus II software, and to be able to understand and explain the output results. The report clearly shows that the project fulfilled the objectives and the simulation was successful in verifying the real-world application of this controller.