

THE UNIVERSITY OF THE WEST INDIES
ST. AUGUSTINE, TRINIDAD & TOBAGO, WEST INDIES

FACULTY OF ENGINEERING

Department of Electrical & Computer Engineering

BSc. in Electrical & Computer Engineering

ECNG 3016

Digital Electronics III

Mini-Project

Urban Traffic Management at UWI South-Gate Intersection
using IoTs and FPGA Technology

Group: Digitizers

Group Member	ID#
Brandon Heera	816030025
Denali Mc Knight	816032736
Ashanni Sonny	816027848

Course Lecturer: Dr. Marcus George

Date Submitted: 05/18/2024



CHEATING, PLAGIARISM AND COLLUSION DECLARATION FORM

According to Rules 3.31 and 3.32 of The UWI Faculty of Engineering Undergraduate Regulations and Syllabuses 2018/2019:

3.31 "Cheating, Plagiarism and Collusion are serious offences under University Regulations.



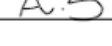
- (a) Cheating is any attempt to benefit one's self or another by deceit or fraud.
- (b) Plagiarism is the unauthorised and/or unacknowledged use of another person's intellectual efforts and creations howsoever recorded, including whether formally published or in manuscript or in typescript or other printed or electronically presented form and includes taking passages, ideas or structures from another work or author without proper and unequivocal attribution of such source(s), using the conventions for attributions or citing used in this University. Plagiarism is a form of cheating.
- (c) For the purposes of these Regulations, 'collusion' shall mean the unauthorised or unlawful collaboration or agreement between two or more students in the preparation, writing or production of a course assignment for examination and assessment, to the extent that they have produced the same or substantially the same paper, project report, as the case may be, as if it were their separate and individual efforts, in circumstances where they knew or had reason to know that the assignment or a part thereof was not intended to be a group project, but was rather to be the product of each student's individual efforts.

3.32 Cheating, plagiarism and collusion shall be reported to the Campus Committee on Examinations and the penalties would be in accordance with the University Examination Regulations."

We, the members of Group Digitizers, have read and understood Rules 3.31 and 3.32 of The UWI Faculty of Engineering Undergraduate Regulations and Syllabuses 2018/2019 on Cheating, Plagiarism and Collusion.

We understand that this submission is subject to the electronic plagiarism checker, Turnitin.

We declare that this assignment is our own work and does not involve cheating, plagiarism or collusion.

No.	Name in block letters	Signature	Date
1	Brandon Heera		18/05/24
2	Denali Mc Knight		18/05/24
3	Ashanni Sonny		18/05/24
4			
5			
6			
7			
8			
9			
10			

Contents

Overview	1
1. System Modelling	2
a) Traffic Density (No, Low/Mild, Moderate, Heavy)	2
b) Traffic Light Malfunction	2
c) Vehicular Malfunction	3
d) Existence of Vehicular Accidents	3
e) Adverse Weather Conditions	4
f) Ambulance Thoroughfare	4
g) Protest Activity	5
2. System Design	6
a) FSM-D Interface Definition.....	6
b) Datapath Design.....	7
c) Datapath Interface Definition	8
d) Control Interface Definition.....	8
e) FSM-D Architectural Model.....	9
3. FSM System Control.....	10
a) Specification	10
b) State Transition Tables	16
Under Normal circumstances for the transition table is as follows:	16
c) State Diagram.....	21
4. System Implementation	22

5. System Verification	28
Results.....	46
6. On-board Testing	76
7. Appendix.....	86
7.1 Traffic control system interface.	86
7.2 Datapath	91
7.3 Frequency Divider	97
7.4 Counters	98
7.5 State logic process (lights).....	99
7.6 Control Path	108

Overview

This project details the design and implementation of a realistic Urban Traffic Management system for the UWI South-Gate Intersection using IoTs and FPGA technology. It is a complex engineering problem, below is a diagram of the layout of the junction, it shows the traffic flow in the junction.

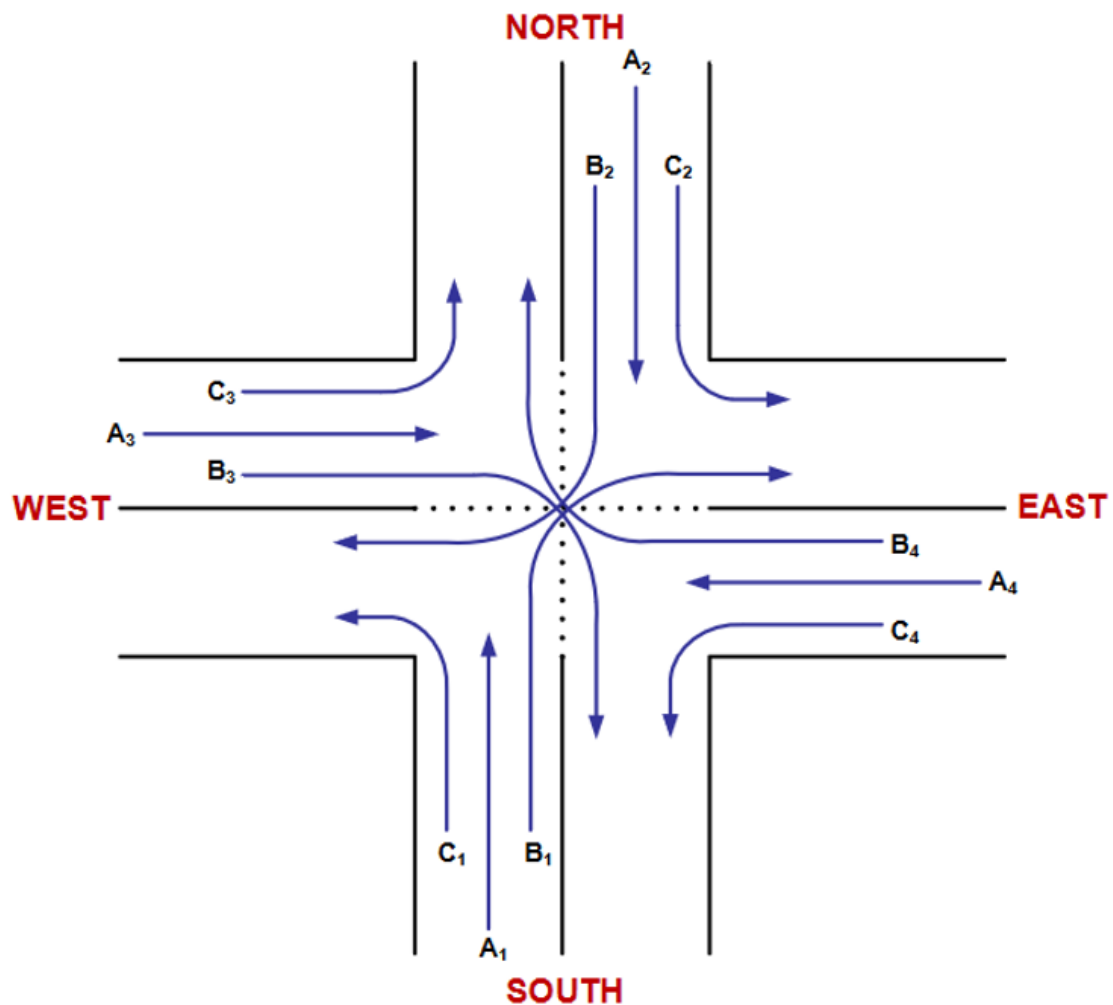


Figure 1: Outline of traffic flow at the UWI South-Gate intersection

1. System Modelling

The following scenarios were considered for the development of the Traffic Management system.

a) Traffic Density (No, Low/Mild, Moderate, Heavy)

Realistic Scenario	Heavy traffic at A3
Assumptions	All traffic lights work as intended. There are no other lanes with heavy traffic.
Impact	A3 has moderate traffic every afternoon/evening due to the large number of vehicles leaving the Port of Spain area causing major backup along the eastbound lane.
Solution Proposal	Alter the timings of the A3 lights allowing them a reduced red-light interval and in turn greater green light interval.
Implications	This proposed solution reduces the congestion along the eastbound lane by allowing a greater flow in this direction. All other lanes would experience a slight increase in red-light wait times.

b) Traffic Light Malfunction

Realistic Scenario	B3 traffic light malfunction. Light is completely off.
Assumptions	All other traffic lights function as they should.
Impact	B3 traffic backed up due to hesitant drivers to proceed south. Higher probability of accidents due to collisions between B3 and A1 or B1 or B2 or A4.
Solution Proposal	Use traffic indicators to direct B3 traffic into A3, use the U-turn at Pasea lights then proceed West towards the C4 exit.
Implications	This proposed solution prevents backup of cars on the eastbound lane heading south and allows for a safe alternative while the B3 traffic light is being addressed.

c) Vehicular Malfunction

Realistic Scenario	A vehicle has malfunctioned and is shut down at A1
Assumptions	All traffic lights work as intended. All other lanes are clear of vehicular malfunctions.
Impact	A1 traffic backed up due to the stopped vehicle. Minor traffic disruption is also occurring in lane C1 and B1 as cars from A1 must merge into these lanes to pass the malfunctioned vehicle.
Solution Proposal	Use traffic indicators to direct A1 traffic into C1, use the Curepe interchange to head back eastbound and exit north using C3. Cars that pass the malfunctioned vehicle can still travel A1 when the opportunity is theirs.
Implications	This proposed solution prevents backup of cars using A1 on the northbound and allows for a safe alternative until the car stopped in A1 is moved.

d) Existence of Vehicular Accidents

Realistic Scenario	Accident occurs on the eastbound lane causing heavy traffic at A3 and B3.
Assumptions	All traffic lights work as intended. There are no blockages on any other lane.
Impact	Drivers seek an alternative route causing heavy traffic on all lanes of the eastbound highway.
Solution Proposal	Use traffic indicators to direct cars into the C3 lane to head northbound make a U-turn at UWI then head southbound into A2 and C2. This would allow traffic from A3 and B3 to bypass the accident and continue in their respective directions safely.
Implications	The traffic congestion due to the accident is reduced by this proposal. It may have a slightly higher risk of accidents as both A3 and B3 are required to merge into C3.

e) Adverse Weather Conditions

Realistic Scenario	Heavy rains reduce the visibility of traffic lights and other cars.
Assumptions	All traffic lights work as intended. The only effect of the heavy rains is reduced visibility.
Impact	The probability of accidents occurring increases as drivers have greater difficulty in seeing the colour of the traffic lights as well as vehicles in front of them leading to them driving more cautiously.
Solution Proposal	Increase the timings of the period between the switch of one light turning red and the other turning green to allow vehicles that were not able to safely stop on the red-light safe passage. Increase the time of the amber lights to provide vehicles with greater time to slow down and come to a complete stop.
Implications	This method only mitigates the effects of the issue but does not provide a solution to the issue. Flow is sacrificed for increased safety as a result of these timing variations.

f) Ambulance Thoroughfare

Realistic Scenario	An ambulance wants to go from B2 to westbound.
Assumptions	All traffic lights work as intended. The only emergency vehicle approaching the 4-way intersection is the ambulance stated above.
Impact	Cars try to manoeuvre for the ambulance to pass but this method is not as efficient and increases the probability of an accident occurring.
Solution Proposal	Use cameras and microphones to sense approaching emergency vehicles with their lights and sirens on and allow the traffic lights on the lane where the vehicle is in to turn green while all others turn and remain red to allow swift throughfare of the emergency vehicle.
Implications	It requires HD camera and microphone availability in all directions of the intersection. The light must stay on for a sufficient period to allow the vehicle to pass. The ambulance can safely reach its desired lane quickly.

g) Protest Activity

Realistic Scenario	A protest on the highway causes a blockage at A4
Assumptions	All traffic lights work as intended. There are no other protests at the intersection and all other lanes are clear to pass.
Impact	A4 traffic is backed up due to the protest causing them to seek an alternative route.
Solution Proposal	Use traffic indicators to direct cars into the C4 lane where they would travel southbound and eventually return facing north into A1 where they would merge into C1 and enter the westbound lane.
Implications	This proposed solution prevents backup of cars on the westbound lane A4 and allows for a safe alternative while A4 is blocked by the protest.

2. System Design

a) FSM-D Interface Definition

The FSM-D Interface Definition outlines the structure, behaviour, and interactions of a finite state machine (FSM) within a given system.

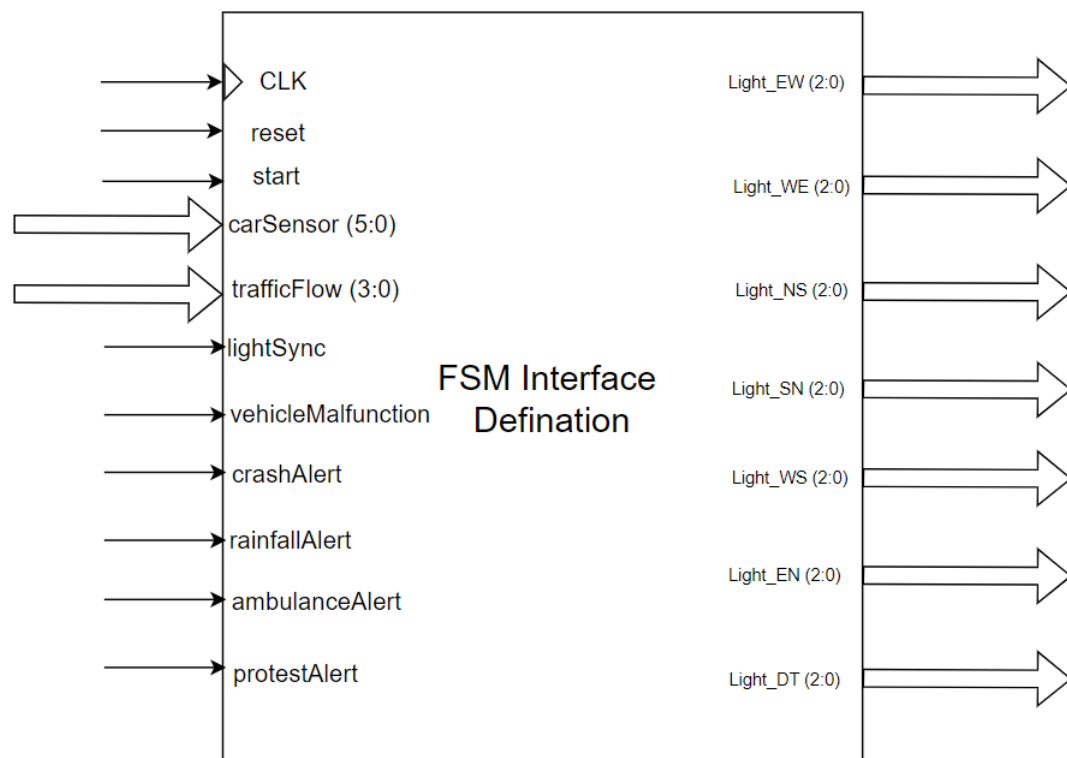


Figure 2: FSM Interface Definition

Sensor Inputs:

The system has multiple inputs which have been briefly listed before, below is a complete list of the sensor inputs of the system and their representative name to the system.

Sensor Name	Operation
trafficFlow	This sensor monitors the number of cars passing through the junction.
lightSync	This sensor monitors if there is an error or malfunction on any of the traffic lights.
vehicleMalfunction	This sensor monitors the junction for accidents between vehicles.
crashAlert	This sensor monitors the junction for accidents between vehicles.
rainfallAlert	This sensor monitors the junction for heavy rain.
ambulanceAlert	This sensor detects an ambulance passing through the junction.
protestAlert	This sensor detects protest activity at the junction.

b) Datapath Design

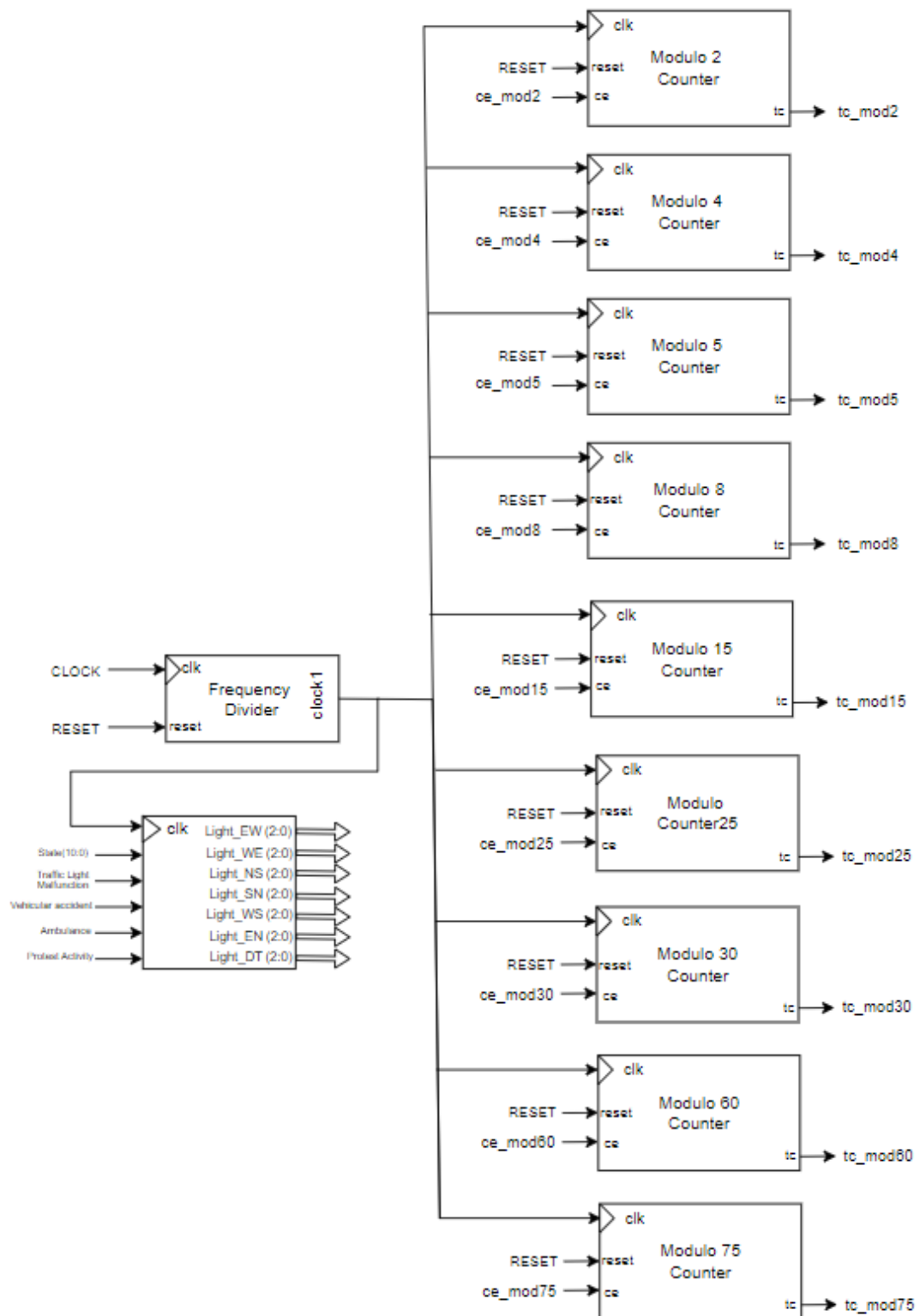


Figure 3: Datapath Design

c) Datapath Interface Definition

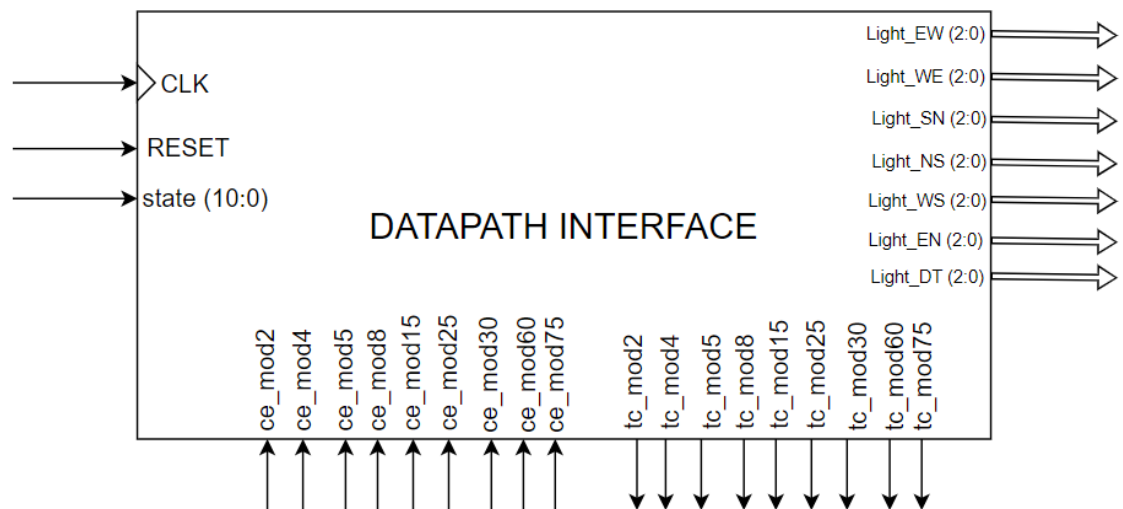


Figure 4: Datapath Interface Definition

d) Control Interface Definition

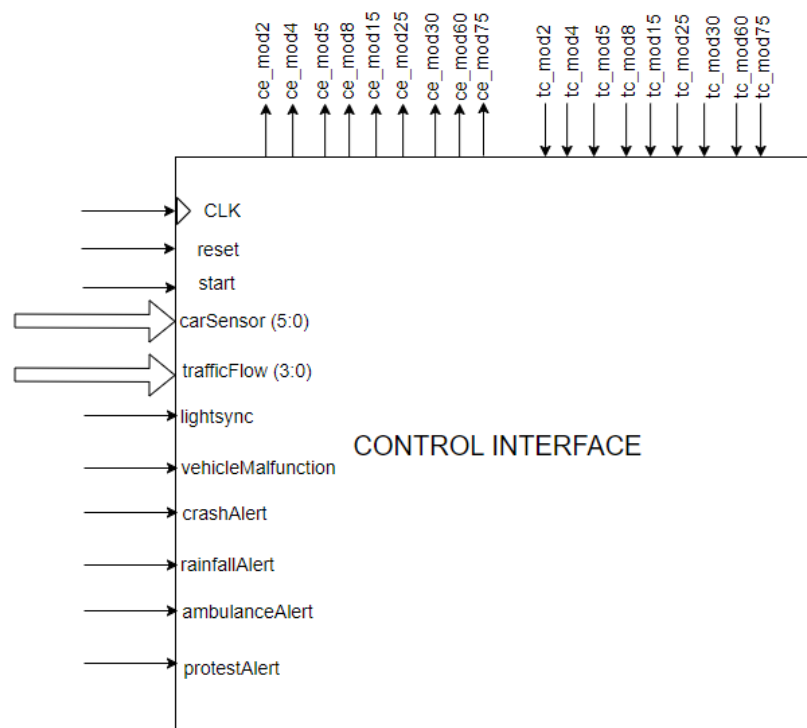


Figure 5: Control Interface Definition

e) FSM-D Architectural Model

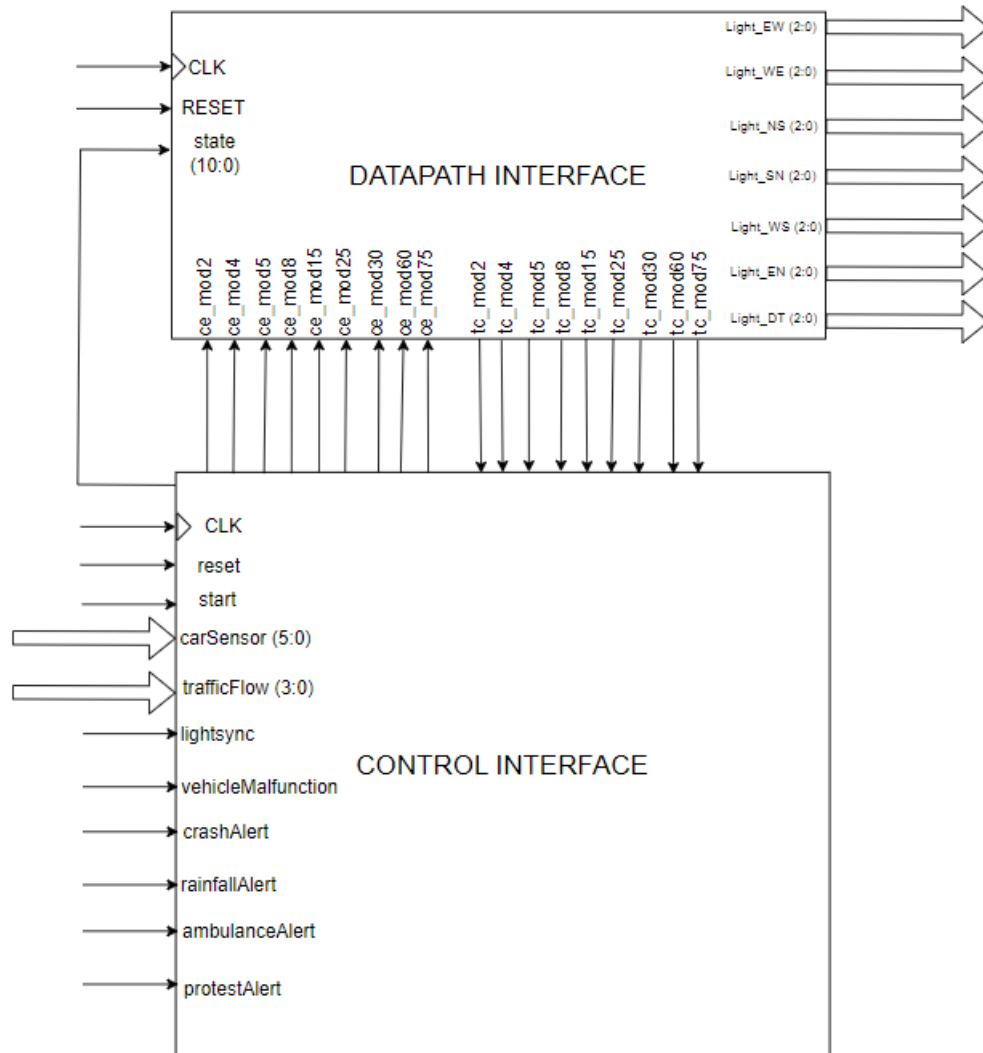


Figure 6: FSM-D Architectural Model

3. FSM System Control

a) Specification

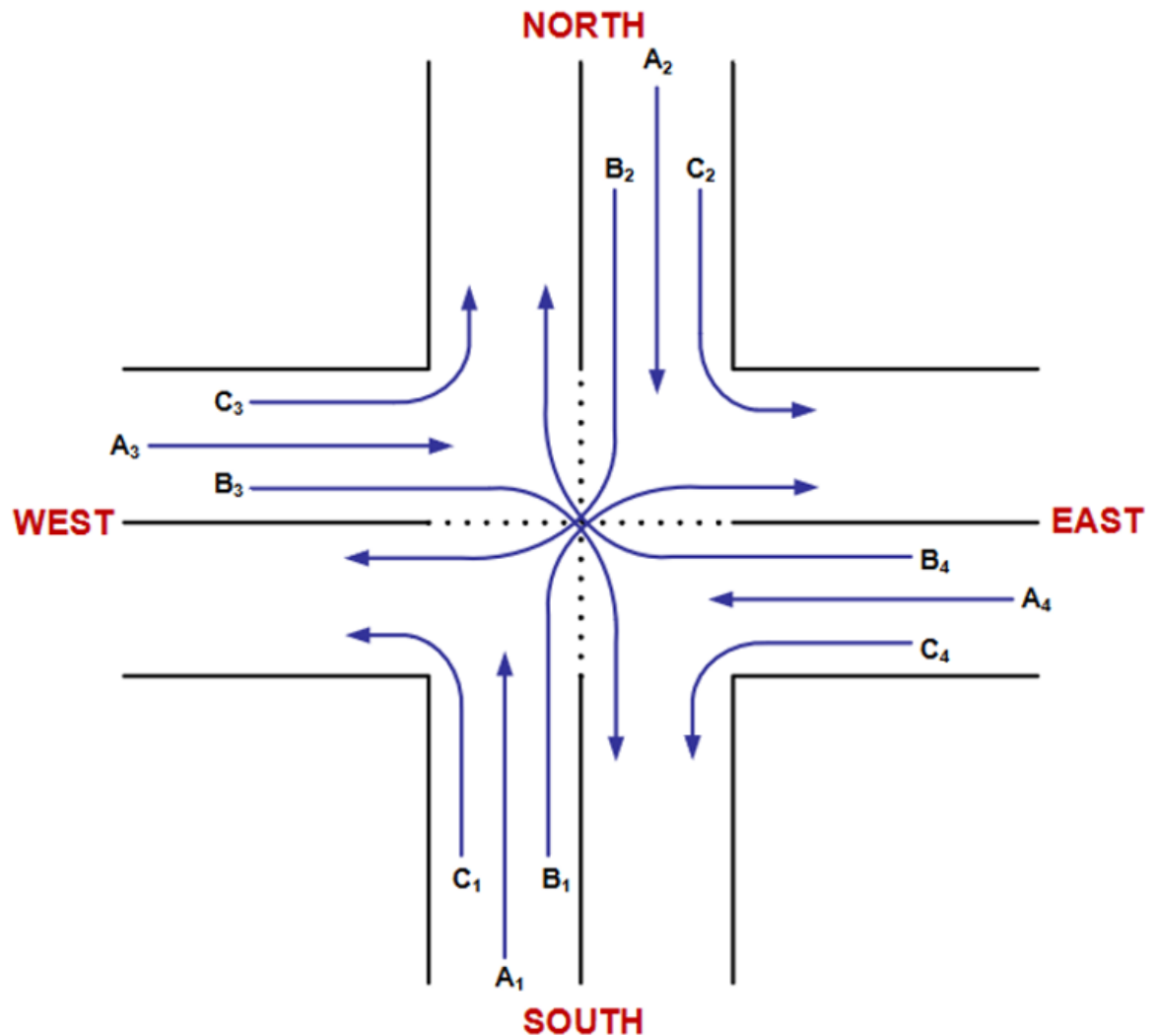


Figure 7: Diagram of traffic flow at the UWI South-Gate intersection

This system controls the traffic lights at the UWI South Gate intersection. The traffic flow through the junction is controlled by the following:

- The south lights control A₁ B₁
- The north lights control A₂ B₂
- The West-East and East-West lights control A₃ A₄

- The West-South and East-North lights respectively control $B_3 B_4$

It is to be noted that $C_1 C_2 C_3 C_4$ are merging lanes hence they will not be discussed as once the road is clear they can pass. This will help reduce the complexity of the system. It must also be noted that the lights on certain lanes turn on and off at the same time hence they can be grouped together to make the following:

- Group 1: $A_1 B_1 = G_1$
- Group 2: $A_2 B_2 = G_2$
- Group 3: $A_3 A_4 = G_3$
- Group 4: $B_3 B_4 = G_4$
- Group All: G_A this is a special use case for turning all lights red.
- Group All Blue: G_B this is a special use case for turning on the blue guidance lights.

The system iterates through the groups lighting each relevant light of the group in the order:

$$G_3 \rightarrow G_A \rightarrow G_2 \rightarrow G_A \rightarrow G_1 \rightarrow G_A \rightarrow G_4 \rightarrow G_A \rightarrow G_3 \dots$$

The traffic management system iterates through turning on the green light for a group for a given period then the yellow light for a given period then the red light of the group. A new Group is selected to be turned on, it iterates through each group until it reaches the start and loops continuously. The sequency follows:

$$\begin{aligned} 001\ 001\ 100\ 001\ 000 &\rightarrow 001\ 001\ 010\ 001\ 000 \rightarrow 001\ 001\ 001\ 001\ 000 \\ &\rightarrow 001\ 001\ 001\ 100\ 000 \rightarrow 001\ 001\ 100\ 010\ 000 \\ &\rightarrow 001\ 001\ 001\ 001\ 000 \rightarrow 100\ 001\ 001\ 001\ 000 \\ &\rightarrow 010\ 001\ 001\ 001\ 000 \rightarrow 001\ 001\ 001\ 001\ 000 \\ &\rightarrow 001\ 100\ 001\ 001\ 000 \rightarrow 001\ 010\ 001\ 001\ 000 \\ &\rightarrow 001\ 001\ 001\ 001\ 000 \rightarrow 001\ 001\ 100\ 001\ 000 \rightarrow \dots \end{aligned}$$

The current light that is on in each group is represented by a '1' while a light off is represented by a '0'. Each group is represented by 3-bit values ($g_N y_N r_N$) where N is the group number, g is a green light, y is a yellow light and r is a red light. A special case is G_B it consists of ($b_1 b_2 b_3$) where they correspond to blue lights in the order west, south, east. The code 001 001 100 001 000 means $G_1 G_2 G_4$ are on a red light while G_3 is on a green light and all blue lights are off. The code 001 001 010 001 000 means $G_1 G_2 G_4$ are on a red light while G_3 is on a yellow light and all blue lights are off. The code 001 001 001 001 000 means G_A are on a red light and all blue lights are off.

Under normal circumstances

- The major traffic flows, A_3 and A_4 , are given 60 seconds to flow on green, and 8 seconds on yellow. (G_3)
- A_1, B_1, A_2 and B_2 are given 30 seconds on green and 5 seconds on yellow. ($G_1 G_2$)
- B_3 and B_4 are given 15 seconds on green and 5 seconds on yellow. (G_4)

With heavy traffic at A_3 and A_4

- The major traffic flows, A_3 and A_4 , are given 75 seconds to flow on green, and 8 seconds on yellow. (G_3)
- A_1, B_1, A_2 and B_2 are given 25 seconds on green and 5 seconds on yellow. ($G_1 G_2$)
- B_3 and B_4 are given 15 seconds on green and 5 seconds on yellow. (G_4)

With a traffic light malfunction at B₃

- The traffic light will indicate to the traffic at B₃ to merge to A₃.
- The B₃ light will flash red to indicate the existence of a malfunction if possible.
- The detour light associated with B₃ the west light (b₁) will be illuminated to indicate the merge and need for detour.
- The other lights function as aforementioned in normal circumstances, however B₃ will be removed from its group and have an X to show its flashing in the transition table.

With vehicular malfunction at A₁

- The traffic light will indicate to the traffic at A₁ to merge into C₁.
- The A₁ light will continue to work on its usual timing however it will be always flashing throughout the operation. This will still be allowing cars to pass at the when required as cars will still be able to drive around the malfunctioned vehicle.
- The detour light associated with A₁ the south light (b₂) will be illuminated to indicate the merge and need for detour.
- The other lights function as aforementioned in normal circumstances, however A₁ will be removed from its group and have an X to show its flashing in the transition table but still operate on the same timing as the group.

With vehicular accident at A₃

- The traffic lights will indicate to the traffic at A₃ to merge into C₃.
- The A₃ light will continue to work on its usual timing however it will be always flashing throughout the operation while still allowing cars to pass at the when required as cars may still be able to drive around the vehicular accident.
- The detour light associated with A₃ the west light (b₁) will be illuminated to indicate the merge and need for detour.
- The other lights function as aforementioned in normal circumstances, however A₃ will be removed from its group and have an X to show its flashing in the transition table but still operate on the same timing as the group.

With adverse weather conditions

- The major traffic flows, A₃ and A₄, are given 1 minute to flow on green, and 15 seconds on yellow.
- A₁, B₁, A₂ and B₂ are given 30 seconds on green and 8 seconds on yellow.
- B₃ B₄ are given 15 seconds on green and 8 seconds on yellow.
- The wait time between red at one light to green at another is 4 seconds.

Ambulance throughfare at B₂

- All lights except B₂ are red.
- This light is turned green for 15 seconds the light then yellow for 3 seconds.
- B₂ will be removed from its group so its operation can be seen.

Protest Activity at A₄

- The traffic light will indicate A₄ traffic to merge into C₄.
- The A₄ light will always flash in red to indicate the existence of blockage and does not allow cars to pass as there could be accidents involving humans and loss of life.
- The detour light associated with A₄ the East light (b₃) will be illuminated to indicate the merge and need for detour.
- The other lights function as aforementioned in normal circumstances, however A₄ will be removed from its group and have an X to show its flashing.

b) State Transition Tables

Under Normal circumstances for the transition table is as follows:

Present State	tc_mod2	tc_mod5	tc_mod8	tc_mod15	tc_mod30	tc_mod60	Next State	$g_1y_1r_1 g_2y_2r_2 g_3y_3r_3 g_4y_4r_4 b_1b_2b_3$
S0	0	0	0	0	0	0	S0	001 001 100 001 000
S0	0	0	0	0	0	1	S1	001 001 010 001 000
S1	0	0	0	0	0	0	S1	001 001 010 001 000
S1	0	0	1	0	0	0	S2	001 001 001 001 000
S2	0	0	0	0	0	0	S2	001 001 001 001 000
S2	1	0	0	0	0	0	S3	001 100 001 001 000
S3	0	0	0	0	0	0	S3	001 100 001 001 000
S3	0	0	0	0	1	0	S4	001 010 001 001 000
S4	0	0	0	0	0	0	S4	001 010 001 001 000
S4	0	1	0	0	0	0	S5	001 001 001 001 000
S5	0	0	0	0	0	0	S5	001 001 001 001 000
S5	1	0	0	0	0	0	S6	100 001 001 001 000
S6	0	0	0	0	0	0	S6	100 001 001 001 000
S6	0	0	0	0	1	0	S7	010 001 001 001 000
S7	0	0	0	0	0	0	S7	010 001 001 001 000
S7	0	1	0	0	0	0	S8	001 001 001 001 000
S8	0	0	0	0	0	0	S8	001 001 001 001 000
S8	1	0	0	0	0	0	S9	001 001 001 100 000
S9	0	0	0	0	0	0	S9	001 001 001 100 000
S9	0	0	0	1	0	0	S10	001 001 001 010 000
S10	0	0	0	0	0	0	S10	001 001 001 010 000
S10	0	1	0	0	0	0	S11	001 001 001 001 000
S11	0	0	0	0	0	0	S11	001 001 001 001 000
S11	1	0	0	0	0	0	S0	001 001 100 001 000

Under heavy traffic conditions the transition table is as follows:

Present State	tc_mod2	tc_mod5	tc_mod8	tc_mod15	tc_mod25	tc_mod75	Next State	$g_1y_1r_1 g_2y_2r_2 g_3y_3r_3 g_4y_4r_4 b_1b_2b_3$
S0	0	0	0	0	0	0	S0	001 001 100 001 000
S0	0	0	0	0	0	1	S1	001 001 010 001 000
S1	0	0	0	0	0	0	S1	001 001 010 001 000
S1	0	0	1	0	0	0	S2	001 001 001 001 000
S2	0	0	0	0	0	0	S2	001 001 001 001 000
S2	1	0	0	0	0	0	S3	001 100 001 001 000
S3	0	0	0	0	0	0	S3	001 100 001 001 000
S3	0	0	0	0	1	0	S4	001 010 001 001 000

S4	0	0	0	0	0	0	S4	001 010 001 001 000
S4	0	1	0	0	0	0	S5	001 001 001 001 000
S5	0	0	0	0	0	0	S5	001 001 001 001 000
S5	1	0	0	0	0	0	S6	100 001 001 001 000
S6	0	0	0	0	0	0	S6	100 001 001 001 000
S6	0	0	0	0	1	0	S7	010 001 001 001 000
S7	0	0	0	0	0	0	S7	010 001 001 001 000
S7	0	1	0	0	0	0	S8	001 001 001 001 000
S8	0	0	0	0	0	0	S8	001 001 001 001 000
S8	1	0	0	0	0	0	S9	001 001 001 100 000
S9	0	0	0	0	0	0	S9	001 001 001 100 000
S9	0	0	0	1	0	0	S10	001 001 001 010 000
S10	0	0	0	0	0	0	S10	001 001 001 010 000
S10	0	1	0	0	0	0	S11	001 001 001 001 000
S11	0	0	0	0	0	0	S11	001 001 001 001 000
S11	1	0	0	0	0	0	S0	001 001 100 001 000

Traffic light malfunction transition table:

Present State	tc_mod2	tc_mod5	tc_mod8	tc_mod15	tc_mod30	tc_mod60	Next State	g ₁ y ₁ r ₁ g ₂ y ₂ r ₂ g ₃ y ₃ r ₃ g ₄ y ₄ r ₄ b ₁ b ₂ b ₃	B ₃
S0	0	0	0	0	0	0	S0	001 001 100 001 100	X
S0	0	0	0	0	0	1	S1	001 001 010 001 100	X
S1	0	0	0	0	0	0	S1	001 001 010 001 100	X
S1	0	0	1	0	0	0	S2	001 001 001 001 100	X
S2	0	0	0	0	0	0	S2	001 001 001 001 100	X
S2	1	0	0	0	0	0	S3	001 100 001 001 100	X
S3	0	0	0	0	0	0	S3	001 100 001 001 100	X
S3	0	0	0	0	1	0	S4	001 010 001 001 100	X
S4	0	0	0	0	0	0	S4	001 010 001 001 100	X
S4	0	1	0	0	0	0	S5	001 001 001 001 100	X
S5	0	0	0	0	0	0	S5	001 001 001 001 100	X
S5	1	0	0	0	0	0	S6	100 001 001 001 100	X
S6	0	0	0	0	0	0	S6	100 001 001 001 100	X
S6	0	0	0	0	1	0	S7	010 001 001 001 100	X
S7	0	0	0	0	0	0	S7	010 001 001 001 100	X
S7	0	1	0	0	0	0	S8	001 001 001 001 100	X
S8	0	0	0	0	0	0	S8	001 001 001 001 100	X
S8	1	0	0	0	0	0	S9	001 001 001 100 100	X
S9	0	0	0	0	0	0	S9	001 001 001 100 100	X
S9	0	0	0	1	0	0	S10	001 001 001 010 100	X
S10	0	0	0	0	0	0	S10	001 001 001 010 100	X
S10	0	1	0	0	0	0	S11	001 001 001 001 100	X
S11	0	0	0	0	0	0	S11	001 001 001 001 100	X
S11	1	0	0	0	0	0	S0	001 001 100 001 100	X

Vehicular malfunction transition table:

Present State	tc_mod2	tc_mod5	tc_mod8	tc_mod15	tc_mod30	tc_mod60	Next State	$g_1y_1r_1 g_2y_2r_2 g_3y_3r_3 g_4y_4r_4$ $b_1b_2b_3$	A_3
S0	0	0	0	0	0	0	S0	001 001 100 001 010	X
S0	0	0	0	0	0	1	S1	001 001 010 001 010	X
S1	0	0	0	0	0	0	S1	001 001 010 001 010	X
S1	0	0	1	0	0	0	S2	001 001 001 001 010	X
S2	0	0	0	0	0	0	S2	001 001 001 001 010	X
S2	1	0	0	0	0	0	S3	001 100 001 001 010	X
S3	0	0	0	0	0	0	S3	001 100 001 001 010	X
S3	0	0	0	0	1	0	S4	001 010 001 001 010	X
S4	0	0	0	0	0	0	S4	001 010 001 001 010	X
S4	0	1	0	0	0	0	S5	001 001 001 001 010	X
S5	0	0	0	0	0	0	S5	001 001 001 001 010	X
S5	1	0	0	0	0	0	S6	100 001 001 001 010	X
S6	0	0	0	0	0	0	S6	100 001 001 001 010	X
S6	0	0	0	0	1	0	S7	010 001 001 001 010	X
S7	0	0	0	0	0	0	S7	010 001 001 001 010	X
S7	0	1	0	0	0	0	S8	001 001 001 001 010	X
S8	0	0	0	0	0	0	S8	001 001 001 001 010	X
S8	1	0	0	0	0	0	S9	001 001 001 100 010	X
S9	0	0	0	0	0	0	S9	001 001 001 100 010	X
S9	0	0	0	1	0	0	S10	001 001 001 010 010	X
S10	0	0	0	0	0	0	S10	001 001 001 010 010	X
S10	0	1	0	0	0	0	S11	001 001 001 001 010	X
S11	0	0	0	0	0	0	S11	001 001 001 001 010	X
S11	1	0	0	0	0	0	S0	001 001 100 001 010	X

Vehicular accident transition table:

Present State	tc_mod2	tc_mod5	tc_mod8	tc_mod15	tc_mod30	tc_mod60	Next State	$g_1y_1r_1 g_2y_2r_2 g_3y_3r_3 g_4y_4r_4$ $b_1b_2b_3$	A_3
S0	0	0	0	0	0	0	S0	001 001 100 001 100	X
S0	0	0	0	0	0	1	S1	001 001 010 001 100	X
S1	0	0	0	0	0	0	S1	001 001 010 001 100	X
S1	0	0	1	0	0	0	S2	001 001 001 001 100	X
S2	0	0	0	0	0	0	S2	001 001 001 001 100	X
S2	1	0	0	0	0	0	S3	001 100 001 001 100	X
S3	0	0	0	0	0	0	S3	001 100 001 001 100	X
S3	0	0	0	0	1	0	S4	001 010 001 001 100	X
S4	0	0	0	0	0	0	S4	001 010 001 001 100	X
S4	0	1	0	0	0	0	S5	001 001 001 001 100	X

S5	0	0	0	0	0	0	S5	001 001 001 001 100	X
S5	1	0	0	0	0	0	S6	100 001 001 001 100	X
S6	0	0	0	0	0	0	S6	100 001 001 001 100	X
S6	0	0	0	0	1	0	S7	010 001 001 001 100	X
S7	0	0	0	0	0	0	S7	010 001 001 001 100	X
S7	0	1	0	0	0	0	S8	001 001 001 001 100	X
S8	0	0	0	0	0	0	S8	001 001 001 001 100	X
S8	1	0	0	0	0	0	S9	001 001 001 100 100	X
S9	0	0	0	0	0	0	S9	001 001 001 100 100	X
S9	0	0	0	1	0	0	S10	001 001 001 010 100	X
S10	0	0	0	0	0	0	S10	001 001 001 010 100	X
S10	0	1	0	0	0	0	S11	001 001 001 001 100	X
S11	0	0	0	0	0	0	S11	001 001 001 001 100	X
S11	1	0	0	0	0	0	S0	001 001 100 001 100	X

Adverse weather transition table:

Present State	tc_mod4	tc_mod8	tc_mod15	tc_mod30	tc_mod60	Next State	$g_1y_1r_1g_2y_2r_2g_3y_3r_3g_4y_4r_4$ $b_1b_2b_3$
S0	0	0	0	0	0	S0	001 001 100 001 000
S0	0	0	0	0	1	S1	001 001 010 001 000
S1	0	0	0	0	0	S1	001 001 010 001 000
S1	0	0	1	0	0	S2	001 001 001 001 000
S2	0	0	0	0	0	S2	001 001 001 001 000
S2	1	0	0	0	0	S3	001 100 001 001 000
S3	0	0	0	0	0	S3	001 100 001 001 000
S3	0	0	0	1	0	S4	001 010 001 001 000
S4	0	0	0	0	0	S4	001 010 001 001 000
S4	0	1	0	0	0	S5	001 001 001 001 000
S5	0	0	0	0	0	S5	001 001 001 001 000
S5	1	0	0	0	0	S6	100 001 001 001 000
S6	0	0	0	0	0	S6	100 001 001 001 000
S6	0	0	0	1	0	S7	010 001 001 001 000
S7	0	0	0	0	0	S7	010 001 001 001 000
S7	0	1	0	0	0	S8	001 001 001 001 000
S8	0	0	0	0	0	S8	001 001 001 001 000
S8	1	0	0	0	0	S9	001 001 001 100 000
S9	0	0	0	0	0	S9	001 001 001 100 000
S9	0	0	1	0	0	S10	001 001 001 010 000
S10	0	0	0	0	0	S10	001 001 001 010 000
S10	0	1	0	0	0	S11	001 001 001 001 000
S11	0	0	0	0	0	S11	001 001 001 001 000
S11	1	0	0	0	0	S0	001 001 100 001 000

Ambulance throughfare transition table:

Present State	tc_mod2	tc_mod4	tc_mod15	Next State	$g_1Y_1r_1 g_2Y_2r_2 g_3Y_3r_3 g_4Y_4r_4 b_1b_2b_3$	$B_2(gyr)$
S0	0	0	0	S0	001 001 100 001 000	001
S0	0	0	1	S1	001 001 100 001 000	100
S1	0	0	0	S1	001 001 100 001 000	100
S1	0	1	0	S2	001 001 100 001 000	010
S2	0	0	0	S2	001 001 100 001 000	010
S2	1	0	0	S0	001 001 100 001 000	001

Protest Activity transition table:

Present State	tc_mod2	tc_mod5	tc_mod8	tc_mod15	tc_mod30	tc_mod60	Next State	$g_1Y_1r_1 g_2Y_2r_2 g_3Y_3r_3 g_4Y_4r_4 b_1b_2b_3$	A_4
S0	0	0	0	0	0	0	S0	001 001 100 001 001	X
S0	0	0	0	0	0	1	S1	001 001 010 001 001	X
S1	0	0	0	0	0	0	S1	001 001 010 001 001	X
S1	0	0	1	0	0	0	S2	001 001 001 001 001	X
S2	0	0	0	0	0	0	S2	001 001 001 001 001	X
S2	1	0	0	0	0	0	S3	001 100 001 001 001	X
S3	0	0	0	0	0	0	S3	001 100 001 001 001	X
S3	0	0	0	0	1	0	S4	001 010 001 001 001	X
S4	0	0	0	0	0	0	S4	001 010 001 001 001	X
S4	0	1	0	0	0	0	S5	001 001 001 001 001	X
S5	0	0	0	0	0	0	S5	001 001 001 001 001	X
S5	1	0	0	0	0	0	S6	100 001 001 001 001	X
S6	0	0	0	0	0	0	S6	100 001 001 001 001	X
S6	0	0	0	0	1	0	S7	010 001 001 001 001	X
S7	0	0	0	0	0	0	S7	010 001 001 001 001	X
S7	0	1	0	0	0	0	S8	001 001 001 001 001	X
S8	0	0	0	0	0	0	S8	001 001 001 001 001	X
S8	1	0	0	0	0	0	S9	001 001 001 100 001	X
S9	0	0	0	0	0	0	S9	001 001 001 100 001	X
S9	0	0	0	1	0	0	S10	001 001 001 010 001	X
S10	0	0	0	0	0	0	S10	001 001 001 010 001	X
S10	0	1	0	0	0	0	S11	001 001 001 001 001	X
S11	0	0	0	0	0	0	S11	001 001 001 001 001	X
S11	1	0	0	0	0	0	S0	001 001 100 001 001	X

c) State Diagram

There are numerous modes the system can go into, and they were all laid out in the transition tables section their operation can be seen and only the state diagram of normal circumstances will be shown. The state diagram for the system operating in normal circumstances is as follows:

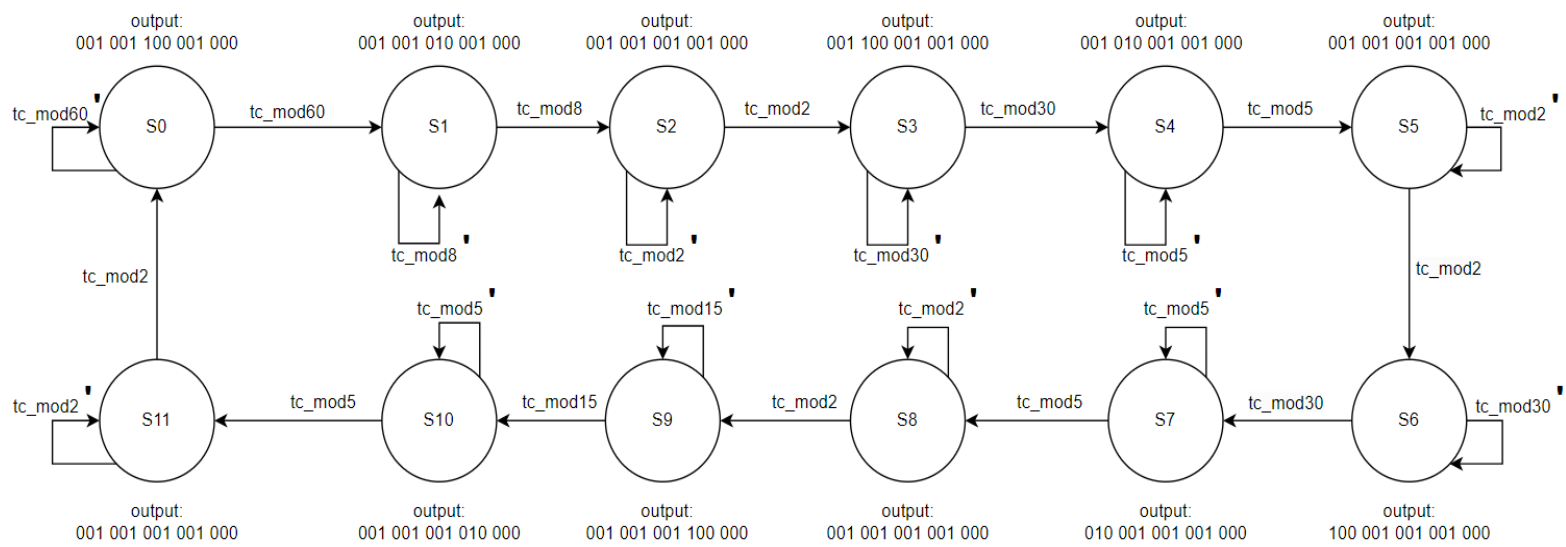


Figure 8: State Diagram

4. System Implementation

The system uses the FSM-D design. This system's inputs are sent to both the control and data path. The data and control path share communication between them through signals. The output of the data and control path go out of the system.

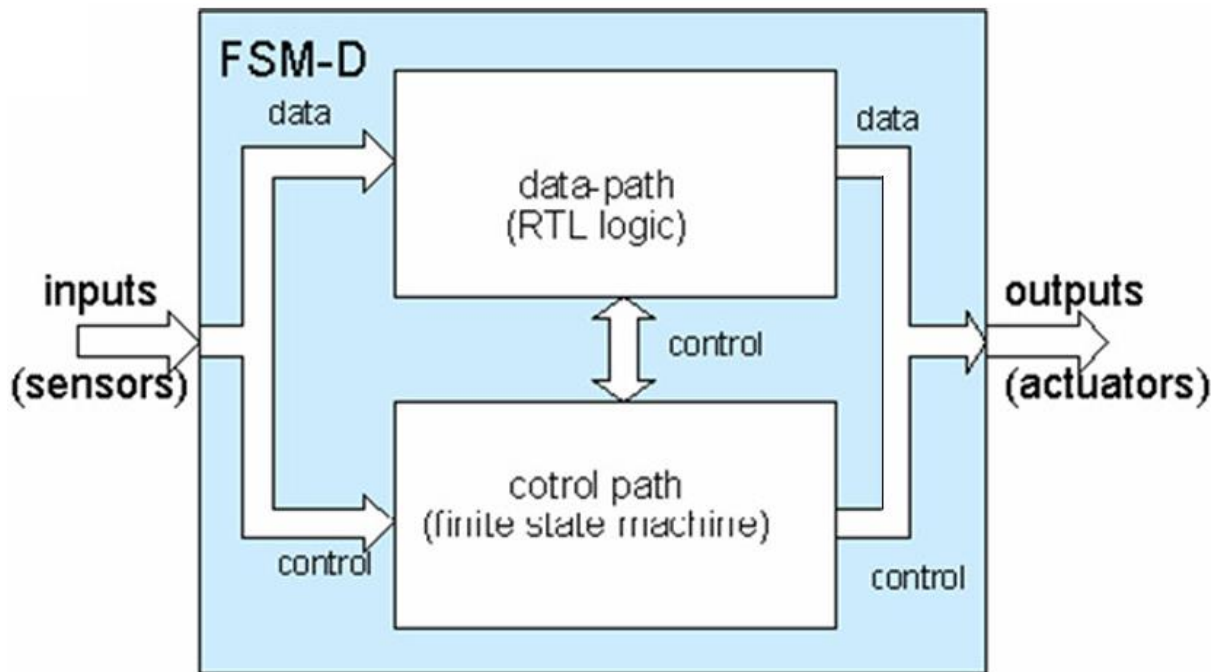


Figure 9: System Layout

In the case of our system the schematics are below. It shows the inputs on the left and outputs on the right. It is implemented thoroughly using VHDL and is documented in the appendix in section 7.1.

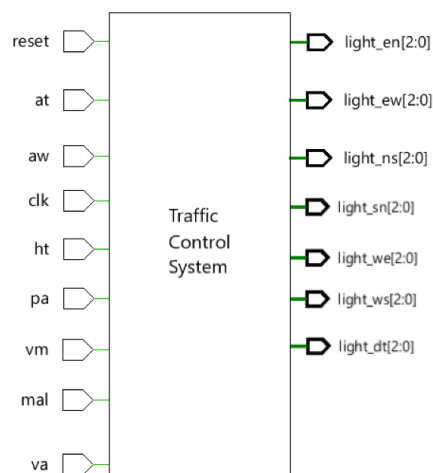


Figure 10: Traffic Control System

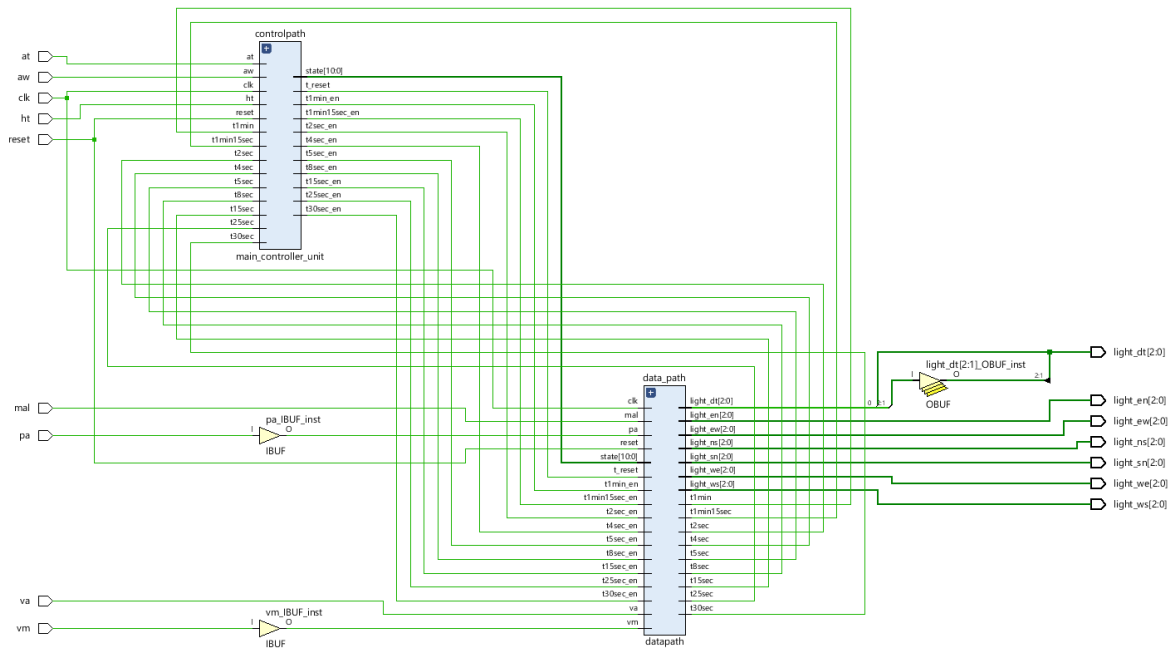


Figure 11: System Schematic

The system's data path was implemented using the specifications, it is documented in appendix 7.2 and its block diagrams are below.

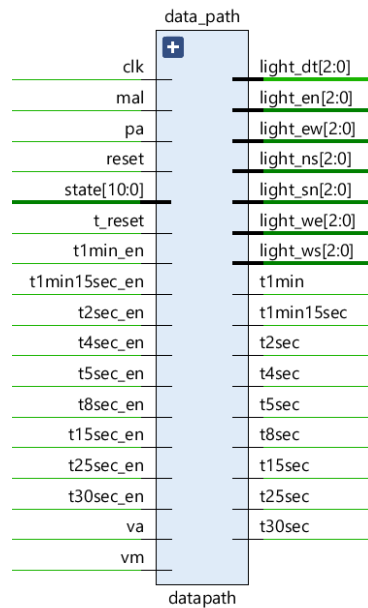
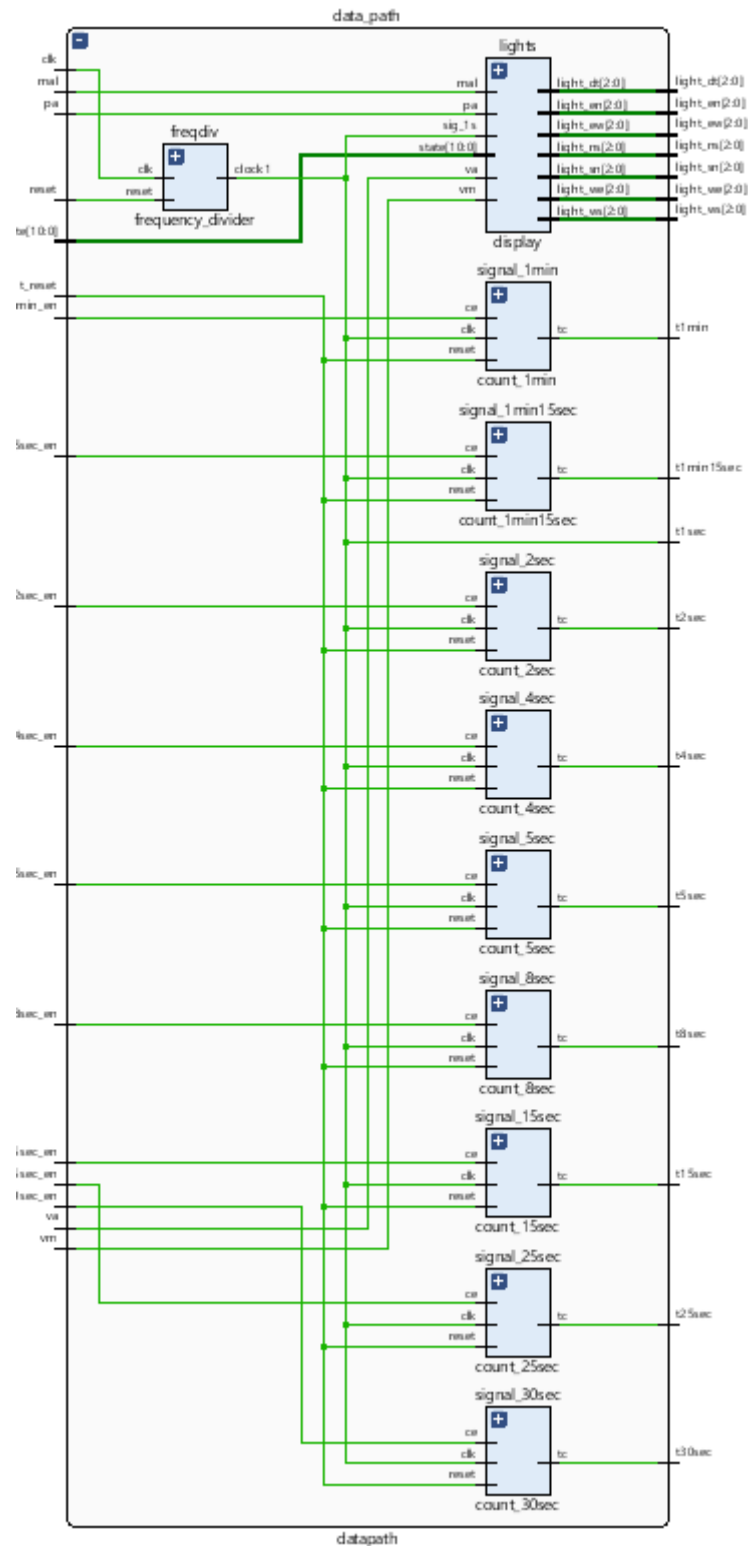


Figure 12: System Datapath



The system's Frequency Divider was implemented using the specifications, it is documented in appendix 7.3 and its block diagrams are below.

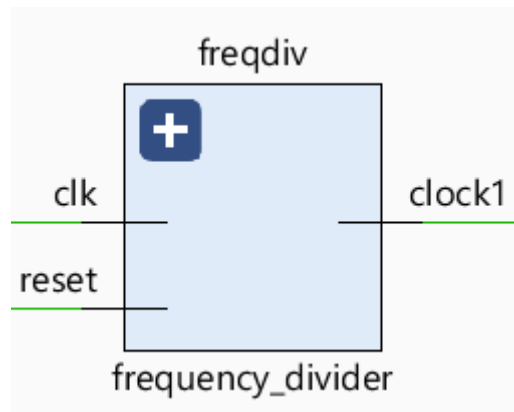
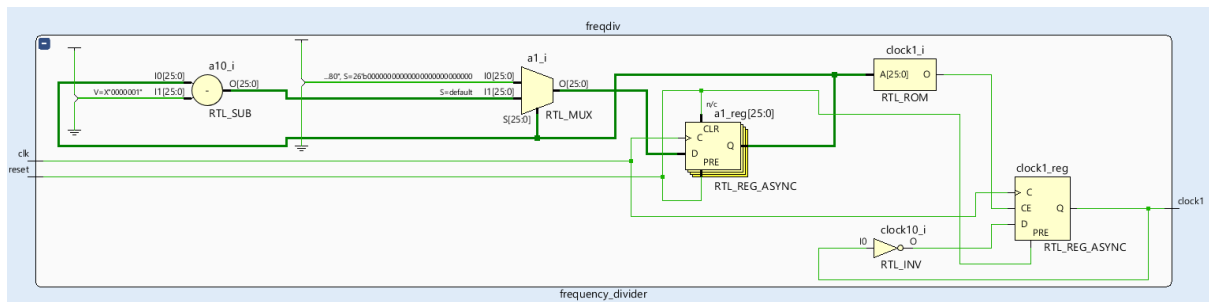


Figure 13: System Frequency Divider



The system's Counters was implemented using the specifications, it is documented in appendix 7.4 and its block diagrams are below.

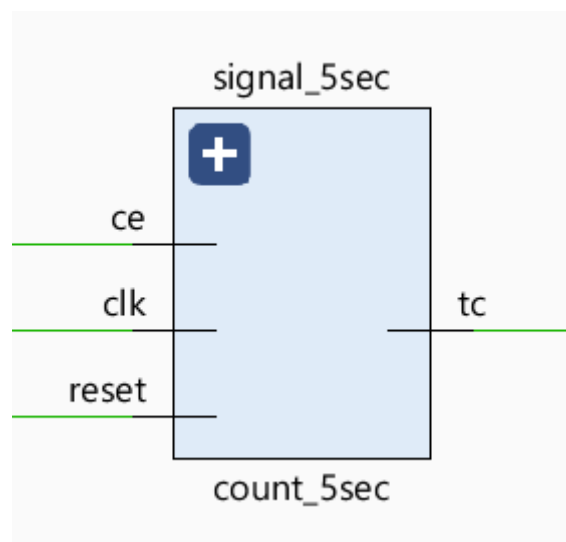
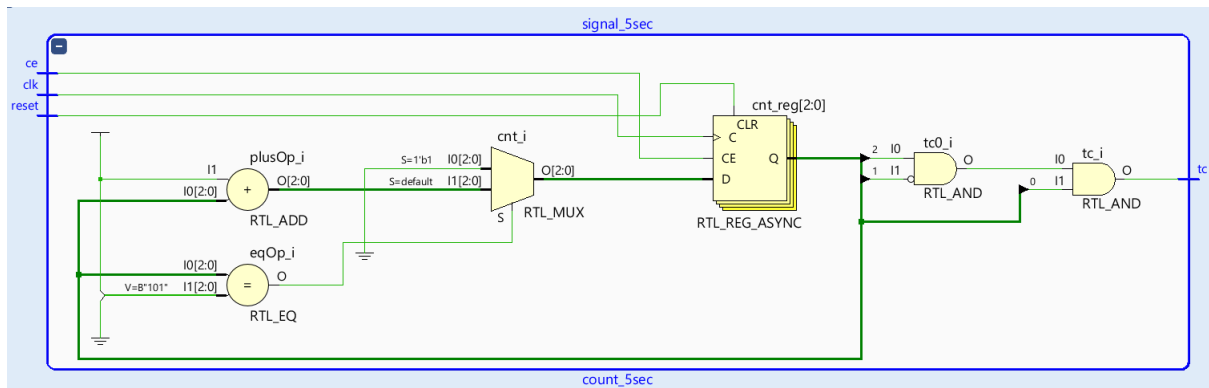


Figure 14: System Counter



The system's State logic process for the lights were implemented using the specifications, it is documented in appendix 7.5 and its block diagrams are below.

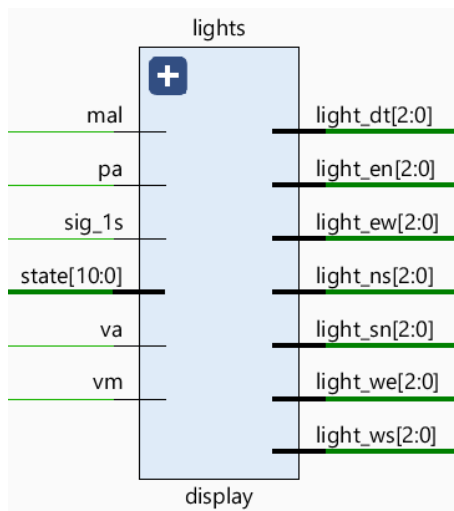
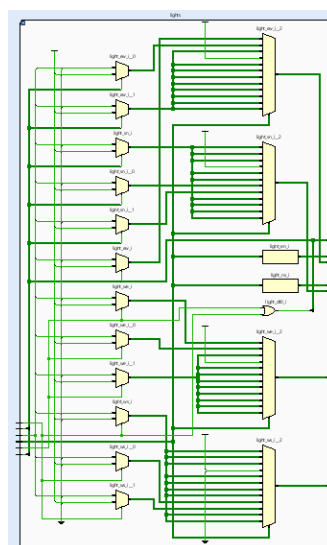


Figure 15: System State Logic Process



The system's Control Path was implemented using the specifications, it is documented in appendix 7.6 and its block diagrams are below.

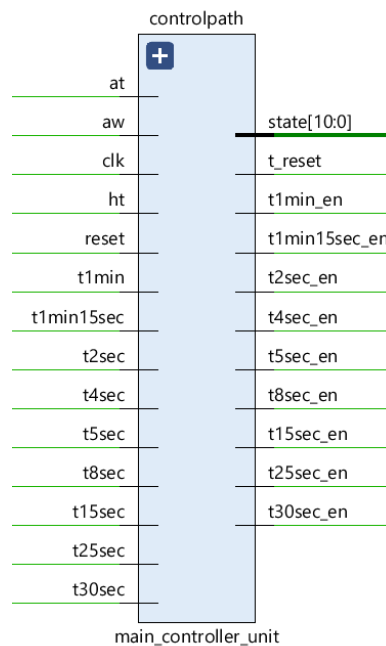
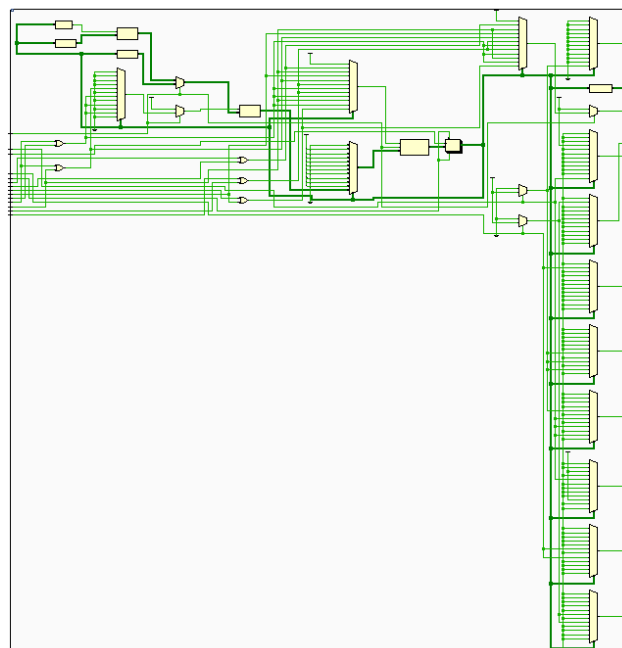


Figure 16: System Control Path



The system's basic components were briefly tests as well as the inputs do not necessarily look the same due to them being on the Basys board.

5. System Verification

Test ID	TC00
Scenario	Under Normal Conditions
Description	This tests to see if the system correctly functions with no scenario of failure. It should test to see if each traffic flow is serviced and for the required time.
Test Steps	<ol style="list-style-type: none">1. System Reset is driven high and returned low.2. The order of the lights illuminated and their timing are observed and recorded
Expected Results	<ol style="list-style-type: none">1. Light_ew and Light_we turn green for 60 seconds while all other lights are red.2. Light_ew and Light_we turn amber for 8 seconds while all other lights are red.3. All lights turn red for 2 seconds.4. Light_ns turns green for 30 seconds while all other lights are red.5. Light_ns turns amber for 5 seconds while all other lights are red.6. All lights turn red for 2 seconds.7. Light_sn turns green for 30 seconds while all other lights are red.8. Light_sn turns amber for 5 seconds while all other lights are red.9. All lights turn red for 2 seconds.

	<p>10. Light_en and Light_ws turn green for 15 seconds while all other lights are red.</p> <p>11. Light_en and Light_ws turn amber for 5 seconds while all other lights are red.</p> <p>12. All lights turn red for 2 seconds.</p> <p>13. Steps 1 to 12 are repeated again for two cycles.</p>
--	--

Test ID	TC01
Scenario	Heavy Traffic at A3 and A4 (Light_ew and Light_we)
Description	This tests to see how the system responds to an instance of heavy traffic flowing east to west and west to east. It should ensure that the added time allotted for this flow of traffic is given. The behaviour of the other traffic lights should be unaffected.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The input signal that indicates heavy traffic is held high. 3. The order of the lights illuminated and their timing are observed and recorded. 4. After one cycle, the input signal is returned and held low. 5. The order of the lights illuminated and their timing are observed and recorded.
Expected Results	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 75 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 8 seconds while all other lights are red. 3. All lights turn red for 2 seconds. 4. Light_ns turns green for 25 seconds while all other lights are red. 5. Light_ns turns amber for 5 seconds while all other lights are red. 6. All lights turn red for 2 seconds.

	<p>7. Light_sn turns green for 25 seconds while all other lights are red.</p> <p>8. Light_sn turns amber for 5 seconds while all other lights are red.</p> <p>9. All lights turn red for 2 seconds.</p> <p>10. Light_en and Light_ws turn green for 15 seconds while all other lights are red.</p> <p>11. Light_en and Light_ws turn amber for 5 seconds while all other lights are red.</p> <p>12. All lights turn red for 2 seconds.</p> <p>13. Heavy traffic input driven low.</p> <p>14. Light_ew and Light_we turn green for 60 seconds while all other lights are red.</p> <p>15. Light_ew and Light_we turn amber for 8 seconds while all other lights are red.</p> <p>16. Steps 3 to 12 are repeated.</p>
--	---

Test ID	TC02
Scenario	Traffic light malfunction at B3 (Light_ws)
Description	<p>This tests to see how the system handles the malfunction of a single traffic light. The light is expected to flash red if possible and an extra light is used to signal to traffic to make a detour. The behaviour of the other traffic lights should be unaffected. This test also will assure that when the malfunction is recovered the system seamlessly reintegrates the light and it functions as it required instantly.</p>
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The input signal that indicates traffic light malfunction is held high. 3. The order of the lights illuminated and their timing are observed and recorded. 4. While G4 is on green, the input signal is returned and held low. 5. The lights are observed to assure seamless transition when the malfunction occurs.
Expected Results	<ol style="list-style-type: none"> 1. Light_ws flashes red and the detour light (Light_dt) is illuminated while the following steps are carried out. 2. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 3. Light_ew and Light_we turn amber for 8 seconds while all other lights are red. 4. All lights turn red for 2 seconds.

	<p>5. Light_ns turns green for 30 seconds while all other lights are red.</p> <p>6. Light_ns turns amber for 5 seconds while all other lights are red.</p> <p>7. All lights turn red for 2 seconds.</p> <p>8. Light_sn turns green for 30 seconds while all other lights are red.</p> <p>9. Light_sn turns amber for 5 seconds while all other lights are red.</p> <p>10. All lights turn red for 2 seconds.</p> <p>11. When Light_en turn green, the malfunction signal is driven low.</p> <p>12. Light_ws turns green and both lights stay for 15 seconds while all other lights are red. The detour light is also turned off from this point.</p> <p>13. Light_en and Light_ws turn amber for 5 seconds while all other lights are red.</p> <p>14. All lights turn red for 2 seconds.</p> <p>15. Steps 2 to 14 are repeated keeping the signal low and therefore ignoring step 11.</p>
--	---

Test ID	TC03
Scenario	Vehicular malfunction at A1 (Light_sn)
Description	<p>This tests to see how the system operates with the detection of a vehicular malfunction. The light is expected to flash its required colours and an extra light is used to signal to traffic to make a detour.</p> <p>The behaviour of the other traffic lights should be unaffected. This test also will assure that the system can assume the required state the instant the malfunction is detected.</p>
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The system is allowed to run until Light_sn turns green. 3. After 5 seconds of Light_sn being green, the input signal that indicates a vehicular malfunction is driven and held high. 4. The order of the lights illuminated and their timing are observed and recorded. 5. The lights are observed to assure seamless transition when the malfunction occurs.
Expected Results	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 8 seconds while all other lights are red. 3. All lights turn red for 2 seconds. 4. Light_ns turns green for 30 seconds while all other lights are red. 5. Light_ns turns amber for 5 seconds while all other lights are red.

	<ol style="list-style-type: none"> 6. All lights turn red for 2 seconds. 7. Light_sn turns green for 5 seconds and the malfunction input is driven high. 8. Light_sn flashes green for the remaining 25 seconds and the detour light is illuminated. 9. Light_sn flashes amber for 5 seconds while all other lights are red. 10. Light_sn flashes red and the detour light is illuminated for the following steps. 11. All lights turn red for 2 seconds. 12. Light_ws and Light_en turn green for 15 seconds while all other lights are red. 13. Light_ws and Light_en turn amber for 5 seconds while all other lights are red. 14. All lights turn red for 2 seconds. 15. Steps 1 to 15 are repeated while Light_sn flashes, the detour light is illuminated and the input is kept high.
--	--

Test ID	TC04
Scenario	Vehicular accident at A3 (Light_we)
Description	This tests to see how the system operates with the detection of a vehicular accident. The light is expected to flash its required colours and an extra light is used to signal to traffic to make a detour. The behaviour of the other traffic lights should be unaffected. This test also will assure that the system can assume the required state the instant the malfunction is detected.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The system is allowed to run until Light_we turns amber. 3. After 3 seconds of Light_we being amber, the input signal that indicates a vehicular accident is driven and held high. 4. The order of the lights illuminated and their timing are observed and recorded. 5. The lights are observed to assure seamless transition when the malfunction occurs.
Expected Results	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 3 seconds while all other lights are red. 3. The input is driven high. 4. Light_we flashes amber for the remaining 5 seconds while the detour light is illuminated and Light_ew stays solid amber. 5. Light_we flashes red and the detour light is illuminated for the following steps.

	<p>6. All lights turn red for 2 seconds.</p> <p>7. Light_ns turns green for 30 seconds while all other lights are red.</p> <p>8. Light_ns turns amber for 5 seconds while all other lights are red.</p> <p>9. All lights turn red for 2 seconds.</p> <p>10. Light_sn turns green for 30 seconds while all other lights are red.</p> <p>11. Light_sn turns amber for 5 seconds while all other lights are red.</p> <p>12. All lights turn red for 2 seconds.</p> <p>13. Light_ws and Light_en turn green for 15 seconds while all other lights are red.</p> <p>14. Light_en and Light_ws turn amber for 5 seconds while all other lights are red.</p> <p>15. All lights turn red for 2 seconds.</p> <p>16. Steps 1 to 15 are repeated while Light_we flashes, the detour light is illuminated and the input is held high</p>
--	---

Test ID	TC05
Scenario	Adverse weather conditions
Description	This tests to see how the system responds to an instance of adverse weather conditions. It should ensure that the added time allotted for the amber lights and the transition from amber to red to green is allowed.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The input signal that indicates adverse is held high. 3. The order of the lights illuminated and their timing are observed and recorded. 4. After one cycle, the input signal is returned and held low. 5. The order of the lights illuminated and their timing are observed and recorded.
Expected Results	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 15 seconds while all other lights are red. 3. All lights turn red for 4 seconds. 4. Light_ns turns green for 30 seconds while all other lights are red. 5. Light_ns turns amber for 8 seconds while all other lights are red. 6. All lights turn red for 4 seconds.

	<p>7. Light_sn turns green for 30 seconds while all other lights are red.</p> <p>8. Light_sn turns amber for 8 seconds while all other lights are red.</p> <p>9. All lights turn red for 4 seconds.</p> <p>10. Light_en and Light_ws turn green for 15 seconds while all other lights are red.</p> <p>11. Light_en and Light_ws turn amber for 8 seconds while all other lights are red.</p> <p>12. All lights turn red for 4 seconds.</p> <p>13. Adverse weather input driven low.</p> <p>14. Light_ew and Light_we turn green for 60 seconds while all other lights are red.</p> <p>15. Light_ew and Light_we turn amber for 8 seconds while all other lights are red.</p> <p>16. All lights turn red for 2 seconds.</p> <p>17. Light_ns turns green for 30 seconds while all other lights are red.</p> <p>18. Light_ns turns amber for 5 seconds while all other lights are red.</p> <p>19. All lights turn red for 2 seconds.</p> <p>20. Light_sn turns green for 30 seconds while all other lights are red.</p> <p>21. Light_sn turns amber for 5 seconds while all other lights are red.</p>
--	--

	<p>22. All lights turn red for 2 seconds.</p> <p>23. Light_en and Light_ws turn green for 15 seconds while all other lights are red.</p> <p>24. Light_en and Light_ws turn amber for 5 seconds while all other lights are red.</p> <p>25. All lights turn red for 2 seconds.</p>
--	--

Test ID	TC06
Scenario	Ambulance throughfare at B2 (Light_ns)
Description	This tests to see how the system operates when an ambulance wishes to go from the north to westbound road. All lights are required to transition to red and the light for to allow the ambulance to pass is turned green. After the ambulance has passed, the system is expected to reassume the state it was in before the interruption.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The system is allowed to run until Light_sn turns amber, then the ambulance thoroughfare input is momentarily high. 3. The order of the lights illuminated and their timing are observed and recorded. 4. Steps 2 and 3 are repeated when light_ew turns green.
Expected Results	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 8 seconds while all other lights are red. 3. All lights turn red for 2 seconds. 4. Light_ns turns green for 30 seconds while all other lights are red. 5. Light_ns turns amber for 5 seconds while all other lights are red. 6. All lights turn red for 2 seconds. 7. Light_sn turns green for 30 seconds while all other lights are red.

	<p>8. As Light_sn turns amber, the input is driven high momentarily.</p> <p>9. Light_ns turns green for 15 seconds while all other lights are red.</p> <p>10. Light_ns turns amber for 4 seconds while all other lights are red.</p> <p>11. All lights turn red for 2 seconds.</p> <p>12. Light_sn turns amber for 8 seconds while all other lights are red.</p> <p>13. All lights turn red for 2 seconds.</p> <p>14. Light_ws and Light_en turn green for 15 seconds while all other lights are red.</p> <p>15. Light_ws and Light_en turn amber for 5 seconds while all other lights are red.</p> <p>16. All lights turn red for 2 seconds.</p> <p>17. As Light_ew and Light_we turn green, the input is driven high momentarily.</p> <p>18. Light_ns turns green for 15 seconds while all other lights are red.</p> <p>19. Light_ns turns amber for 4 seconds while all other lights are red.</p> <p>20. All lights turn red for 2 seconds.</p> <p>21. Light_ew and Light_we turn green for 60 seconds while all other lights are red.</p>
--	---

	<p>22. Light_ew and Light_we turn amber for 8 seconds while all other lights are red.</p> <p>23. All lights turn red for 2 seconds.</p>
--	---

Test ID	TC07
Scenario	Protest Activity at A4 (Light_ew)
Description	This tests to see how the system responds to a report of protest activity at light A4. The light is expected to flash red and an extra light is used to signal to traffic to make a detour. The behaviour of the other traffic lights should be unaffected. This test also will assure that when the report is received, that the light instantly reacts to assure that no accidents occur.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The system is allowed to run until Light_en and Light_ws turns green. 3. The input signal that indicates protest activity is driven and held high. 4. The order of the lights illuminated and their timing are observed and recorded. 5. When light_we turns amber, the input is driven low and held. 6. The order of the lights illuminated and their timing are observed and recorded.
Expected Results	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 8 seconds while all other lights are red. 3. All lights turn red for 2 seconds.

	<ol style="list-style-type: none"> 4. Light_ns turns green for 30 seconds while all other lights are red. 5. Light_ns turns amber for 5 seconds while all other lights are red. 6. All lights turn red for 2 seconds. 7. Light_sn turns green for 30 seconds while all other lights are red. 8. Light_sn turns amber for 5 seconds while all other lights are red. 9. All lights turn red for 2 seconds. 10. As Light_en and Light_ws turn green, the protest activity signal is driven low. 11. Light_ew flashes red and the detour light associated is illuminated for the following steps. 12. Light_ws and Light_en turn green for 15 seconds while all other lights are red. 13. Light_en and Light_ws turn amber for 5 seconds while all other lights are red. 14. All lights turn red for 2 seconds. 15. Light_we turns green for 60 seconds while Light_ew flashes red and all other lights are red . 16. When light_we turns amber, the input is driven low. 17. Light_ew and Light_we turn amber for 8 seconds and the detour light is turned off while all other lights are red.
--	---

	18. All lights turn red for 2 seconds.
--	--

Results

The system was subjected to the test cases outline using the simulator in Xilinx Vivado 19.2.

The testbench used is in the Appendix. The frequency divider time was changed for 1 second in real time to be 1 microsecond (us) in simulation time to the ease computational stress.

```

architecture Behavioral of test_sys is
component Traffic_control_system is
  Port (clk : in std_logic;
        reset : in std_logic;
        ht : in std_logic;
        mal : in std_logic;
        va : in std_logic;
        vm : in std_logic;
        pa : in std_logic;
        at : in std_logic;
        aw : in std_logic;
        light_ew : out std_logic_vector(2 downto 0);
        light_we : out std_logic_vector(2 downto 0);
        light_ns : out std_logic_vector(2 downto 0);
        light_sn : out std_logic_vector(2 downto 0);
        light_ws : out std_logic_vector(2 downto 0);
        light_en : out std_logic_vector(2 downto 0);
        light_dt : out std_logic_vector(2 downto 0));
end component;
---STEP #2-----

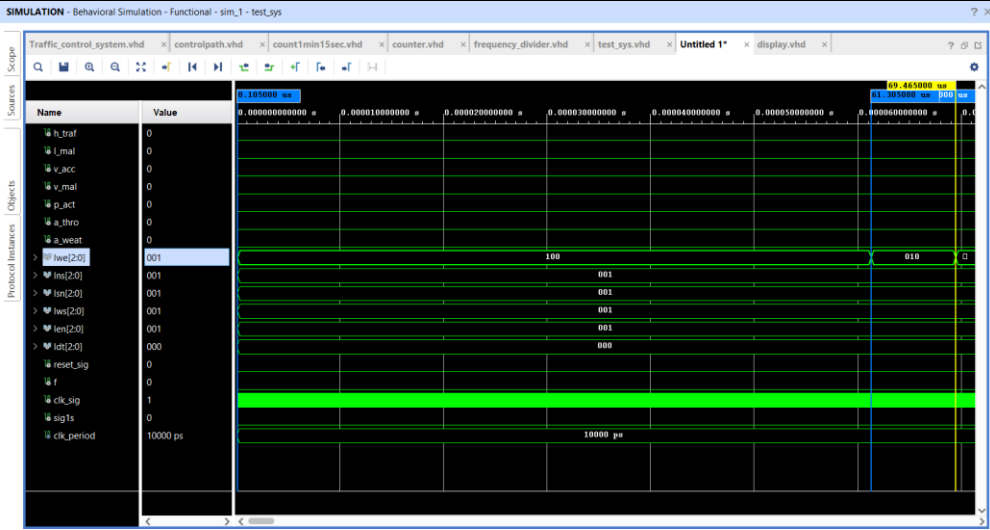
```

Figure 17: Test Bench

This is the declaration of the traffic light system.

System I/O signal	Testbench signal	Description
clk	clk_sig	Clock signal
reset	reset_sig	Reset signal
ht	h_traf	Heavy traffic signal
mal	l_mal	Light malfunction signal
va	v_acc	Vehicular accident signal
vm	v_mal	Vehicular malfunction signal
pa	p_act	Protest activity signal
at	a_thro	Ambulance throughfare signal
aw	a_weat	Adverse weather signal
light_ew	lew	Traffic light signal from east to west
light_we	lwe	Traffic light signal from west to west
light_ns	lns	Traffic light signal from north to south
light_sn	lsn	Traffic light signal from south to north
light_ws	lws	Traffic light signal from west to south
light_en	len	Traffic light signal from east north
light_dt	ldt	Traffic light signal for detours

Test ID	TC00
Scenario	Under Normal Conditions
Description	This tests to see if the system correctly functions with no scenario of failure. It should test to see if each traffic flow is serviced and for the required time.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The order of the lights illuminated and their timing are observed and recorded
Testbench process code	<pre> stim_proc: process begin --periodic clock waveform clk wait for clk_period*10; --insert stimulus here --hold reset state for 20ns reset_sig <= '1'; wait for 20 ps; --restore reset to 0 and wait indefinitely reset_sig <= '0'; wait for 20 ps; wait; end process; </pre>
Expected Results	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 8 seconds while all other lights are red. 3. All lights turn red for 2 seconds. 4. Light_ns turns green for 30 seconds while all other lights are red. 5. Light_ns turns amber for 5 seconds while all other lights are red. 6. All lights turn red for 2 seconds. 7. Light_sn turns green for 30 seconds while all other lights are red.

	<p>8. Light_sn turns amber for 5 seconds while all other lights are red.</p> <p>9. All lights turn red for 2 seconds.</p> <p>10. Light_en and Light_ws turn green for 15 seconds while all other lights are red.</p> <p>11. Light_en and Light_ws turn amber for 5 seconds while all other lights are red.</p> <p>12. All lights turn red for 2 seconds.</p> <p>13. Steps 1 to 12 are repeated again for two cycles.</p>
Actual Results	Same as expected results
Proof	 <p>Figure 18: Screenshot showing timing for light_we.</p>



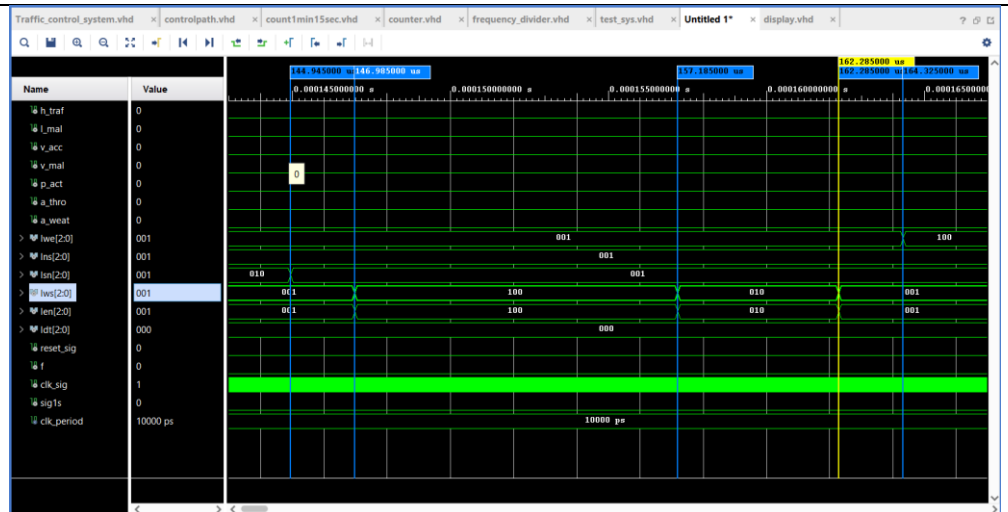


Figure 21: Screenshot showing timing for light_ws and light_en.

Test ID	TC01
Scenario	Heavy Traffic at A3 and A4 (Light_ew and Light_we)
Description	This tests to see how the system responds to an instance of heavy traffic flowing east to west and west to east. It should ensure that the added time allotted for this flow of traffic is given. The behaviour of the other traffic lights should be unaffected.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The input signal that indicates heavy traffic is held high. 3. The order of the lights illuminated and their timing are observed and recorded. 4. After one cycle, the input signal is returned and held low. 5. The order of the lights illuminated and their timing are observed and recorded.
Testbench process code	<pre>) h_traf<= '1';) --insert stimulus here ○ --hold reset state for 20ns ○ reset_sig <= '1'; wait for 20 ps; ○ --restore reset to 0 and wait indefinitely ○ reset_sig <= '0'; wait for 20 ps; ○ wait until (lew(1)='1'); ○ --wait for 3 us; h_traf<= '0'; ○ ○ </pre>
Expected Results	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 75 seconds while all other lights are red.

	<p>2. Light_ew and Light_we turn amber for 8 seconds while all other lights are red.</p> <p>3. All lights turn red for 2 seconds.</p> <p>4. Light_ns turns green for 25 seconds while all other lights are red.</p> <p>5. Light_ns turns amber for 5 seconds while all other lights are red.</p> <p>6. All lights turn red for 2 seconds.</p> <p>7. Light_sn turns green for 25 seconds while all other lights are red.</p> <p>8. Light_sn turns amber for 5 seconds while all other lights are red.</p> <p>9. All lights turn red for 2 seconds.</p> <p>10. Light_en and Light_ws turn green for 15 seconds while all other lights are red.</p> <p>11. Light_en and Light_ws turn amber for 5 seconds while all other lights are red.</p> <p>12. All lights turn red for 2 seconds.</p> <p>13. Heavy traffic input driven low.</p> <p>14. Light_ew and Light_we turn green for 60 seconds while all other lights are red.</p> <p>15. Light_ew and Light_we turn amber for 8 seconds while all other lights are red.</p> <p>16. Steps 3 to 12 are repeated.</p>
Actual results	<p>Same as expected results.</p> <p>There is an extra second for the light</p>

Proof



Test ID	TC02
Scenario	Traffic light malfunction at B3 (Light_ws)
Description	This tests to see how the system handles the malfunction of a single traffic light. The light is expected to flash red if possible and an extra light is used to signal to traffic to make a detour. The behaviour of the other traffic lights should be unaffected. This test also will assure that when the malfunction is recovered the system seamlessly reintegrates the light and it functions as it required instantly.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The input signal that indicates traffic light malfunction is held high. 3. The order of the lights illuminated and their timing are observed and recorded. 4. While G4 is on green, the input signal is returned and held low. 5. The lights are observed to assure seamless transition when the malfunction occurs.
Testbench process code	<pre> stim_proc: process begin --periodic clock waveform clk wait for clk_period*10; --insert stimulus here l_mal<= '1'; --hold reset state for 20ns reset_sig <= '1'; wait for 20 ps; --restore reset to 0 and wait indefinitely reset_sig <= '0'; wait for 20 ps; wait until (len(2)='1'); l_mal<= '0'; wait; end process; </pre>
Expected Results	<ol style="list-style-type: none"> 1. Light_ws flashes red and the detour light (Light_dt) is illuminated while the following steps are carried out.

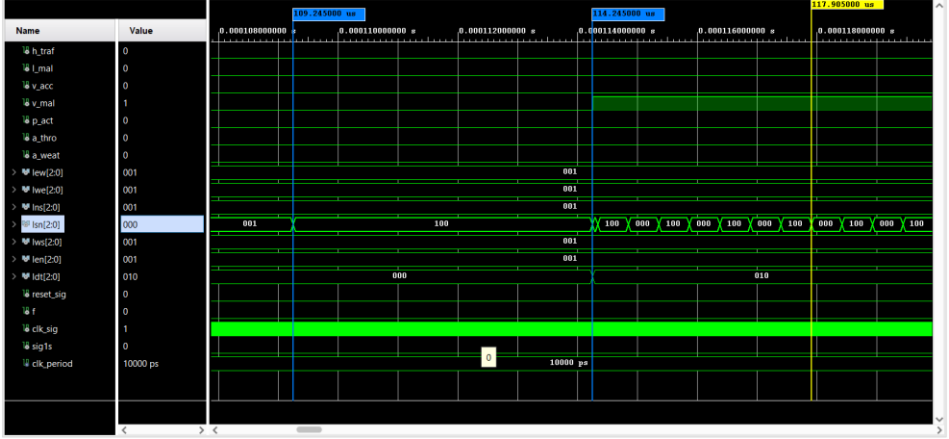
	<ol style="list-style-type: none"> 2. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 3. Light_ew and Light_we turn amber for 8 seconds while all other lights are red. 4. All lights turn red for 2 seconds. 5. Light_ns turns green for 30 seconds while all other lights are red. 6. Light_ns turns amber for 5 seconds while all other lights are red. 7. All lights turn red for 2 seconds. 8. Light_sn turns green for 30 seconds while all other lights are red. 9. Light_sn turns amber for 5 seconds while all other lights are red. 10. All lights turn red for 2 seconds. 11. When Light_en turn green, the malfunction signal is driven low. 12. Light_ws turns green and both lights stay for 15 seconds while all other lights are red. The detour light is also turned off from this point. 13. Light_en and Light_ws turn amber for 5 seconds while all other lights are red. 14. All lights turn red for 2 seconds. 15. Steps 2 to 14 are repeated keeping the signal low and therefore ignoring step 11.
Actual Results	Same as expected results

Figure 24: Screenshot showing light_ws switching at a 1Hz frequency with light malfunction signal on. The waveform shows light_ws as a green signal switching between high and low states at a regular interval, while light_mal is a constant high signal.

Figure 25: Screenshot showing `light_ws` stop switching as light malfunction signal goes low.

Test ID	TC03
Scenario	Vehicular malfunction at A1 (Light_sn)
Description	This tests to see how the system operates with the detection of a vehicular malfunction. The light is expected to flash its required colours and an extra light is used to signal to traffic to make a detour. The behaviour of the other traffic lights should be unaffected. This test also will assure that the system can assume the required state the instant the malfunction is detected.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The system is allowed to run until Light_sn turns green. 3. After 5 seconds of Light_sn being green, The input signal that indicates a vehicular malfunction is driven and held high. 4. The order of the lights illuminated and their timing are observed and recorded. 5. The lights are observed to assure seamless transition when the malfunction occurs.
Testbench process code	<pre> stim_proc: process begin --periodic clock waveform clk ○ wait for clk_period*10; --insert stimulus here --hold reset state for 20ns ○ reset_sig <= '1'; ○ wait for 20 ps; --restore reset to 0 and wait indefinitely ○ reset_sig <= '0'; ○ wait for 20 ps; ○ wait until (lsn(2)='1'); ○ wait for 5 us; ○ v_mal<= '1'; ○ wait; end process; </pre>

<p>Expected Results</p>	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 8 seconds while all other lights are red. 3. All lights turn red for 2 seconds. 4. Light_ns turns green for 30 seconds while all other lights are red. 5. Light_ns turns amber for 5 seconds while all other lights are red. 6. All lights turn red for 2 seconds. 7. Light_sn turns green for 5 seconds and the malfunction input is driven high. 8. Light_sn flashes green for the remaining 25 seconds and the detour light is illuminated. 9. Light_sn flashes amber for 5 seconds while all other lights are red. 10. Light_sn flashes red and the detour light is illuminated for the following steps. 11. All lights turn red for 2 seconds. 12. Light_ws and Light_en turn green for 15 seconds while all other lights are red. 13. Light_ws and Light_en turn amber for 5 seconds while all other lights are red. 14. All lights turn red for 2 seconds. 15. Steps 1 to 15 are repeated while Light_sn flashes, the detour light is illuminated and the input is kept high.
--------------------------------	---

<p>Actual Results</p>	<p>Same as expected results</p>
<p>Proof</p>	 <p>Figure 26: Screenshot showing light_sn flashing as soon as vehicular malfunction signal goes high.</p>

Test ID	TC04
Scenario	Vehicular accident at A3 (Light_we)
Description	This tests to see how the system operates with the detection of a vehicular accident. The light is expected to flash its required colours and an extra light is used to signal to traffic to make a detour. The behaviour of the other traffic lights should be unaffected. This test also will assure that the system can assume the required state the instant the malfunction is detected.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The system is allowed to run until Light_we turns amber. 3. After 3 seconds of Light_we being amber, The input signal that indicates a vehicular accident is driven and held high. 4. The order of the lights illuminated and their timing are observed and recorded. 5. The lights are observed to assure seamless transition when the malfunction occurs.
Testbench process code	<pre> stim_proc: process begin --periodic clock waveform clk ○ wait for clk_period*10; --insert stimulus here --hold reset state for 20ns ○ reset_sig <= '1'; ○ wait for 20 ps; --restore reset to 0 and wait indefinitely ○ reset_sig <= '0'; ○ wait for 20 ps; ○ wait until (lwe(2)='1'); ○ wait for 5 us; ○ v_acc<= '1'; ○ wait; end process; </pre>

<p>Expected Results</p>	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 3 seconds while all other lights are red. 3. The input is driven high. 4. Light_we flashes amber for the remaining 5 seconds while the detour light is illuminated and Light_ew stays solid amber. 5. Light_we flashes red and the detour light is illuminated for the following steps. 6. All lights turn red for 2 seconds. 7. Light_ns turns green for 30 seconds while all other lights are red. 8. Light_ns turns amber for 5 seconds while all other lights are red. 9. All lights turn red for 2 seconds. 10. Light_sn turns green for 30 seconds while all other lights are red. 11. Light_sn turns amber for 5 seconds while all other lights are red. 12. All lights turn red for 2 seconds. 13. Light_ws and Light_en turn green for 15 seconds while all other lights are red. 14. Light_en and Light_ws turn amber for 5 seconds while all other lights are red. 15. All lights turn red for 2 seconds. 16. Steps 1 to 15 are repeated while Light_we flashes, the detour light is illuminated and the input is held high
--------------------------------	--


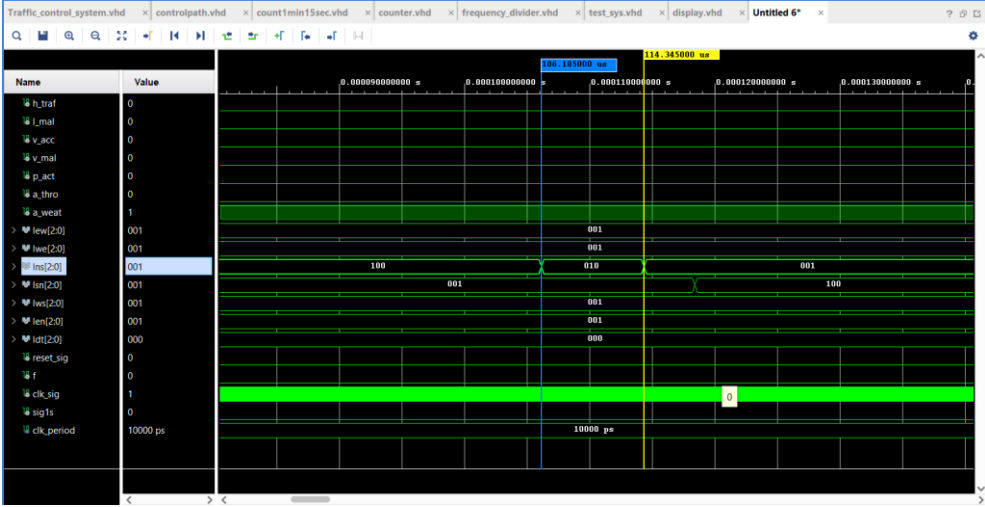
Actual results	Same as expected results.
Proof	

Figure 27: Screenshot showing light_we flashing when vehicular accident signal goes high

Test ID	TC05
Scenario	Adverse weather conditions
Description	This tests to see how the system responds to an instance of adverse weather conditions. It should ensure that the added time allotted for the amber lights and the transition from amber to red to green is allowed.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The input signal that indicates adverse is held high. 3. The order of the lights illuminated and their timing are observed and recorded. 4. After one cycle, the input signal is returned and held low. 5. The order of the lights illuminated and their timing are observed and recorded.
Testbench process code	<pre> stim_proc: process begin --periodic clock waveform clk ○ wait for clk_period*10; --insert stimulus here a_weat<='1'; --hold reset state for 20ns ○ reset_sig <= '1'; ○ wait for 20 ps; --restore reset to 0 and wait indefinitely ○ reset_sig <= '0'; ○ wait for 20 ps; ○ wait until (lwe(2)='1'); ○ --wait for 3 us; ○ a_weat<= '0'; </pre>

<p>Expected Results</p>	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 15 seconds while all other lights are red. 3. All lights turn red for 4 seconds. 4. Light_ns turns green for 30 seconds while all other lights are red. 5. Light_ns turns amber for 8 seconds while all other lights are red. 6. All lights turn red for 4 seconds. 7. Light_sn turns green for 30 seconds while all other lights are red. 8. Light_sn turns amber for 8 seconds while all other lights are red. 9. All lights turn red for 4 seconds. 10. Light_en and Light_ws turn green for 15 seconds while all other lights are red. 11. Light_en and Light_ws turn amber for 8 seconds while all other lights are red. 12. All lights turn red for 4 seconds. 13. Adverse weather input driven low. 14. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 15. Light_ew and Light_we turn amber for 8 seconds while all other lights are red. 16. All lights turn red for 2 seconds. 17. Light_ns turns green for 30 seconds while all other lights are red. 18. Light_ns turns amber for 5 seconds while all other lights are red. 19. All lights turn red for 2 seconds.
--------------------------------	---

	<p>20. Light_sn turns green for 30 seconds while all other lights are red.</p> <p>21. Light_sn turns amber for 5 seconds while all other lights are red.</p> <p>22. All lights turn red for 2 seconds.</p> <p>23. Light_en and Light_ws turn green for 15 seconds while all other lights are red.</p> <p>24. Light_en and Light_ws turn amber for 5 seconds while all other lights are red.</p> <p>25. All lights turn red for 2 seconds.</p>
Actual results	Same as expected
Proof	 <p>Figure 28: Screenshot showing amber time with adverse weather signal high</p>

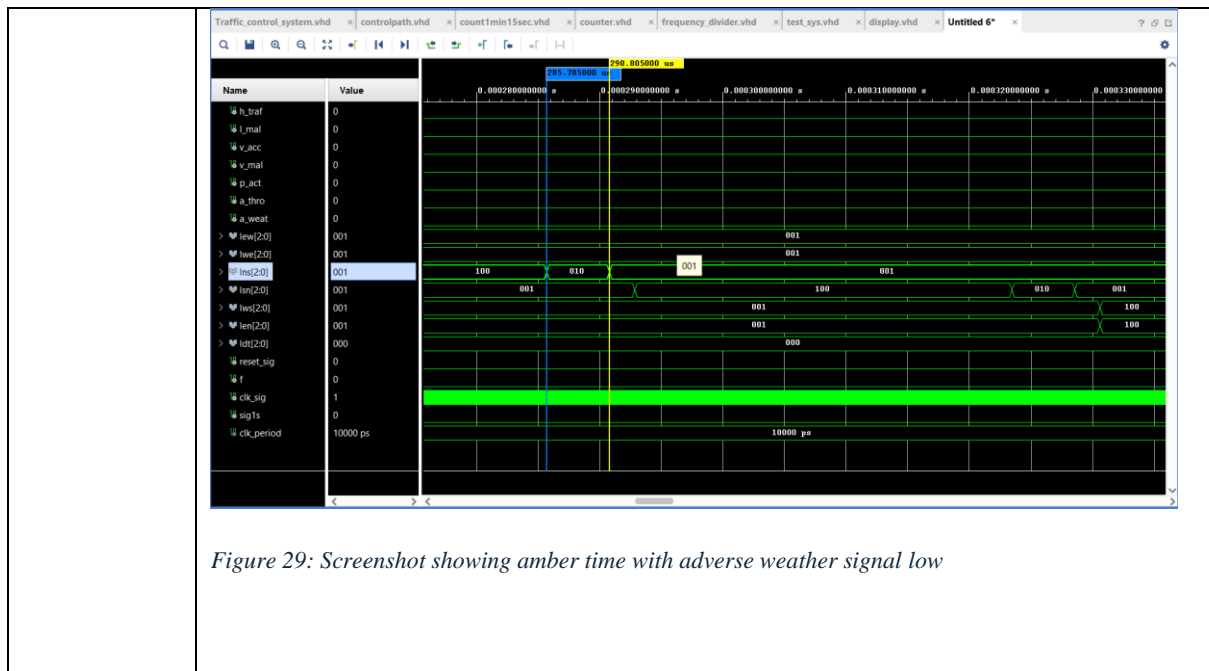


Figure 29: Screenshot showing amber time with adverse weather signal low

Test ID	TC06
Scenario	Ambulance throughfare at B2 (Light_ns)
Description	This tests to see how the system operates when an ambulance wishes to go from the north to westbound road. All lights are required to transition to red and the light for to allow the ambulance to pass is turned green. After the ambulance has passed, the system is expected to reassume the state it was in before the interruption.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The system is allowed to run until Light_sn turns amber, then the ambulance thoroughfare input is momentarily high. 3. The order of the lights illuminated and their timing are observed and recorded. 4. Steps 2 and 3 are repeated when light_ew turns green.

Testbench process code	<pre> --hold reset state for 20ns reset_sig <= '1'; wait for 20 ps; --restore reset to 0 and wait indefinitely reset_sig <= '0'; wait for 20 ps; wait until (lsn(1)='1'); --wait for 3 us; a_thro<= '1'; wait for 20 ps; a_thro<= '0'; wait until (lwe(2)='1'); --wait for 3 us; a_thro<= '1'; wait for 20 ps; a_thro<= '0'; </pre>
Expected Results	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 8 seconds while all other lights are red. 3. All lights turn red for 2 seconds. 4. Light_ns turns green for 30 seconds while all other lights are red. 5. Light_ns turns amber for 5 seconds while all other lights are red. 6. All lights turn red for 2 seconds. 7. Light_sn turns green for 30 seconds while all other lights are red. 8. As Light_sn turns amber, the input is driven high momentarily. 9. Light_ns turns green for 15 seconds while all other lights are red. 10. Light_ns turns amber for 4 seconds while all other lights are red. 11. All lights turn red for 2 seconds. 12. Light_sn turns amber for 8 seconds while all other lights are red.

	<p>13. All lights turn red for 2 seconds.</p> <p>14. Light_ws and Light_en turn green for 15 seconds while all other lights are red.</p> <p>15. Light_ws and Light_en turn amber for 5 seconds while all other lights are red.</p> <p>16. All lights turn red for 2 seconds.</p> <p>17. As Light_ew and Light_we turn green, the input is driven high momentarily.</p> <p>18. Light_ns turns green for 15 seconds while all other lights are red.</p> <p>19. Light_ns turns amber for 4 seconds while all other lights are red.</p> <p>20. All lights turn red for 2 seconds.</p> <p>21. Light_ew and Light_we turn green for 60 seconds while all other lights are red.</p> <p>22. Light_ew and Light_we turn amber for 8 seconds while all other lights are red.</p> <p>23. All lights turn red for 2 seconds.</p>
Actual results	Same as expected results

Proof



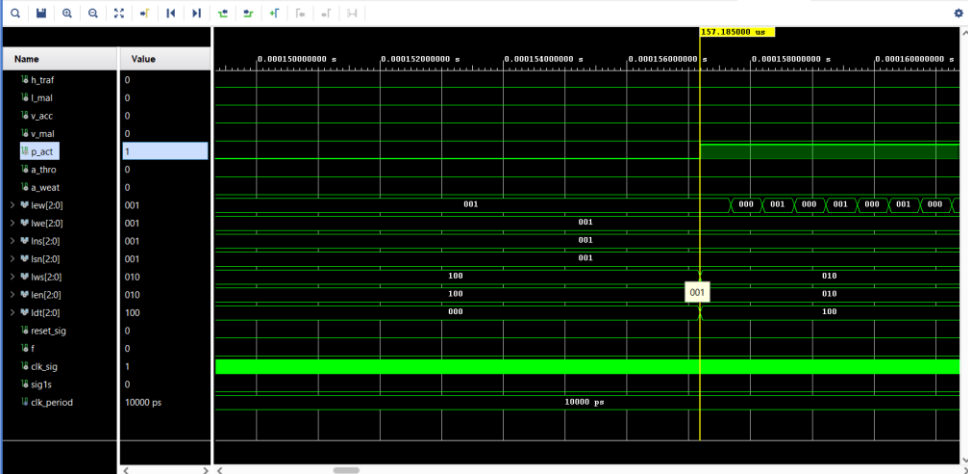
Figure 30: Screenshot showing light_ns turning green when ambulance throughfare signal goes high and return to same state after ambulance has passed.



Figure 31: Screenshot showing light_ns turning green when ambulance throughfare signal goes high and return to same state after ambulance has passed.

Test ID	TC07
Scenario	Protest Activity at A4 (Light_ew)
Description	This tests to see how the system responds to a report of protest activity at light A4. The light is expected to flash red and an extra light is used to signal to traffic to make a detour. The behaviour of the other traffic lights should be unaffected. This test also will assure that when the report is received, that the light instantly reacts to assure that no accidents occur.
Test Steps	<ol style="list-style-type: none"> 1. System Reset is driven high and returned low. 2. The system is allowed to run until Light_en and Light_ws turns green. 3. The input signal that indicates protest activity is driven and held high. 4. The order of the lights illuminated and their timing are observed and recorded. 5. When light_we turns amber, the input is driven low and held. 6. The order of the lights illuminated and their timing are observed and recorded.

Testbench process code	<pre> 105 -- Stimulus process ----- 106 ----- 107 stim_proc: process 108 begin 109 --periodic clock waveform clk 110 wait for clk_period*10; 111 112 --insert stimulus here 113 --hold reset state for 20ns 114 reset_sig <= '1'; 115 wait for 20 ps; 116 117 --restore reset to 0 and wait indefinitely 118 reset_sig <= '0'; 119 wait for 20 ps; 120 121 wait until (len(1)='1'); 122 --wait for 3 us; 123 p_act<= '1'; 124 125 wait until (lwe(1)='1'); 126 --wait for 3 us; 127 p_act<= '0'; 128 129 130 wait; 131 end process; 132 133 end Behavioral; 134 </pre>
Expected Results	<ol style="list-style-type: none"> 1. Light_ew and Light_we turn green for 60 seconds while all other lights are red. 2. Light_ew and Light_we turn amber for 8 seconds while all other lights are red. 3. All lights turn red for 2 seconds. 4. Light_ns turns green for 30 seconds while all other lights are red. 5. Light_ns turns amber for 5 seconds while all other lights are red. 6. All lights turn red for 2 seconds. 7. Light_sn turns green for 30 seconds while all other lights are red. 8. Light_sn turns amber for 5 seconds while all other lights are red. 9. All lights turn red for 2 seconds. 10. As Light_en and Light_ws turn green, the protest activity signal is driven low.

	<p>11. Light_ew flashes red and the detour light associated is illuminated for the following steps.</p> <p>12. Light_ws and Light_en turn green for 15 seconds while all other lights are red.</p> <p>13. Light_en and Light_ws turn amber for 5 seconds while all other lights are red.</p> <p>14. All lights turn red for 2 seconds.</p> <p>15. Light_we turns green for 60 seconds while Light_ew flashes red and all other lights are red .</p> <p>16. When light_we turns amber, the input is driven low.</p> <p>17. Light_ew and Light_we turn amber for 8 seconds and the detour light is turned off while all other lights are red.</p> <p>18. All lights turn red for 2 seconds.</p>
<p>Actual Results</p>	<p>Same as expected results.</p>
<p>Proof</p>	 <p>Figure 32: Screenshot showing light_ew flashing when protest activity signal goes high</p>

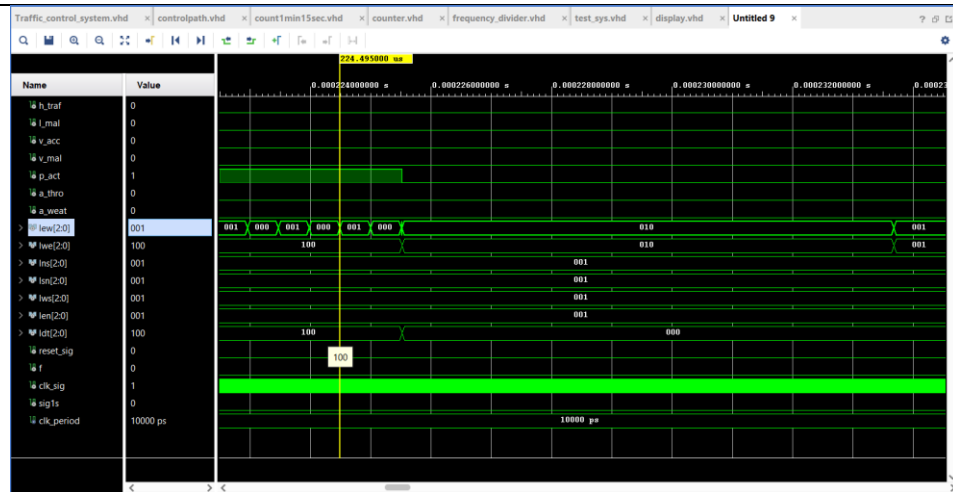


Figure 33: Screenshot showing light_ew stop flashing as protest activity signal goes low

6. On-board Testing

The onboard tests will just show that the system can traverse its modes properly. Hence the system will be started up allowed to run freely in its normal circumstances it will be allowed to complete all states in this mode, the mode will then be switched, and the system will be allowed to complete all states, this will be repeated until the system cycles through all modes. A picture of the system in each mode will be taken to give a brief example. The system will be timed to check that it matches up with the simulated system. The diagram of the testing system is below, it shows the lights that controls traffic in different directions as well as the lights that shows the detour lights.

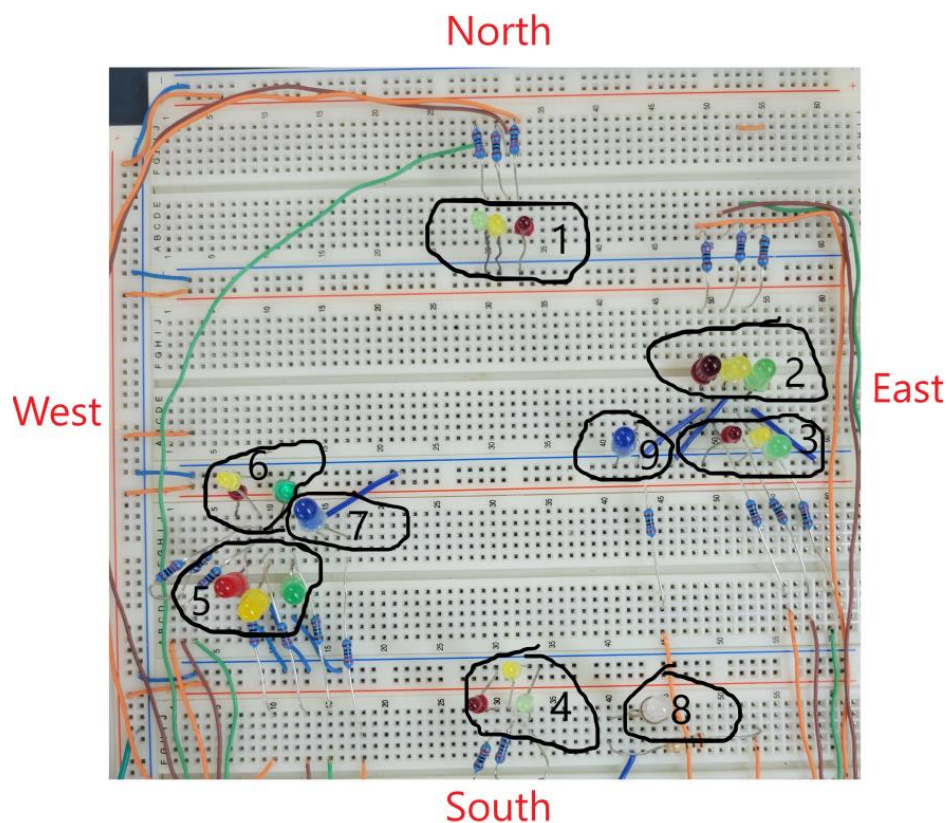


Figure 34: Breadboard Layout of Intersection

The labels in each of the circles correspond to the following:

1. This set of lights controls $A_2 B_2$.
2. This set of lights controls B_4 .
3. This set of lights controls A_4 .
4. This set of lights controls $A_1 B_1$.
5. This set of lights controls B_3 .
6. This set of lights controls A_3 .
7. This light controls b_1 (the west detour light).
8. This light controls b_2 (the south detour light).
9. This light controls b_3 (the east detour light).
10. The testing results show that the system operates as it should with regards to the simulation. The timing of the different counters was precise, when measured with a stopwatch while accounting for human reaction time. They are recorded in the table below:

Expected delay time	Recorded
2	2.12
4	4.09
5	5.05
8	8.19
15	15.12
25	25.11
30	30.23
60	60.07
75	75.15

Accounting for human error the times correlate with their expected values, hence they show the simulation was correct.

The following set of pictures show that the system operates in the different modes, via cycling through the modes as stated before:

- Normal circumstances

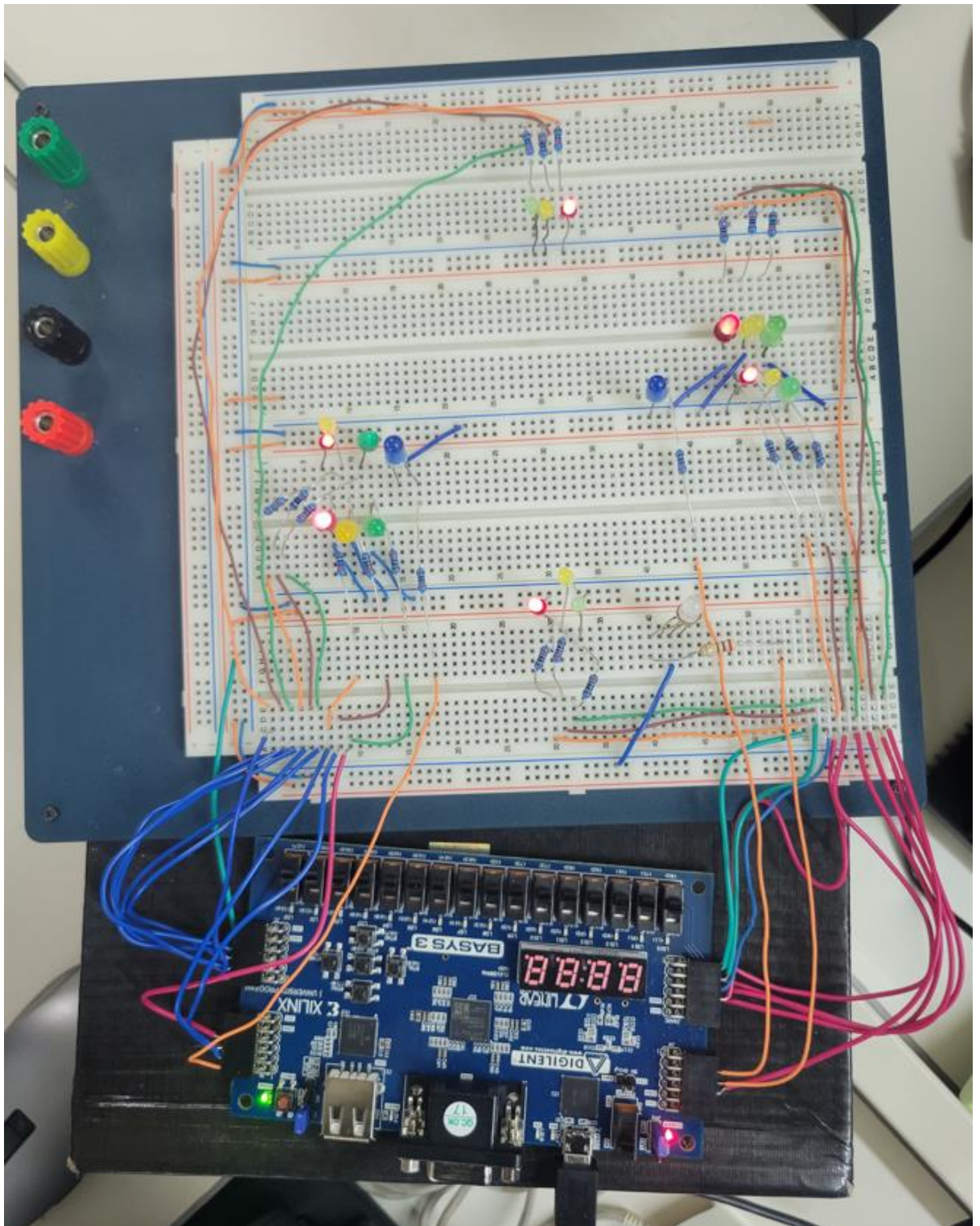


Figure 35: Proof of system functioning correctly case #1

- heavy traffic

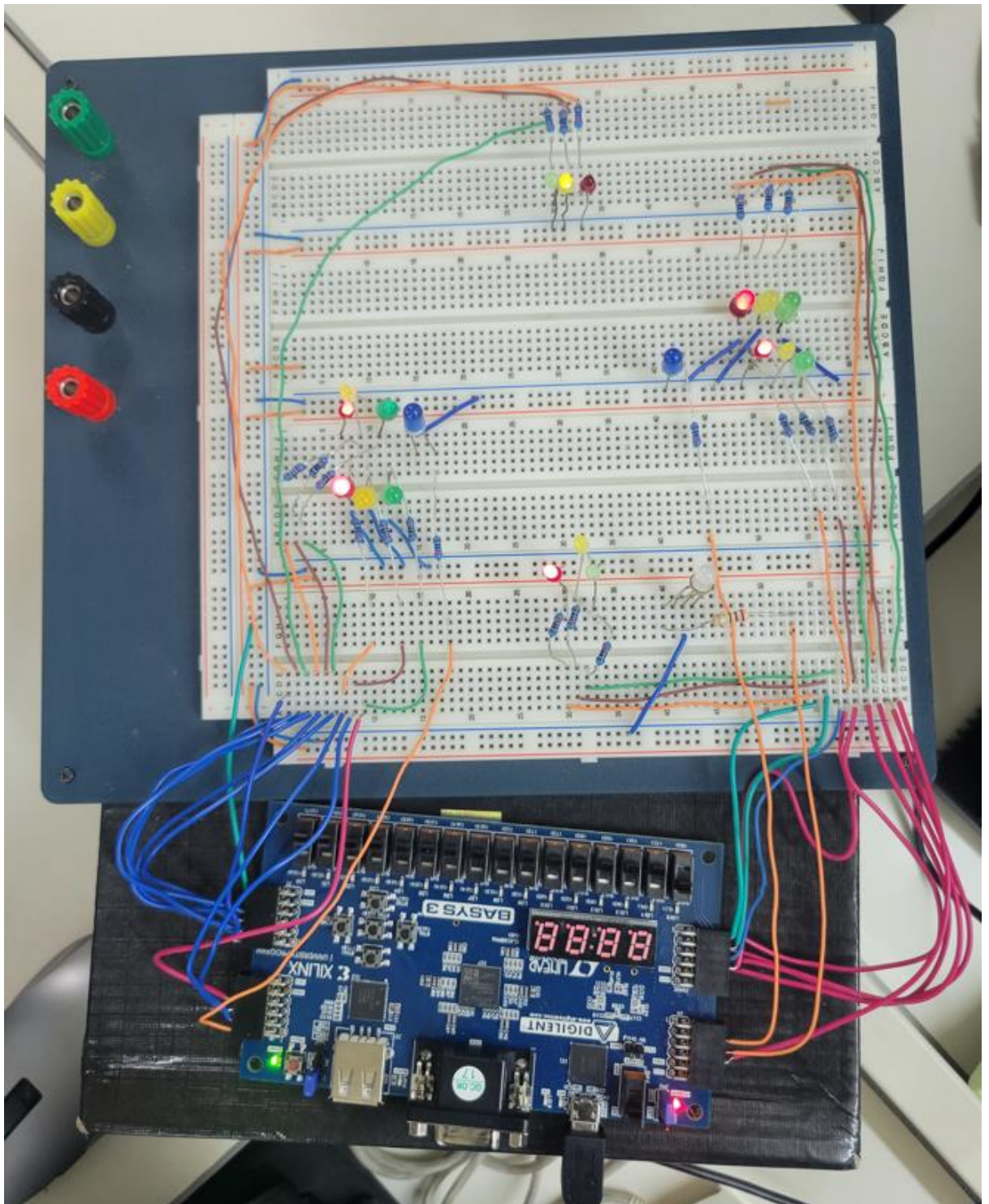


Figure 36: Proof of system functioning correctly case #2

- Traffic light malfunction

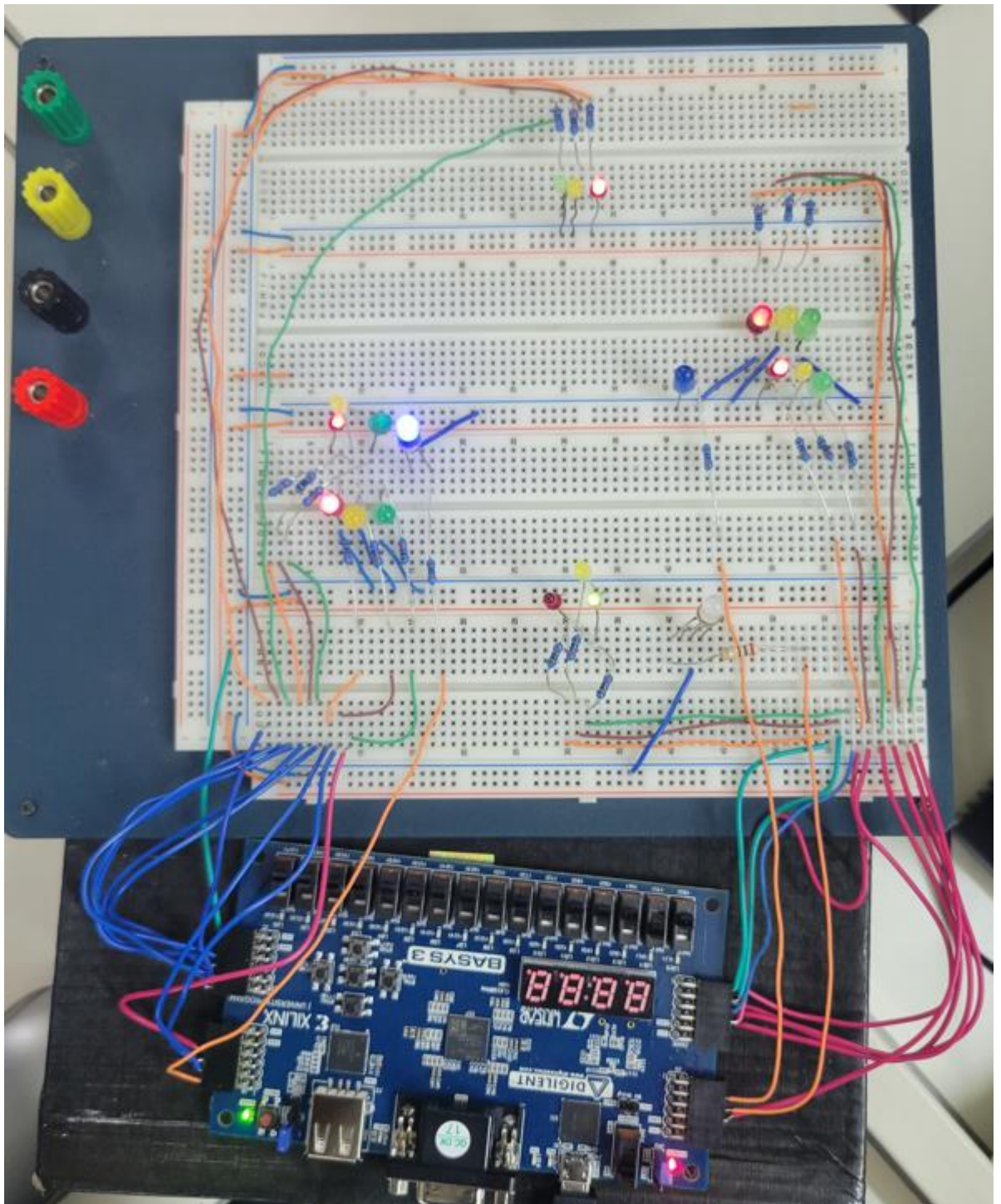


Figure 37: Proof of system functioning correctly case #3

- Vehicular malfunction

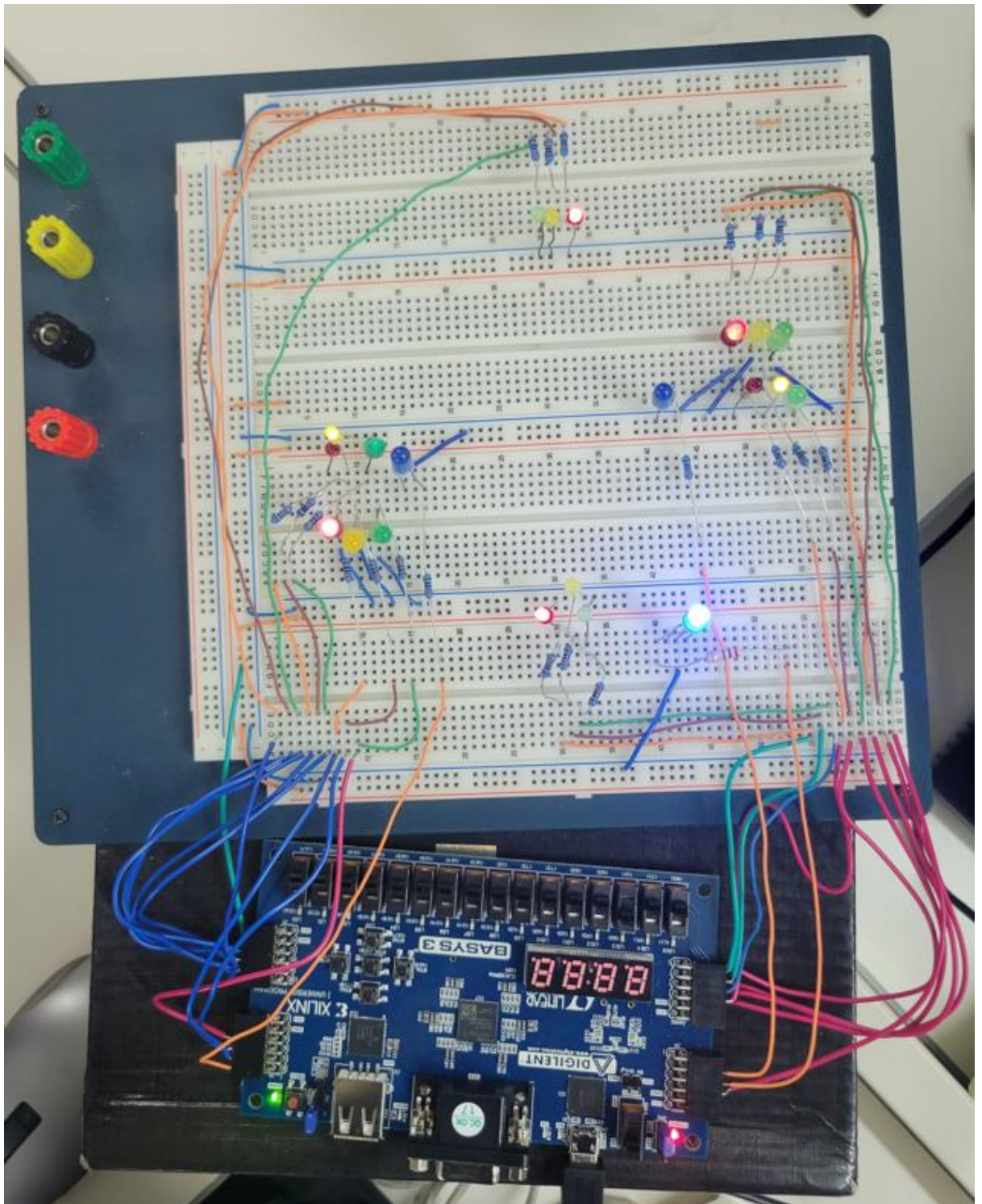


Figure 38: Proof of system functioning correctly case #4

- Vehicular accident

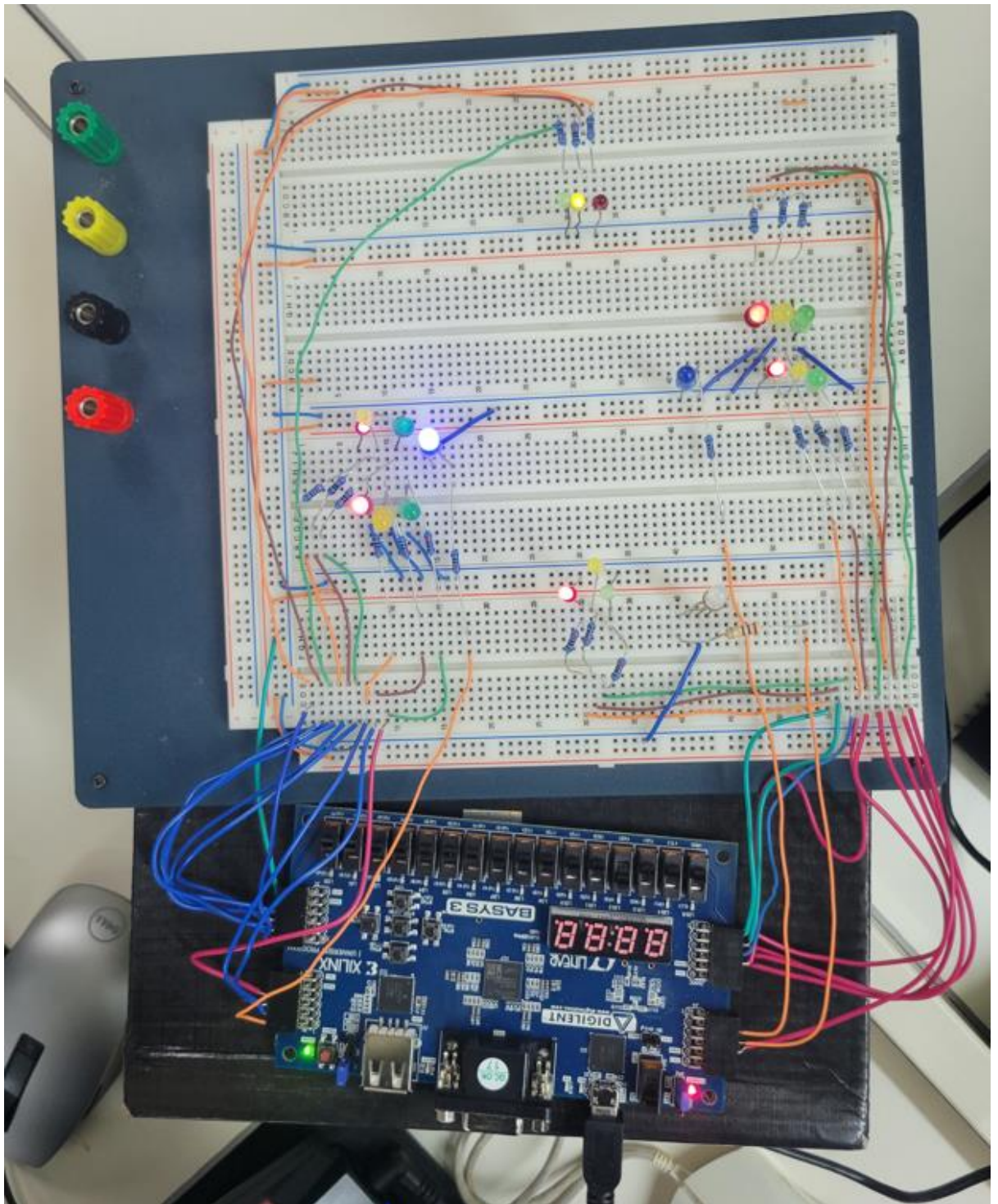


Figure 39: Proof of system functioning correctly case #5

- Adverse weather

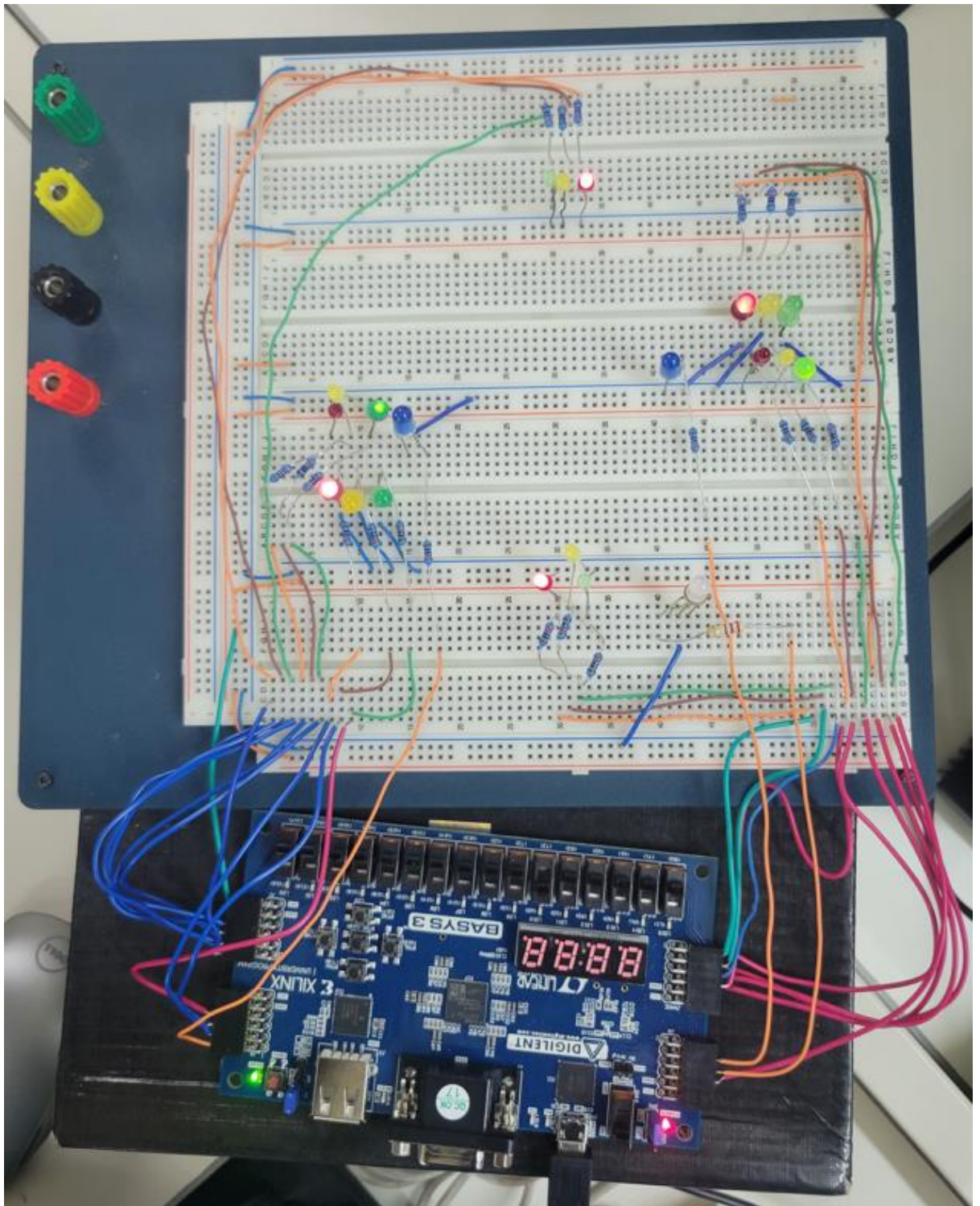


Figure 40: Proof of system functioning correctly case #6

- Ambulance throughfare

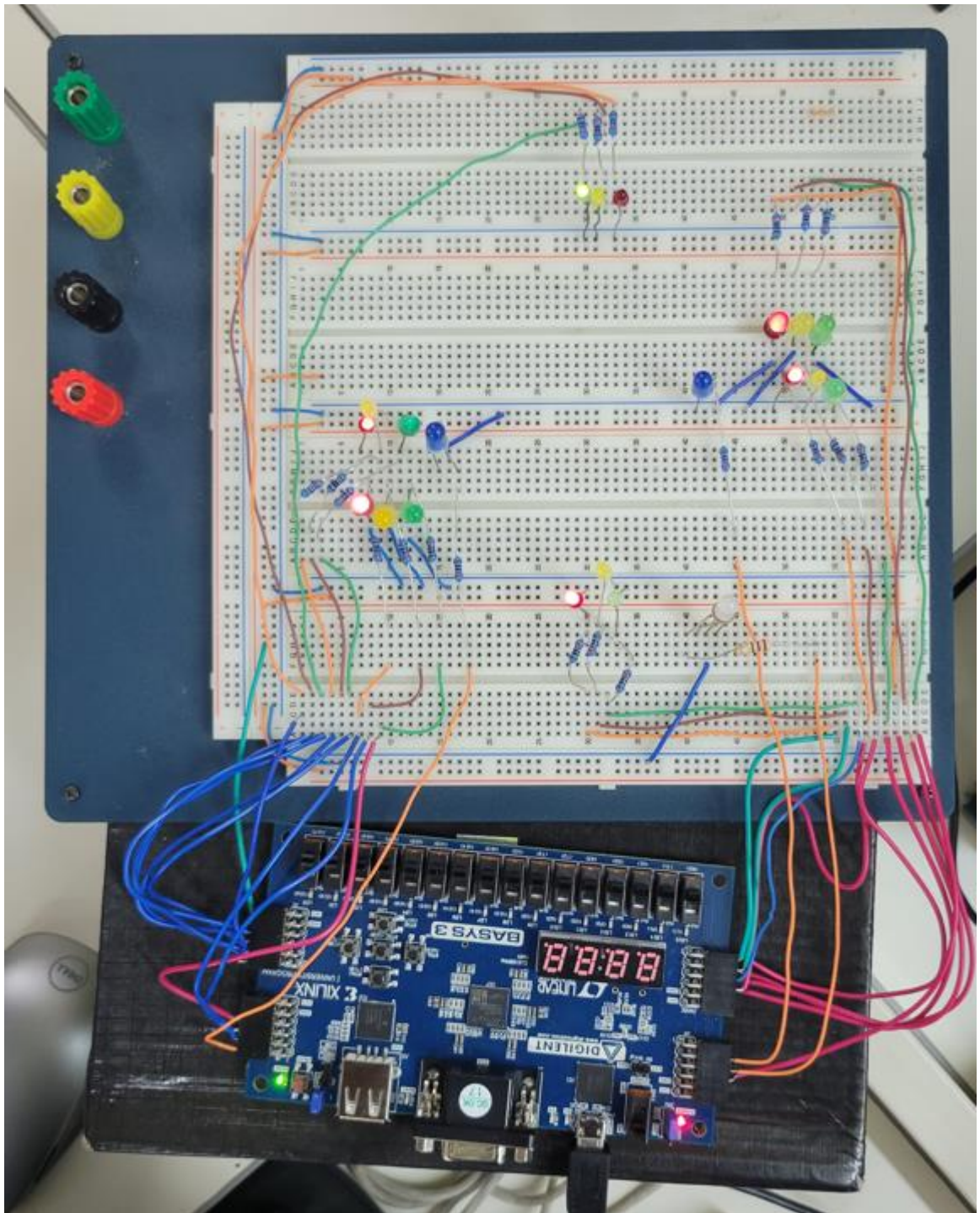


Figure 41: Proof of system functioning correctly case #7

- Protest Activity

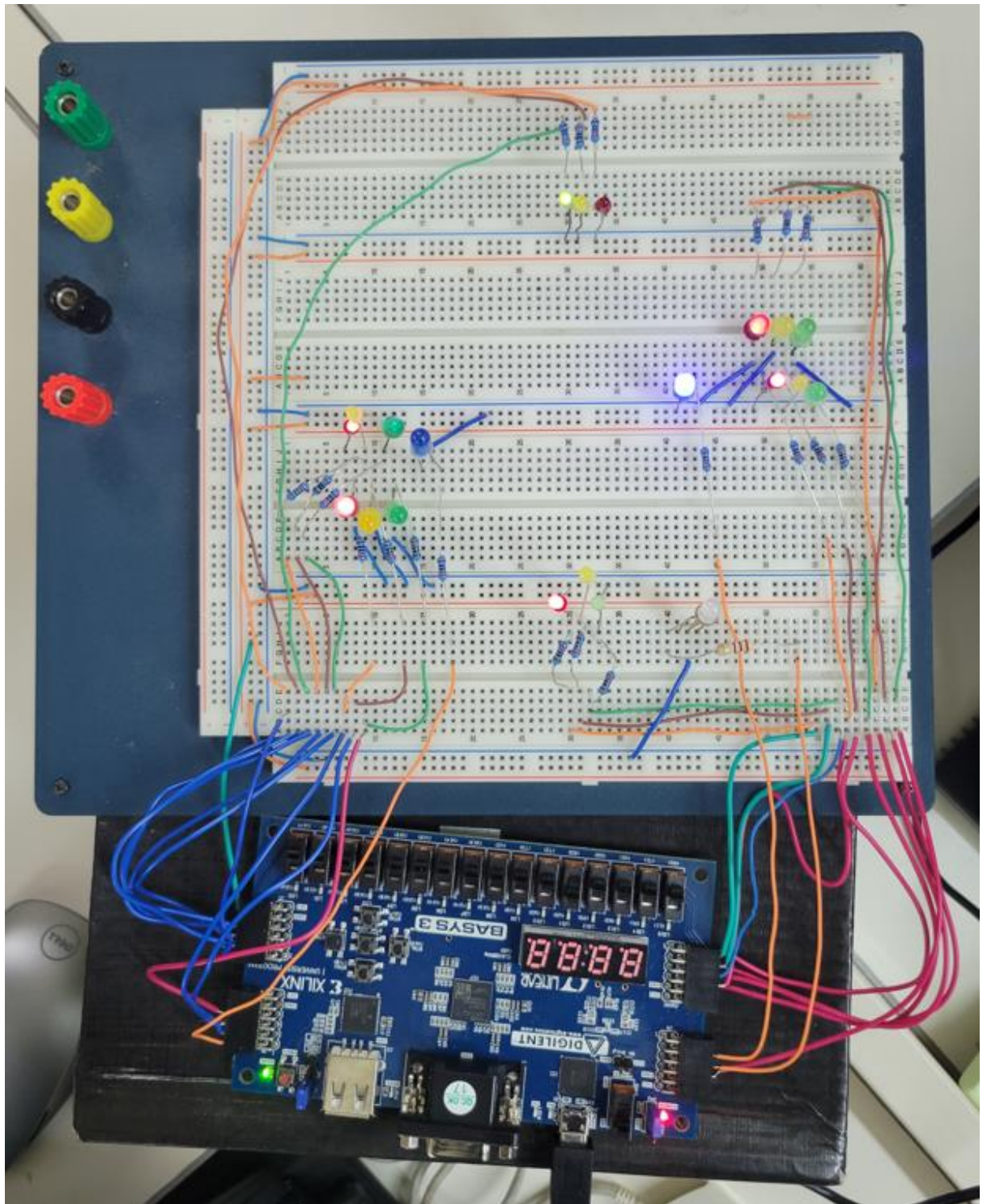


Figure 42: Proof of system functioning correctly case #8

These show the system cycles through its modes successfully, it also does not reset each time the mode changes, it just continues from where it was.

7. Appendix

7.1 Traffic control system interface.

entity Traffic_control_system is

```
Port (clk : in std_logic;
      reset : in std_logic;
      ht : in std_logic;
      mal : in std_logic;
      va : in std_logic;
      vm : in std_logic;
      pa : in std_logic;
      at : in std_logic;
      aw : in std_logic;
      light_ew : out std_logic_vector(2 downto 0);
      light_we : out std_logic_vector(2 downto 0);
      light_ns : out std_logic_vector(2 downto 0);
      light_sn : out std_logic_vector(2 downto 0);
      light_ws : out std_logic_vector(2 downto 0);
      light_en : out std_logic_vector(2 downto 0);
      light_dt : out std_logic_vector(2 downto 0));
```

end Traffic_control_system;

architecture Behavioral of Traffic_control_system is

component datapath is

```
Port (clk : in std_logic;
      reset : in std_logic;
      t2sec_en : in std_logic;
      t4sec_en : in std_logic;
      t5sec_en : in std_logic;
      t8sec_en : in std_logic;
      t15sec_en : in std_logic;
```

```

t25sec_en : in std_logic;
t30sec_en : in std_logic;
t1min_en : in std_logic;
t1min15sec_en : in std_logic;
t_reset : in std_logic;
mal : in std_logic;
vm : in std_logic;
va : in std_logic;
pa : in std_logic;
t1sec : out std_logic;
t2sec : out std_logic;
t4sec : out std_logic;
t5sec : out std_logic;
t8sec : out std_logic;
t15sec : out std_logic;
t25sec : out std_logic;
t30sec : out std_logic;
t1min : out std_logic;
t1min15sec : out std_logic;
state : in std_logic_vector(10 downto 0);
light_ew : out std_logic_vector(2 downto 0);
light_we : out std_logic_vector(2 downto 0);
light_ns : out std_logic_vector(2 downto 0);
light_sn : out std_logic_vector(2 downto 0);
light_ws : out std_logic_vector(2 downto 0);
light_en : out std_logic_vector(2 downto 0);
light_dt : out std_logic_vector(2 downto 0));
end component;

```

component main_controller_unit is

```

Port(clk : in std_logic;
      reset : in std_logic;

```

```

    at : in std_logic;
    aw : in std_logic;
    ht : in std_logic;
    t2sec : in std_logic;
    t4sec : in std_logic;
    t5sec : in std_logic;
    t8sec : in std_logic;
    t15sec : in std_logic;
    t25sec : in std_logic;
    t30sec : in std_logic;
    t1min : in std_logic;
    t1min15sec : in std_logic;
    t2sec_en : out std_logic;
    t4sec_en : out std_logic;
    t5sec_en : out std_logic;
    t8sec_en : out std_logic;
    t15sec_en : out std_logic;
    t25sec_en : out std_logic;
    t30sec_en : out std_logic;
    t1min_en : out std_logic;
    t1min15sec_en : out std_logic;
    t_reset : out std_logic;
    state : out std_logic_vector(10 downto 0));
end component;

signal sig_1s : std_logic;

signal
t1m15s,t1m,t30s,t25s,t15s,t8s,t5s,t4s,t2s,t1m15s_en,t1m_en,t30s_en,t25s_en,t15s_en,t8s_en,t
5s_en,t4s_en,t2s_en,t_reset : std_logic;

signal st : std_logic_vector(10 downto 0);

begin

data_path : datapath

```

```

port map(clk=>clk,
         reset=>reset,
         t1min15sec_en=>t1m15s_en,
         t1min_en=>t1m_en,
         t30sec_en=>t30s_en,
         t25sec_en=>t25s_en,
         t15sec_en=>t15s_en,
         t8sec_en=>t8s_en,
         t5sec_en=>t5s_en,
         t4sec_en=>t4s_en,
         t2sec_en=>t2s_en,
         t_reset=>t_reset,
         mal=>mal,
         vm=>vm,
         va=>va,
         pa=>pa,
         t1sec=>sig_1s,
         t2sec=>t2s,
         t4sec=>t4s,
         t5sec=>t5s,
         t8sec=>t8s,
         t15sec=>t15s,
         t25sec=>t25s,
         t30sec=>t30s,
         t1min=>t1m,
         t1min15sec=>t1m15s,
         state=>st,
         light_ew=>light_ew,
         light_we=>light_we,
         light_ns=>light_ns,
         light_sn=>light_sn,
         light_ws=>light_ws,

```

```
light_en=>light_en,
```

```
light_dt=>light_dt);
```

```
controlpath : main_controller_unit
```

```
port map(clk=>clk,
```

```
reset=>reset,
```

```
t2sec_en=>t2s_en,
```

```
t4sec_en=>t4s_en,
```

```
t5sec_en=>t5s_en,
```

```
t8sec_en=>t8s_en,
```

```
t15sec_en=>t15s_en,
```

```
t25sec_en=>t25s_en,
```

```
t30sec_en=>t30s_en,
```

```
t1min_en=>t1m_en,
```

```
t1min15sec_en=>t1m15s_en,
```

```
t_reset=>t_reset,
```

```
at=>at,
```

```
aw=>aw,
```

```
ht=>ht,
```

```
t2sec=>t2s,
```

```
t4sec=>t4s,
```

```
t5sec=>t5s,
```

```
t8sec=>t8s,
```

```
t15sec=>t15s,
```

```
t25sec=>t25s,
```

```
t30sec=>t30s,
```

```
t1min=>t1m,
```

```
t1min15sec=>t1m15s,
```

```
state=>st);
```

```
end Behavioral;
```

7.2 Datapath

entity datapath is

```
Port(clk : in std_logic;
      reset : in std_logic;
      t2sec_en : in std_logic;
      t4sec_en : in std_logic;
      t5sec_en : in std_logic;
      t8sec_en : in std_logic;
      t15sec_en : in std_logic;
      t25sec_en : in std_logic;
      t30sec_en : in std_logic;
      t1min_en : in std_logic;
      t1min15sec_en : in std_logic;
      t_reset : in std_logic;
      state : in std_logic_vector(10 downto 0);
      mal : in std_logic;
      vm : in std_logic;
      va : in std_logic;
      pa : in std_logic;
      t1sec : out std_logic;
      t2sec : out std_logic;
      t4sec : out std_logic;
      t5sec : out std_logic;
      t8sec : out std_logic;
      t15sec : out std_logic;
      t25sec : out std_logic;
      t30sec : out std_logic;
      t1min : out std_logic;
      t1min15sec : out std_logic;
      light_ew : out std_logic_vector(2 downto 0);
      light_we : out std_logic_vector(2 downto 0);
```

```

    light_ns : out std_logic_vector(2 downto 0);
    light_sn : out std_logic_vector(2 downto 0);
    light_ws : out std_logic_vector(2 downto 0);
    light_en : out std_logic_vector(2 downto 0);
    light_dt : out std_logic_vector(2 downto 0));
end datapath;

```

architecture Behavioral of datapath is

component frequency_divider is

```

    Port(clk  : in std_logic;
          reset : in std_logic;
          clock1 : inout std_logic);

```

end component;

component count_1min15sec is

```

    Port(clk : in std_logic;
          reset : in std_logic;
          ce : in std_logic;
          tc : out std_logic);

```

end component;

component count_1min is

```

    Port(clk : in std_logic;
          reset : in std_logic;
          ce : in std_logic;
          tc : out std_logic);

```

end component;

component count_30sec is

```

    Port(clk : in std_logic;
          reset : in std_logic;

```



```
        ce : in std_logic;
        tc : out std_logic);
end component;
```

```
component count_25sec is
    Port(clk : in std_logic;
        reset : in std_logic;
            ce : in std_logic;
            tc : out std_logic);
end component;
```

```
component count_15sec is
    Port(clk : in std_logic;
        reset : in std_logic;
            ce : in std_logic;
            tc : out std_logic);
end component;
```

```
component count_8sec is
    Port(clk : in std_logic;
        reset : in std_logic;
            ce : in std_logic;
            tc : out std_logic);
end component;
```

```
component count_5sec is
    Port(clk : in std_logic;
        reset : in std_logic;
            ce : in std_logic;
            tc : out std_logic);
end component;
```

component count_4sec is

```
Port(clk : in std_logic;
      reset : in std_logic;
      ce : in std_logic;
      tc : out std_logic);
```

end component;

component count_2sec is

```
Port(clk : in std_logic;
      reset : in std_logic;
      ce : in std_logic;
      tc : out std_logic);
```

end component;

component display is

```
Port ( state : in STD_LOGIC_VECTOR (10 downto 0);
      mal: in std_logic;
      vm : in std_logic;
      va : in std_logic;
      pa : in std_logic;
      sig_1s : in std_logic;
      light_ew : out std_logic_vector(2 downto 0);
      light_we : out std_logic_vector(2 downto 0);
      light_ns : out std_logic_vector(2 downto 0);
      light_sn : out std_logic_vector(2 downto 0);
      light_ws : out std_logic_vector(2 downto 0);
      light_en : out std_logic_vector(2 downto 0);
      light_dt : out std_logic_vector(2 downto 0));
```

end component;

signal sig_1s : std_logic;

begin

freqdiv: frequency_divider

```
port map(clk=>clk,  
         reset=>reset,  
         clock1=>sig_1s);
```

signal_1min15sec : count_1min15sec

```
port map(clk=>sig_1s,  
         reset=>t_reset,  
         ce=>t1min15sec_en,  
         tc=>t1min15sec);
```

signal_1min : count_1min

```
port map(clk=>sig_1s,  
         reset=>t_reset,  
         ce=>t1min_en,  
         tc=>t1min);
```

signal_30sec : count_30sec

```
port map(clk=>sig_1s,  
         reset=>t_reset,  
         ce=>t30sec_en,  
         tc=>t30sec);
```

signal_25sec : count_25sec

```
port map(clk=>sig_1s,  
         reset=>t_reset,  
         ce=>t25sec_en,  
         tc=>t25sec);
```

signal_15sec : count_15sec

```
port map(clk=>sig_1s,  
         reset=>t_reset,
```

```
ce=>t15sec_en,  
tc=>t15sec);
```

```
signal_8sec : count_8sec  
port map(clk=>sig_1s,  
reset=>t_reset,  
ce=>t8sec_en,  
tc=>t8sec);
```

```
signal_5sec : count_5sec  
port map(clk=>sig_1s,  
reset=>t_reset,  
ce=>t5sec_en,  
tc=>t5sec);
```

```
signal_4sec : count_4sec  
port map(clk=>sig_1s,  
reset=>t_reset,  
ce=>t4sec_en,  
tc=>t4sec);
```

```
signal_2sec : count_2sec  
port map(clk=>sig_1s,  
reset=>t_reset,  
ce=>t2sec_en,  
tc=>t2sec);
```

lights: display

```
port map(state=>state,  
mal=>mal,  
vm=>vm,  
va=>va,
```

```

        pa=>pa,
        sig_1s=>sig_1s,
        light_ew=>light_ew,
        light_we=>light_we,
        light_ns=>light_ns,
        light_sn=>light_sn,
        light_ws=>light_ws,
        light_en=>light_en,
        light_dt=>light_dt);

t1sec<=sig_1s;
end Behavioral;

```

7.3 Frequency Divider

entity frequency_divider is

```

    Port(clk   : in std_logic;
          reset : in std_logic;
          clock1 : inout std_logic);

```

end frequency_divider;

architecture Behavioral of frequency_divider is

begin

--5MHz clock signal

```

    clk_1Hz : process(clk, reset) -- clock division by 10

```

```

        variable a1: integer range 0 to 50000000;    --10/2 = 5

```

```

        begin

```

```

            if(reset = '1') then

```

```

                a1:= 50000000;

```

```

                clock1 <= '1';

```

```

            elsif(clk'event and clk = '1') then

```

```

        if(a1 = 0)then
            a1:= 500000000;
            clock1 <= not (clock1);
        else
            a1 := a1 - 1;
        end if;
    end if;
end process;
end Behavioral;

```

7.4 Counters

There are multiple counters in the system only one will be given since they are all variations of each other.

entity frequency_divider is

```

    Port(clk  : in std_logic;
          reset : in std_logic;
          clock1 : inout std_logic);

```

end frequency_divider;

architecture Behavioral of frequency_divider is

begin

--5MHz clock signal

```

    clk_1Hz : process(clk, reset) -- clock division by 10

```

```

        variable a1: integer range 0 to 500000000;    --10/2 = 5

```

```

        begin

```

```

            if(reset ='1') then

```

```

                a1:= 500000000;

```

```

                clock1 <= '1';

```

```

            elsif(clk'event and clk = '1') then

```

```

                if(a1 = 0)then

```

```

        a1:= 50000000;
        clock1 <= not (clock1);
    else
        a1 := a1 - 1;
        end if;
    end if;
end process;
end Behavioral;

```

7.5 State logic process (lights)

entity display is

```

    Port ( state : in STD_LOGIC_VECTOR (10 downto 0);
          mal : in std_logic;
          sig_1s : in std_logic;
          vm : in std_logic;
          va : in std_logic;
          pa : in std_logic;
          light_ew : out std_logic_vector(2 downto 0);
          light_we : out std_logic_vector(2 downto 0);
          light_ns : out std_logic_vector(2 downto 0);
          light_sn : out std_logic_vector(2 downto 0);
          light_ws : out std_logic_vector(2 downto 0);
          light_en : out std_logic_vector(2 downto 0);
          light_dt : out std_logic_vector(2 downto 0));
end display;

```

architecture Behavioral of display is

```

begin
    disp : process(state,mal,va,vm,pa,sig_1s)
    begin

```

case state is

```
when "000000000001" =>
  if (pa='1')then
    light_ew <= "00" & sig_1s;
  else
    light_ew <= "100";
  end if;
  if (va='1')then
    light_we <= sig_1s & "00";
  else
    light_we <= "100";
  end if;
  light_ns <= "001";
  if (vm='1')then
    light_sn <= "00" & sig_1s;
  else
    light_sn <= "001";
  end if;
  if (mal='1')then
    light_ws <= "00" & sig_1s;
  else
    light_ws <= "001";
  end if;
  light_en <= "001";
when "000000000010" =>
  if (pa='1')then
    light_ew <= "00" & sig_1s;
  else
    light_ew <= "010";
  end if;
  if (va='1')then
    light_we <= "0" & sig_1s & "0";
```



```

else
    light_we <= "010";
end if;
light_ns <= "001";
if (vm='1')then
    light_sn <= "00" & sig_1s;
else
    light_sn <= "001";
end if;
if (mal='1')then
    light_ws <= "00" & sig_1s;
else
    light_ws <= "001";
end if;
light_en <= "001";
when "00000000100" =>
    if (pa='1')then
        light_ew <= "00" & sig_1s;
    else
        light_ew <= "001";
    end if;
    if (va='1')then
        light_we <= "00" & sig_1s;
    else
        light_we <= "001";
    end if;
    light_ns <= "100";
    if (vm='1')then
        light_sn <= "00" & sig_1s;
    else
        light_sn <= "001";
    end if;

```

```

if (mal='1')then
    light_ws <= "00" & sig_1s;
else
    light_ws <= "001";
end if;
light_en <= "001";
when "00000001000" =>
    if (pa='1')then
        light_ew <= "00" & sig_1s;
    else
        light_ew <= "001";
    end if;
    if (va='1')then
        light_we <= "00" & sig_1s;
    else
        light_we <= "001";
    end if;
    light_ns <= "010";
    if (vm='1')then
        light_sn <= "00" & sig_1s;
    else
        light_sn <= "001";
    end if;
    if (mal='1')then
        light_ws <= "00" & sig_1s;
    else
        light_ws <= "001";
    end if;
    light_en <= "001";
when "00000010000" =>
    if (pa='1')then
        light_ew <= "00" & sig_1s;

```

```

else
    light_ew <= "001";
end if;
if (va='1')then
    light_we <= "00" & sig_1s;
else
    light_we <= "001";
end if;
light_ns <= "001";
if (vm='1')then
    light_sn <= sig_1s & "00" ;
else
    light_sn <= "100";
end if;
if (mal='1')then
    light_ws <= "00" & sig_1s;
else
    light_ws <= "001";
end if;
light_en <= "001";
when "00000100000" =>
    if (pa='1')then
        light_ew <= "00" & sig_1s;
    else
        light_ew <= "001";
    end if;
    if (va='1')then
        light_we <= "00" & sig_1s;
    else
        light_we <= "001";
    end if;
    light_ns <= "001";

```

```

if (vm='1')then
  light_sn <= "0" & sig_1s & "0";
else
  light_sn <= "010";
end if;
if (mal='1')then
  light_ws <= "00" & sig_1s;
else
  light_ws <= "001";
end if;
light_en <= "001";
when "00001000000" =>
  if (pa='1')then
    light_ew <= "00" & sig_1s;
  else
    light_ew <= "001";
  end if;
  if (va='1')then
    light_we <= "00" & sig_1s;
  else
    light_we <= "001";
  end if;
  light_ns <= "001";
  if (vm='1')then
    light_sn <= "00" & sig_1s;
  else
    light_sn <= "001";
  end if;
  if (mal='1')then
    light_ws <= "00" & sig_1s;
  else
    light_ws <= "100";
  end if;
end when;

```

```

    end if;
    light_en <= "100";
when "00010000000" =>
    if (pa='1')then
        light_ew <= "00" & sig_1s;
    else
        light_ew <= "001";
    end if;
    if (va='1')then
        light_we <= "00" & sig_1s;
    else
        light_we <= "001";
    end if;
    light_ns <= "001";
    if (vm='1')then
        light_sn <= "00" & sig_1s;
    else
        light_sn <= "001";
    end if;
    if (mal='1')then
        light_ws <= "00" & sig_1s;
    else
        light_ws <= "010";
    end if;
    light_en <= "010";
when "00100000000" =>
    if (pa='1')then
        light_ew <= "00" & sig_1s;
    else
        light_ew <= "001";
    end if;
    if (va='1')then

```

```

    light_we <= "00" & sig_1s;
else
    light_we <= "001";
end if;
light_ns <= "100";
if (vm='1')then
    light_sn <= "00" & sig_1s;
else
    light_sn <= "001";
end if;
if (mal='1')then
    light_ws <= "00" & sig_1s;
else
    light_ws <= "001";
end if;
light_en <= "001";
when "01000000000" =>
    if (pa='1')then
        light_ew <= "00" & sig_1s;
    else
        light_ew <= "001";
    end if;
    if (va='1')then
        light_we <= "00" & sig_1s;
    else
        light_we <= "001";
    end if;
    light_ns <= "010";
    if (vm='1')then
        light_sn <= "00" & sig_1s;
    else
        light_sn <= "001";

```

```

end if;
if (mal='1')then
    light_ws <= "00" & sig_1s;
else
    light_ws <= "001";
end if;
light_en <= "001";
when "10000000000" =>
    if (pa='1')then
        light_ew <= "00" & sig_1s;
    else
        light_ew <= "001";
    end if;
    if (va='1')then
        light_we <= "00" & sig_1s;
    else
        light_we <= "001";
    end if;
    light_ns <= "001";
    if (vm='1')then
        light_sn <= "00" & sig_1s;
    else
        light_sn <= "001";
    end if;
    if (mal='1')then
        light_ws <= "00" & sig_1s;
    else
        light_ws <= "001";
    end if;
    light_en <= "001";
when others =>
    light_ew <= "111";

```

```

        light_we <= "111";
        light_ns <= "111";
        light_sn <= "111";
        light_ws <= "110";
        light_en <= "110";

    end case;
    if (va='1' or mal='1') then
        light_dt(0)<='1';
    else
        light_dt(0)<='0';
    end if;
    if (vm='1') then
        light_dt(1)<='1';
    else
        light_dt(1)<='0';
    end if;
    if (pa='1') then
        light_dt(2)<='1';
    else
        light_dt(2)<='0';
    end if;

    end process;
end Behavioral;

```

7.6 Control Path

entity main_controller_unit is


```

Port(clk : in std_logic;
      reset : in std_logic;
      at : in std_logic;
      aw : in std_logic;
      ht : in std_logic;
      t2sec : in std_logic;
      t4sec : in std_logic;
      t5sec : in std_logic;
      t8sec : in std_logic;
      t15sec : in std_logic;
      t25sec : in std_logic;
      t30sec : in std_logic;
      t1min : in std_logic;
      t1min15sec : in std_logic;
      t2sec_en : out std_logic;
      t4sec_en : out std_logic;
      t5sec_en : out std_logic;
      t8sec_en : out std_logic;
      t15sec_en : out std_logic;
      t25sec_en : out std_logic;
      t30sec_en : out std_logic;
      t1min_en : out std_logic;
      t1min15sec_en : out std_logic;
      t_reset : out std_logic;
      state : out std_logic_vector(10 downto 0));
end main_controller_unit;

```

architecture Behavioral of main_controller_unit is

```

type main_fsm is (green_ew, amber_ew, green_n, amber_n, green_s, amber_s, green_enws,
                  amber_enws, init, dly, green_at, amber_at);
signal pstate, nstate, dstate, tstate : main_fsm;

```

```
begin
```

```
clkPROC : process(clk, reset)
```

```
begin
```

```
    if(reset = '1')then
```

```
        pstate <= init;
```

```
    elsif(clk'event and clk = '1')then
```

```
        pstate <= dstate;
```

```
    end if;
```

```
end process;
```

```
controlPROC :
```

```
process(pstate,t1min15sec,t1min,t30sec,t25sec,t15sec,t8sec,t5sec,t4sec,t2sec,ht,aw,at,tstate)
```

```
begin
```

```
    case pstate is
```

```
        when init =>
```

```
            t2sec_en <= '0';
```

```
            t4sec_en <= '0';
```

```
            t5sec_en <= '0';
```

```
            t8sec_en <= '0';
```

```
            t15sec_en <= '0';
```

```
            t25sec_en <= '0';
```

```
            t30sec_en <= '0';
```

```
            t1min_en <= '0';
```

```
            t1min15sec_en <= '0';
```

```
            t_reset <= '1';
```

```
            state <= "000000000000";
```

```
            dstate <= green_ew;
```

```

when green_ew=>
    state <= "000000000001";
    t_reset<='0';
    if (ht='1')then
        t1min15sec_en <= '1';
        t1min_en <= '0';
    else
        t1min_en <= '1';
        t1min15sec_en <= '0';
    end if;
    t2sec_en <= '0';
    t4sec_en <= '0';
    t5sec_en <= '0';
    t8sec_en <= '0';
    t15sec_en <= '0';
    t25sec_en <= '0';
    t30sec_en <= '0';
    tstate <= green_ew;
    if (t1min='1' or t1min15sec='1')then
        t_reset<='1';
        dstate <= amber_ew;
    end if;

```

```

when amber_ew =>
    state <= "000000000010";
    t_reset<='0';
    if (aw='1') then
        t15sec_en <= '1';
        t8sec_en <= '0';
    else
        t8sec_en <= '1';
    end if;

```

```

        t15sec_en <= '0';
    end if;
    t2sec_en <= '0';
    t4sec_en <= '0';
    t5sec_en <= '0';
    t25sec_en <= '0';
    t30sec_en <= '0';
    t1min_en <= '0';
    t1min15sec_en <= '0';
    tstate <= amber_ew;
    if (t8sec='1' or t15sec='1')then
        t_reset<='1';
        nstate <= green_n;
        dstate <= dly;
    end if;

    when green_n=>
        state <= "00000000100";
        t_reset<='0';
        if (ht='1')then
            t25sec_en <= '1';
            t30sec_en <= '0';
        else
            t30sec_en <= '1';
            t25sec_en <= '0';
        end if;
        t2sec_en <= '0';
        t4sec_en <= '0';
        t5sec_en <= '0';
        t8sec_en <= '0';
        t15sec_en <= '0';
        t1min_en <= '0';

```

```

        t1min15sec_en <= '0';
        tstate <= green_n;
    if (t30sec='1' or t25sec='1')then
        t_reset<='1';
        dstate <= amber_n;
    end if;

when amber_n =>
    state <= "00000001000";
    t_reset<='0';
    if (aw='1') then
        t8sec_en <= '1';
        t5sec_en <= '0';
    else
        t5sec_en <= '1';
        t8sec_en <= '0';
    end if;
    t2sec_en <= '0';
    t4sec_en <= '0';
    t15sec_en <= '0';
    t25sec_en <= '0';
    t30sec_en <= '0';
    t1min_en <= '0';
    t1min15sec_en <= '0';
    tstate <= amber_n;
    if (t8sec='1' or t5sec='1')then
        t_reset<='1';
        nstate <= green_s;
        dstate <= dly;
    end if;

when green_s=>

```

```

state <= "00000010000";
t_reset<='0';
if (ht='1')then
    t25sec_en <= '1';
    t30sec_en <= '0';
else
    t30sec_en <= '1';
    t25sec_en <= '0';
end if;
t2sec_en <= '0';
t4sec_en <= '0';
t5sec_en <= '0';
t8sec_en <= '0';
t15sec_en <= '0';
t1min_en <= '0';
t1min15sec_en <= '0';
tstate <= green_s;
if (t30sec='1' or t25sec='1')then
    t_reset<='1';
    dstate <= amber_s;
end if;

when amber_s =>
    state<= "000000100000";
    t_reset<='0';
    if (aw='1') then
        t8sec_en <= '1';
        t5sec_en <= '0';
    else
        t5sec_en <= '1';
        t8sec_en <= '0';
    end if;

```

```

t2sec_en <= '0';
t4sec_en <= '0';
t15sec_en <= '0';
t25sec_en <= '0';
t30sec_en <= '0';
t1min_en <= '0';
t1min15sec_en <= '0';
tstate <= amber_s;
if (t8sec='1' or t5sec='1')then
    t_reset<='1';
    nstate <= green_enws;
    dstate <= dly;
end if;

when green_enws=>
    state <= "00001000000";
    t_reset<='0';
    t2sec_en <= '0';
    t4sec_en <= '0';
    t5sec_en <= '0';
    t8sec_en <= '0';
    t15sec_en <= '1';
    t25sec_en <= '0';
    t30sec_en <= '0';
    t1min_en <= '0';
    t1min15sec_en <= '0';
    tstate <= green_enws;
if (t15sec='1')then
    t_reset<='1';
    dstate <= amber_enws;
end if;

```

```

when amber_enws =>
    state <= "000100000000";
    t_reset<='0';
    if (aw='1') then
        t8sec_en <= '1';
        t5sec_en <= '0';
    else
        t5sec_en <= '1';
        t8sec_en <= '0';
    end if;
    t2sec_en <= '0';
    t4sec_en <= '0';
    t15sec_en <= '0';
    t25sec_en <= '0';
    t30sec_en <= '0';
    t1min_en <= '0';
    t1min15sec_en <= '0';
    tstate <= amber_enws;
    if (t8sec='1' or t5sec='1')then
        t_reset<='1';
        nstate <= green_ew;
        dstate <= dly;
    end if;

when dly =>
    state <= "100000000000";
    t_reset<='0';
    t5sec_en <= '0';
    t8sec_en <= '0';
    t15sec_en <= '0';
    t25sec_en <= '0';
    t30sec_en <= '0';

```



```

t1min_en <= '0';
t1min15sec_en <= '0';
if (aw='1') then
    t4sec_en <= '1';
    t2sec_en <= '0';
else
    t2sec_en <= '1';
    t4sec_en <= '0';
end if;

if (t2sec='1' or t4sec='1')then
    t_reset<='1';
    dstate <= nstate;
end if;

when green_at=>
    t_reset<='0';
    state <= "00100000000";
    t2sec_en <= '0';
    t4sec_en <= '0';
    t5sec_en <= '0';
    t8sec_en <= '0';
    t15sec_en <= '1';
    t25sec_en <= '0';
    t30sec_en <= '0';
    t1min_en <= '0';
    t1min15sec_en <= '0';
    if (t15sec='1')then
        t_reset<='1';
        dstate <= amber_at;
    end if;

```

```

when amber_at=>
    t_reset<='0';
    state <= "010000000000";
    t2sec_en <= '0';
    t4sec_en <= '1';
    t5sec_en <= '0';
    t8sec_en <= '0';
    t15sec_en <= '0';
    t25sec_en <= '0';
    t30sec_en <= '0';
    t1min_en <= '0';
    t1min15sec_en <= '0';
    if (t4sec='1')then
        t_reset<='1';
        dstate <= dly;
    end if;
    end case;
    if (at='1') then
        t_reset<='1';
        nstate<=tstate;
        dstate<=green_at;
    end if;
end process;
end Behavioral;

```