

**Digital Circuits-470/470L**

Traffic Light Controller

**Date**: 18/04/2024

**Supervisor**: Dr. Herbert Azuela

**Done By**: Zahra AlMudaweb and Dana Barhoom

**ID**: A00749 and A00200

Table of Contents

[Project Description: 3](#_Toc164355761)

[Project Objectives: 3](#_Toc164355762)

[Methodology: 3](#_Toc164355763)

[Traffic Light Diagram: 3](#_Toc164355764)

[Block Diagram: 5](#_Toc164355765)

[State Diagram: 6](#_Toc164355766)

[VHDL Code: 8](#_Toc164355767)

[Test Bench Code: 10](#_Toc164355768)

[Testing: 12](#_Toc164355769)

[Conclusion: 15](#_Toc164355770)

[Future Work: 15](#_Toc164355771)

[Breakdown of Work: 16](#_Toc164355772)

Table of Figures

[Figure ‎3‑1Traffic Light Interception 4](#_Toc164355296)

[Figure ‎3‑2 Block Diagram 5](#_Toc164355297)

[Figure ‎3‑3 State Diagram 6](#_Toc164355298)

[Figure ‎6‑1 Testbench Test 13](#_Toc164355299)

[Figure ‎6‑2 Manual Test: State 14](#_Toc164355300)

# **Project Description:**

This project uses VHDL, a hardware description language, to implement the logic of an intersection of two traffic lights. First, the design of the module and its rationale are discussed supported with multiple types of diagrams. Then, the component VHDL code and the testbench code are provided and explained along with the results waveform to clarify the logic and test the efficiency of the code.

# **Project Objectives:**

This project's objective is to acquire the skills that enable the design of a VHDL module for the traffic light controller using a state diagram to illustrate how a traffic light controller operates.

# **Methodology:**

*(all diagrams are drawn using draw.io)*

## Traffic Light Diagram:

A diagram of a traffic light

Description automatically generated

Figure ‎3‑1Traffic Light Interception

Rules of the traffic light:

1. This is an intersection between two roads, road A and road B, labeled ‘A’ and ‘B’ in figure 1.
2. Each road has a traffic sensor ‘Sb’ and ‘Sa’. In general, these sensors are intended to identify the presence of a vehicle that has stopped on the streets, or if a car is approaching the roadways.
3. Road A is the main street. This means it is always on green light until a car approaches street B.
4. Street A will remain green for at least 60 seconds, and then the lights change only when a car approaches street B.
5. If Sa is ‘1’ it means that a vehicle has approached road A. If Sb is 1 it means a vehicle is approaching street B.
6. If a vehicle approaches street B, the traffic light will change turning road B into having green light. This cycle will last for 50 seconds.
7. The light will change back to red. The exception being that if there is no vehicle on A, and street B still has a vehicle, the traffic light will continue to be green on street B.
8. The cycle of street B is then extended for 10 more seconds. In addition, if still no vehicles arrive at street A, and street B still has vehicles, the cycle will continue to have a green light.

## Block Diagram:

A diagram of a traffic light

Description automatically generated

Figure ‎3‑2 Block Diagram

This traffic light controller contains 3 inputs and 6 outputs. The three inputs are:

* Sa which represents the sensor for street A.
* Sb which represents the sensor for street B.
* Clock which represents the clock.

The six outputs are:

* Ga which represents the green light on street A.
* Ya which represents the yellow light on street A.
* Ra which represents the red light on street A.
* Gb which represents the green light on street B.
* Yb which represents the yellow light on street B.
* Rb which represents the red light on street B.

## State Diagram:

The state diagram for the traffic light control system is as follows:

A diagram of a diagram

Description automatically generated

Figure ‎3‑3 State Diagram

This to show the working of the traffic light into a state diagram:

1. S0: this is the initial state which contains Ga and Rb. This is because Ga and Rb have values equal to ‘1’ at this state, which is why they are present. All the other output variables have value equal to ‘0’.
2. S0 to S5: the output of these states remains the same, where Ga and Rb have values equal to ‘1’ and all the other output variables are ‘0’. This is because, based on the rules, street A is supposed to remain green for 60s and street B supposed to be red during this time. Therefore, the state transitions occur every clock cycle after 10s.
3. Sb’: this is the value of the sensor (shown in pink in the state diagram), where it has the value Sb = 0. When Sb = 0, it must remain in the same state, else if Sb =1 it must move to the next state S6 (shown in purple).
4. S6: in this state, Ya and Rb are both equal to ‘1’. In the next clock pulse, state transition takes and moves into state S7.
5. S7: in this state both Ra and Gb are equal to ‘1’, where now the street B green traffic light is ON. This output remains the same for 5 state transition S7-S11 representing 50s of green light for street B.
6. S11: At the end of 50s, both sensor values are going to be checked, Sa and Sb. If Sa’ and Sb (shown in blue) where Sa = 0 and Sb = 1 (meaning a car is present in street B still) we must remain in the same state S11. Only when either Sa becomes ‘1’ or Sb becomes ‘0’ (shown in bright pink), the state transition take place to S12.
7. Before the green traffic light of street A is turned on in S12, the yellow traffic light of street B is ON, while the red traffic light of street A is ON. For the next clock pulse, state transition takes place going to the initial state S0 which turns ON the green traffic light of street A and the cycle completes.

# **VHDL Code:**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

Entity traffic\_light is

Port(clk, Sa, Sb: in bit;

Ra, Rb, Ga, Gb, Ya, Yb: out bit);

End traffic\_light;

Architecture traffic\_light\_arch of traffic\_light is

-- signal to keep track of current and next states

Signal state, nextstate: integer range 0 to 12;

-- enum type for the traffic light colors

Type light is (R, Y, G);

-- define SIGNALS to represent colors of TL output

Signal lightA, lightB: light;

BEGIN

-- process to determine next state and control the outputs based on current state and input signals

Process(state, Sa, Sb)

BEGIN

-- initialize output signals to default values

Ra <= '0'; Rb <= '0'; Ga <= '0'; Gb <= '0';

Ya <= '0'; Yb <= '0';

Case state is

When 0 to 4 => Ga <= '1'; Rb <= '1'; -- State 0 to 4: Green at A, Red at B

nextstate <= state+1;

When 5 => Ga <= '1'; Rb <= '1'; -- State 5: Green at A, Red at B

If Sb = '1' then

nextstate <= 6; -- transition to next state if Sb triggers a car

End if;

When 6 => Ya<= '1'; Rb<= '1'; -- State 6: Yellow at A, Red at B

nextstate <= 7;

When 7 to 10 => Ra <= '1'; Gb <= '1'; -- State 7 to 10: Red at A, Green at B

nextstate <= state+1;

When 11 =>Ra <= '1'; Gb <= '1';

If (Sa='1' or Sb='0') then

nextstate <= 12; -- transition to next state if Sa triggers a car or Sb doesn't trigger any car

End if;

When 12 => Ra <= '1'; Yb <= '1'; -- State 12: Red at A, Yellow at B

nextstate <= 0; -- transition back to initial state

End Case;

End Process;

-- process to update current state

Process(clk)

BEGIN

if clk'event and clk = '1' then

state <= nextstate; -- update current state with next state

END if;

END PROCESS;

-- assign each appropriate color to lightA based on current state

lightA <= R when Ra= '1'

Else Y when Ya= '1'

Else G when Ga= '1';

-- assign each appropriate color to lightB based on current state

lightB <= R when Rb= '1'

Else Y when Yb= '1'

Else G when Gb= '1';

END traffic\_light\_arch;

# **Test Bench Code:**

-- Testbench for traffic\_light component

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY traffic\_light\_tb IS

END ENTITY;

ARCHITECTURE behavior OF traffic\_light\_tb IS

-- Component instantiation

COMPONENT traffic\_light

Port(clk, Sa, Sb: in bit;

Ra, Rb, Ga, Gb, Ya, Yb: inout bit);

END COMPONENT;

-- Inputs

SIGNAL clk\_tb : BIT := '0';

SIGNAL Sa\_tb : BIT := '0';

SIGNAL Sb\_tb : BIT := '0';

-- Outputs

SIGNAL Ra\_tb : BIT;

SIGNAL Rb\_tb : BIT;

SIGNAL Ga\_tb : BIT;

SIGNAL Gb\_tb : BIT;

SIGNAL Ya\_tb : BIT;

SIGNAL Yb\_tb : BIT;

BEGIN

-- Instantiate the component

UUT: traffic\_light PORT MAP (

clk => clk\_tb ,

Sa => Sa\_tb ,

Sb => Sb\_tb ,

Ra => Ra\_tb ,

Rb => Rb\_tb ,

Ga => Ga\_tb ,

Gb => Gb\_tb ,

Ya => Ya\_tb ,

Yb => Yb\_tb

);

-- Clock process

PROCESS

BEGIN

clk\_tb <= '0';

WAIT FOR 5 ns;

clk\_tb <= '1';

WAIT FOR 5 ns;

END PROCESS;

-- Stimulus process

PROCESS

BEGIN

WAIT FOR 10 ns; -- Wait for initialization

-- Sa = 1, Sb = 0 (Sensor A activated)

Sa\_tb <= '1';

Sb\_tb <= '0';

WAIT FOR 60 ns;

-- Sa = 0, Sb = 0 (No sensor activated)

Sa\_tb <= '0';

Sb\_tb <= '0';

WAIT FOR 50 ns;

-- Sa = 0, Sb = 1 (Sensor B activated)

Sa\_tb <= '0';

Sb\_tb <= '1';

WAIT FOR 50 ns;

END PROCESS;

END behavior;

# **Testing:**

The component code was tested using two approaches: a testbench code and manual testing. In the testbench code, the component was initially declared, along with input and output signals representing the colors of each traffic light. The initial state had the traffic light on street A displaying green, while the one on street B showed red. The following three cases were tested:

1. When sensor Sa is triggered (indicating cars on Street A):

Results:

1. The green light on street A remains illuminated for 6 cycles (60ns), while the traffic light on street B continues to display red.

2. When both sensors are inactive (no cars on either street):

Results:

1. The traffic light on street A remains green, and the one on street B remains red for 5 cycles (50ns) since street A is the main street.

3. When sensor Sb is activated (indicating cars on Street B):

Results:

1. The traffic light on street A transitions to yellow for a complete cycle (10ns), while the traffic light on street B remains red.
2. Then, the green light on street B illuminates for 5 cycles (50ns), while the traffic light on street A shows red.

Afterward, sensor Sa is activated again, and the cases are repeated. The waveform diagram below illustrates the three test cases.

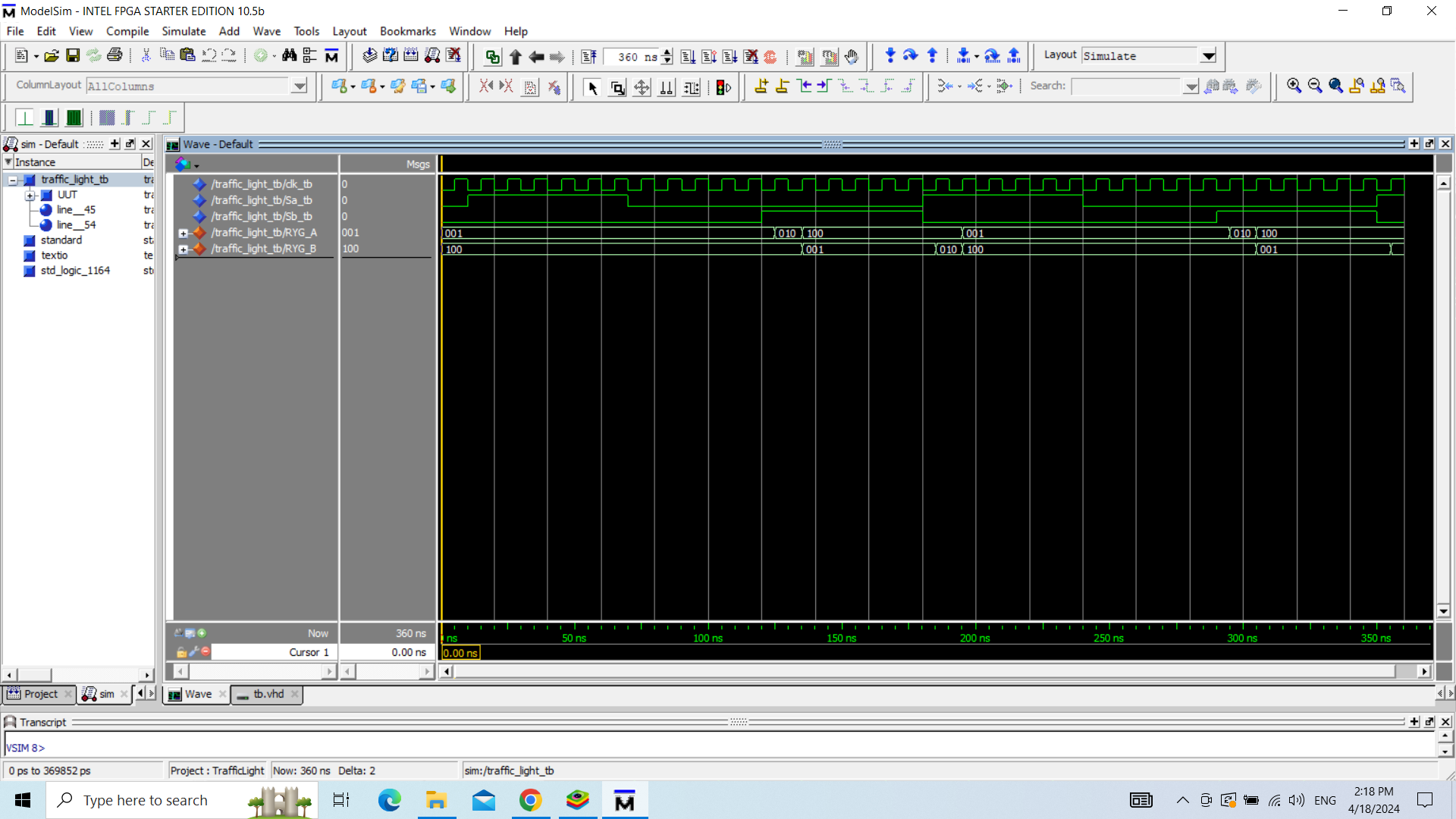


Figure ‎6‑1 Testbench Test

The second test involved manual observation of the traffic light changes, following the rules outlined in the state diagram (See Figure 3-3). The sensors were activated and deactivated accordingly to trigger the state transitions.

The results indicate that during states 0-5, the green light on street A is illuminated while the traffic light on street B displays red. When reaching state 5, the Sb sensor is activated, transitioning the traffic light on street A to yellow in state 6. From state 7 to state 11, the traffic light on street B displays green while the one on street A shows red. After that, the Sa sensor is triggered, transitioning the traffic light on street B to yellow in the next state, and the one on street A remains red. Finally, the traffic light returns to state 0 by turning the green light on street B on, and the red light on street B off.

In this test, each state represents a cycle of 10 seconds, so the green light on street A remains lit for 1 minute (6 states), while the green light on street B remains lit for 50 seconds (5 states).

The waveform diagram below provides a detailed representation of the changes in traffic light colors corresponding to the state transitions.

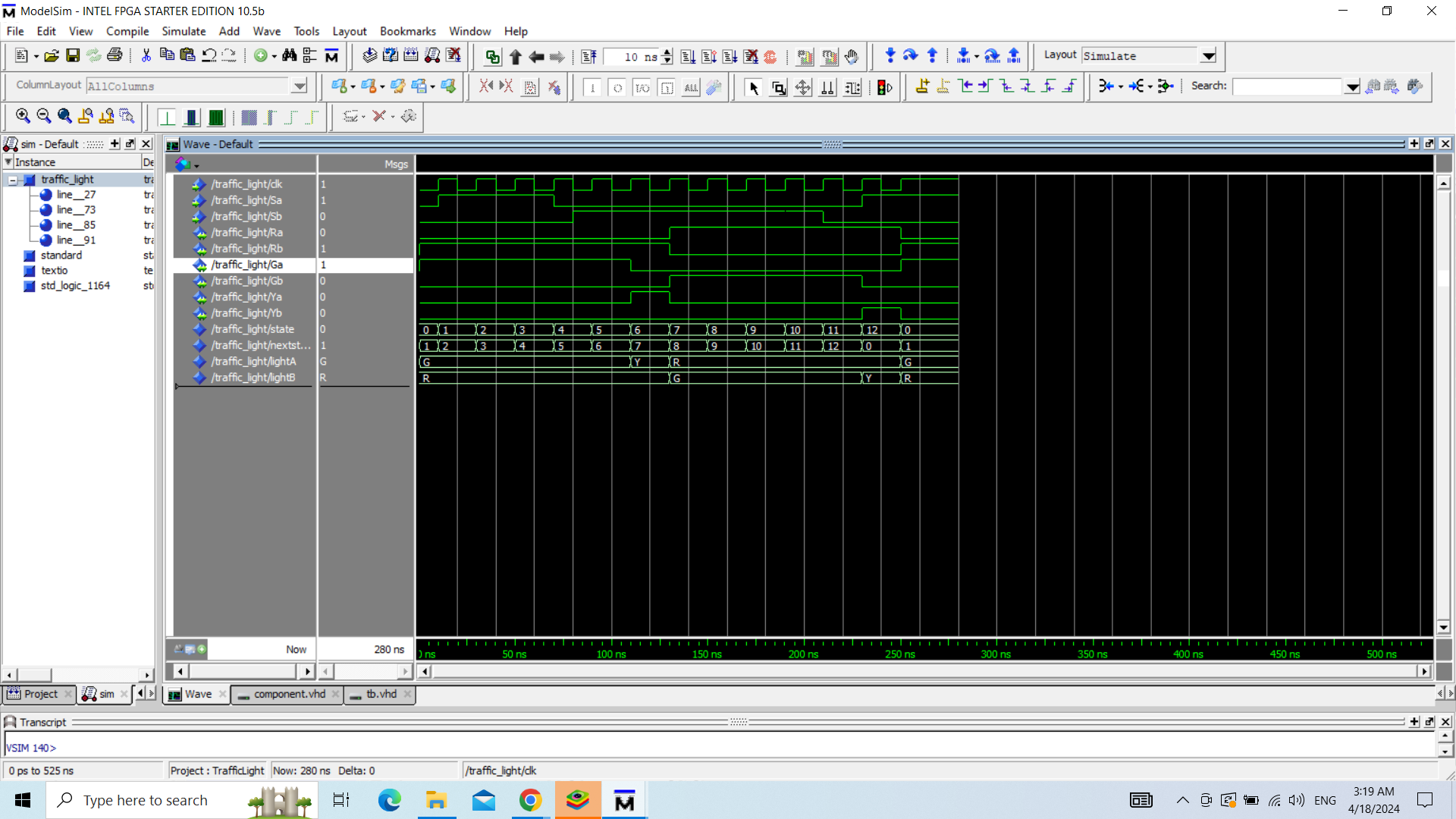


Figure ‎6‑2 Manual Test: State

# **Conclusion:**

In conclusion, through the working and the implementation of this project idea which is the traffic light controller, the purposes, and the skills of working with VHDL module were successfully acquired due to testing and troubleshooting the module to obtain a traffic light controller that illustrates how a real traffic light works. The use of state diagram and block diagram was done to demonstrate the design of the traffic light controller functions. The design of the module and the reasoning behind it were explained to help with the sequence of creating the VHDL code for the system. In order to verify the logic and assess the effectiveness of the component of the VHDL code, the testbench code, and the results, waveforms were presented and described to illustrate the traffic light controller, revealing the success of the project created.

# **Future Work:**

To develop and strengthen our project results, in the future, the testing of the code could be developed by using another program for implementation to demonstrate the traffic light control system. This could be done visually to illustrate the success of the system design created for this project. By completing this, in addition to the VHDL module testing using Modelsim, which was used to complete this project, another program could be used to demonstrate the visual interpretation of how the traffic system works using VHDL. This can be done through the use of a, for example, 3rd national FPGA to implement the system through the use of actual hardware components. This would give a robust interpretation of the application of our project displayed using hardware components of how a real traffic light controller works. This would be a strong indicator of the success of the system created in this project. Consequently, it would be a great visual display of the accomplishment of our goals and objectives.

# **Breakdown of Work:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tasks | Assigned | Due Date | Meeting Minutes/Date | Completion |  |
| Topic Research | **Dana/Zahra** | **30th March** | **60 min/ 26th March** | **Completed by both members on time** | Agreed topic: Traffic Light |
| Planning Distribution | **Zahra** | **30th March** | **30 min/30th March** | **Completed on time** |  |
| Project Description/Objectives | **Dana** | **30th March** | **30 min/30th March** | **Completed on time** |  |
| Traffic Light Rules | **Zahra** | **4th April** | **-** | **Completed on time** |  |
| Traffic Light Diagrams | **Dana** | **4th April** | **-** | **Completed on time** |  |
| VHDL Code | **Dana** | **7th April** | **60 min/ 9th April (team meeting to discuss the code)** | **Completed on time** |  |
| VHDL Testbench | **Zahra** | **9th April** | **60 min/ 11th April (team meeting to discuss the code)** | **Completed on time** |  |
| Testing Discussion | **Dana/Zahra** | **12th April** | **120 min/ 12th April** | **Completed on time** |  |
| Conclusion | **Zahra** | **13th April** | **-** | **Completed on time** |  |
| Future Works | **Dana** | **14th April** | **-** | **Completed on time** |  |