25/25

CSE 312 Electronic Design Automation

Traffic Light System

PROJECT REPORT

Submitted to:

Prof. Dr. Hassan Ali Hassan

Eng. Ahmed Fawzy

Submitted by:

Youssef George Fouad	19P9824
Mostafa Nasrat Metwally	19P4619
Kerollos Wageeh Youssef	19P3468
Anthony Amgad Fayek	19P9880

Table of Contents

List of Figures

0. INTRODUCTION

This project is a traffic light system that manages the cars and pedestrians' traffic in two intersecting highways like the two shown in the below figure.

This project is developed using VHDL language, in order to be synthesized and implemented on an FPGA that will be connected with the real corresponding ports on the road.

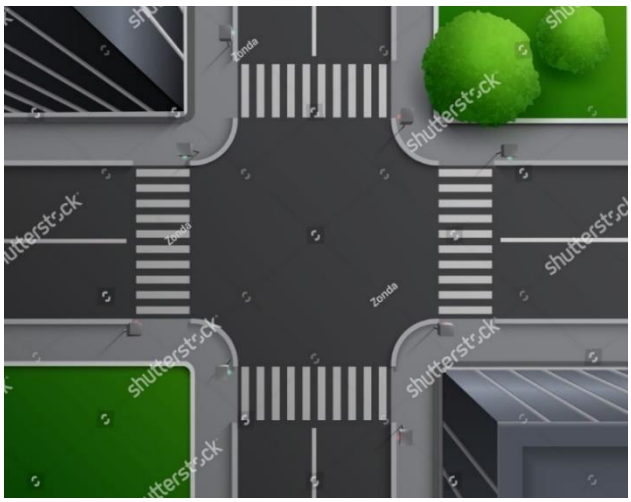

Figure 1: Highways Intersection

It is assumed that the movement of cars is restricted to only moving straight in one direction, so turning to the other road is not allowed.

Each one of the four road directions has two traffic lights, one for pedestrians (colored later on in blue) and one for cars (colored later on in black), as shown in the next page.

Each traffic light has the usual 3 colors: red, yellow, and green lights. Moreover, each will have a screen showing a countdown timer to indicate when the light is going to change, as shown in the next figure.

Figure 2: Traffic Light Model

There are four emergency triggers that may be sent in case of any emergency in one of the four roads in order to allow the movement in this road instantly.

1. Project Description

1.1. Outputs

Each traffic light is represented as a 3-bit *STD_LOGIC_VECTOR*, where the first bit represents the RED light, the second represents the YELLOW light, and the third one represents the GREEN light.

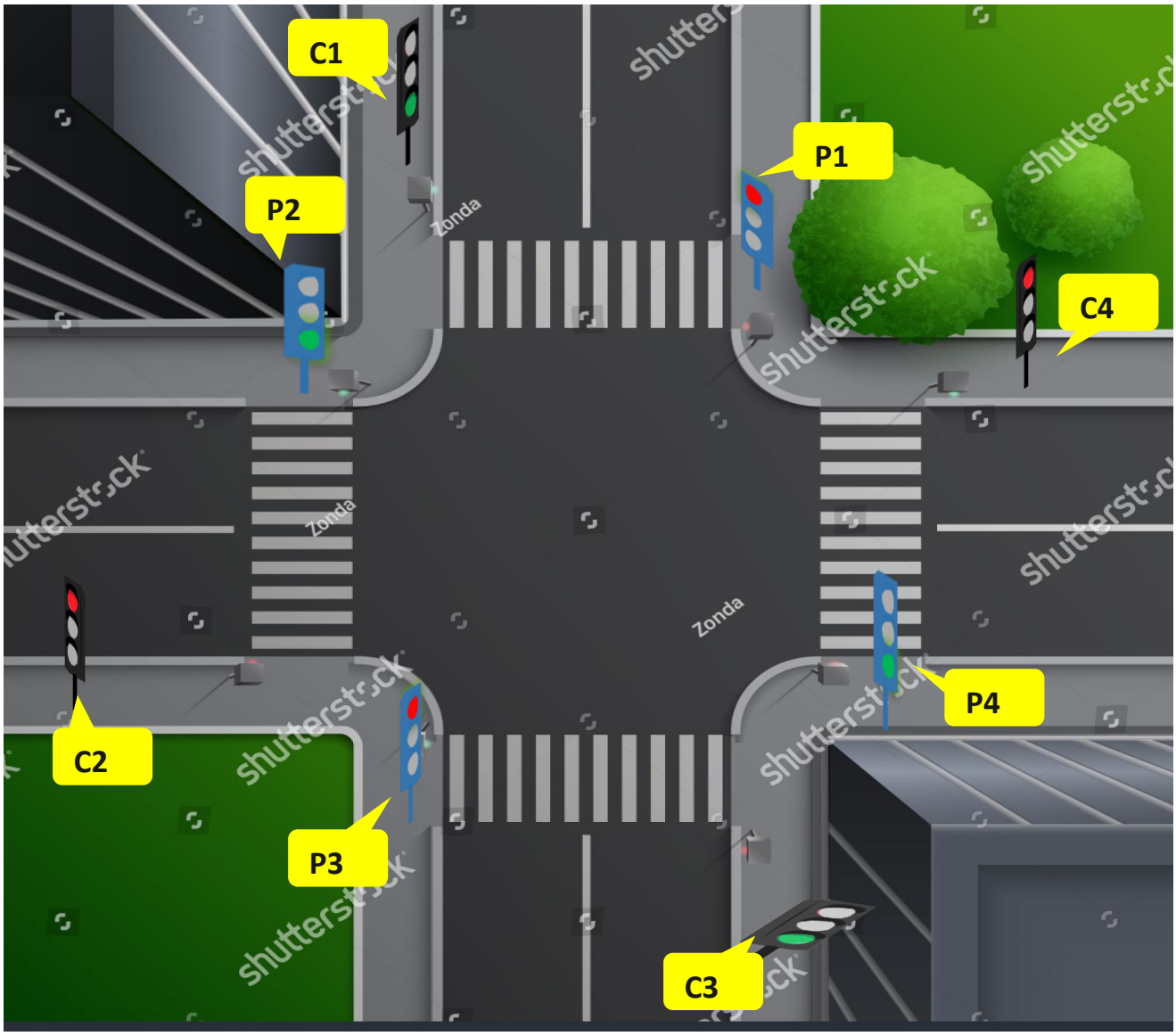

Figure 3: Traffic Lights Description

In addition to the traffic light outputs **C1, C2, C3, C4, P1, P2, P3** and **P4** that are shown above, there are 4 timers that countdown between the change of the traffic light from red to green, or vice versa. They are implemented as 5-bit *STD_LOGIC_VECTOR*s that should be decoded later using the appropriate 7-segment decoders to be shown properly on the traffic light screen.

1.2. Inputs

There are 4 emergency inputs: **Emergency1**, **Emergency2**, **Emergency3** and **Emergency4**, that should be triggered in case of any emergency to allow the movement in the corresponding road instantly.

As a typical finite state machine, there are **clock** and **reset** inputs to the circuit. The clock period is set to 1 second to be synchronous with the timer change. While the **reset** input resets the circuit state when triggered to HIGH and is used in case of any failure.

2. State Diagram

In our project, there are 6 states as shown in the state diagram below. It is represented as a Moore FSM, this is because the traffic light outputs **(C1, C2, C3, C4, P1, P2, P3** and **P4**) depends only on the current state. So, each state is accompanied with its corresponding outputs.

Moreover, when the **reset** input is triggered high, the present state changes asynchronously to the initial one (**st0_C1_C3_P2_P4_GREEN**). Also, **Emergency1** and **Emergency3** sets the state to (**st5_SafetyState**) when triggered. Alike, **Emergency2** and **Emergency4** sets the state to (**st2_SafetyState**) when triggered.

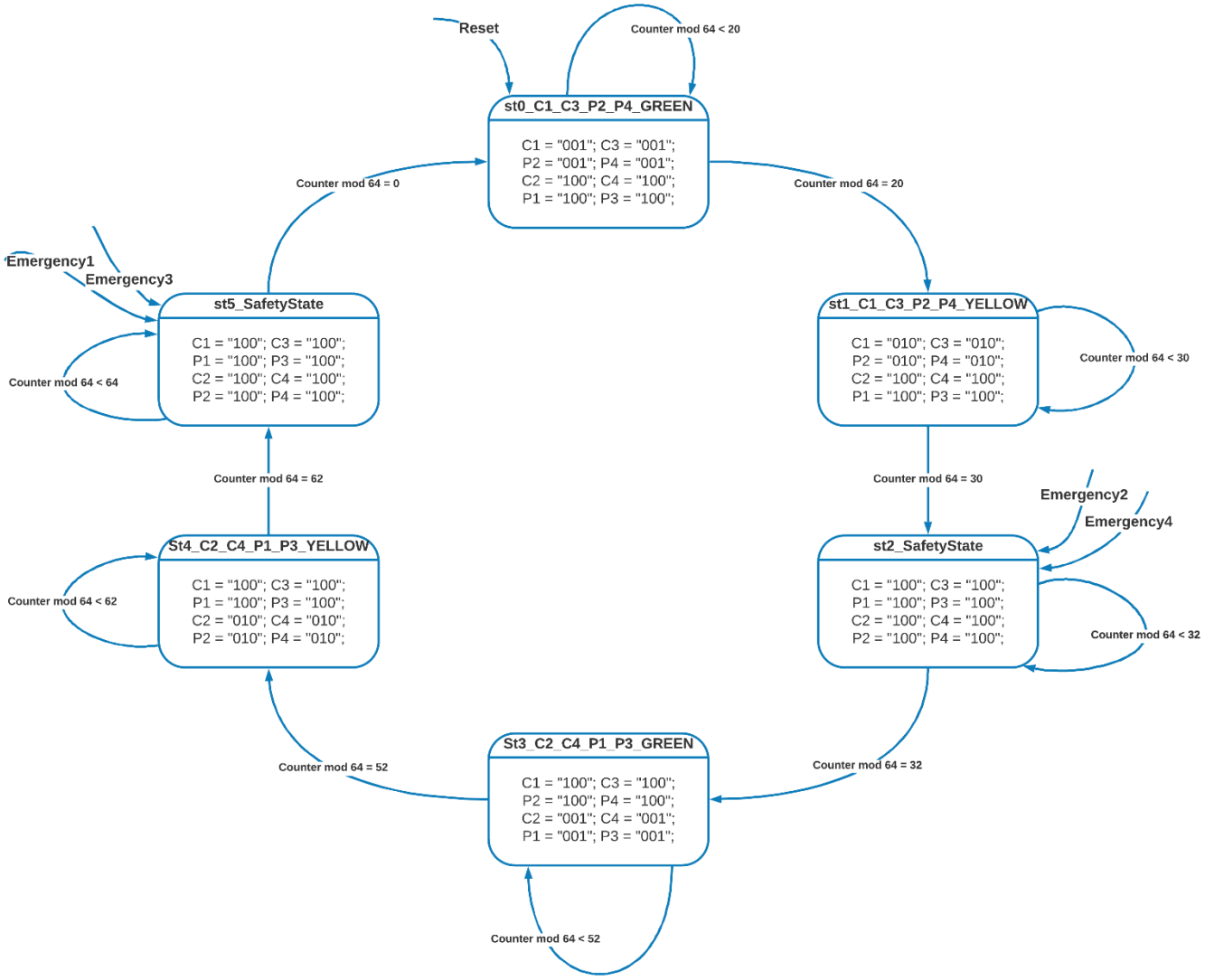

Figure 4: State Diagram

2.1. State 0 (st0_C1_C3_P2_P4_GREEN)

In this state, cars movement is allowed in Highway1, so **C1** and **C3** are GREEN for 20 seconds. While pedestrians' movement is allowed across Highway2, so **P2** and **P4** are GREEN for also 20 seconds.

```
C1 <= "001"; C3 <= "001"; P2 <= "001"; P4 <= "001";     -- GREEN
C2 <= "100"; C4 <= "100"; P1 <= "100"; P3 <= "100";     -- RED
```

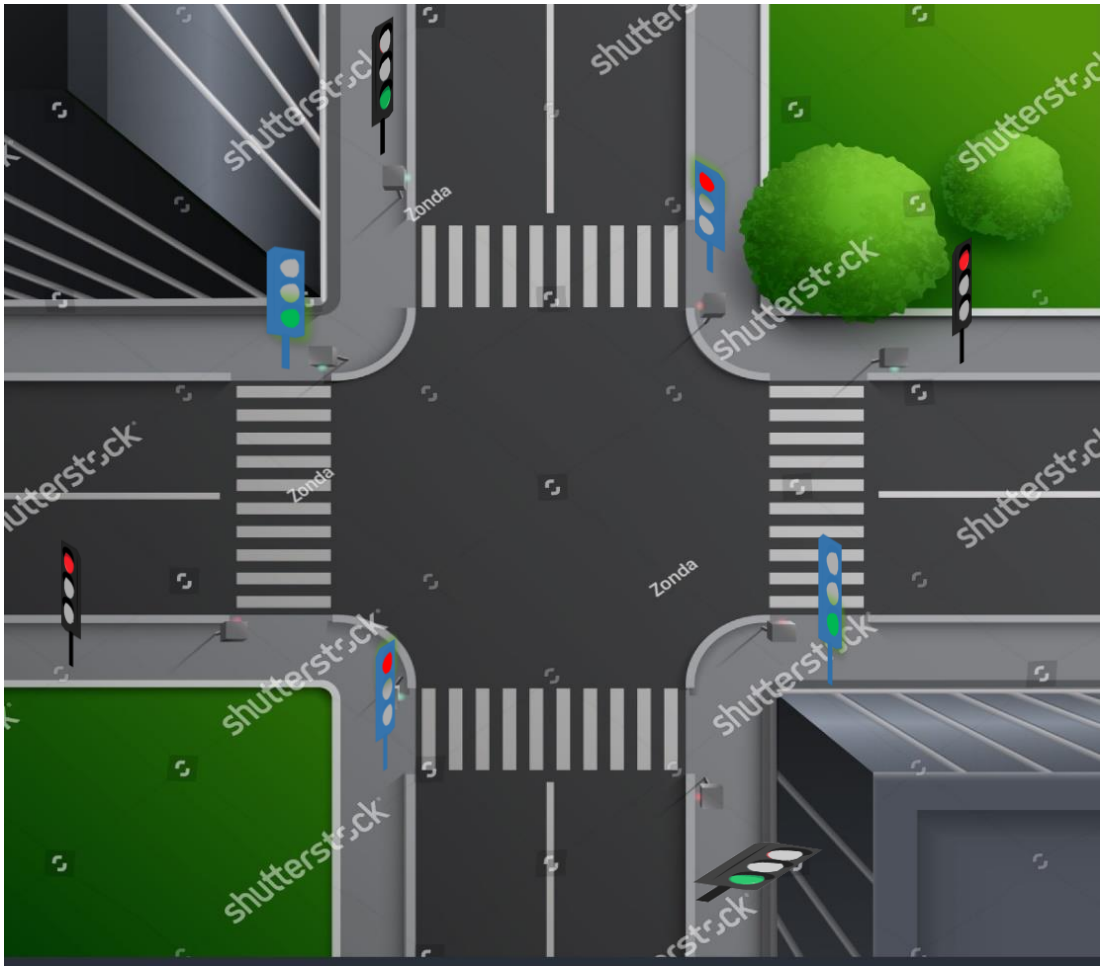

Figure 5: State 0

2.2. State 1 (st1_C1_C3_P2_P4_YELLOW)

In this state, cars movement is still allowed in Highway1, but **C1** and **C3** are changed to YELLOW for another 10 seconds. Also, pedestrians' movement is still allowed across Highway2, but **P2** and **P4** are changed to YELLOW for also 10 seconds.

```
C1 <= "010"; C3 <= "010"; P2 <= "010"; P4 <= "010";     -- YELLOW
C2 <= "100"; C4 <= "100"; P1 <= "100"; P3 <= "100";     -- RED
```

Figure 6: State 1

2.3. State 2 (st2_SafetyState)

To ensure that no accidents happen, we added an additional safety state between the change of movement from one road to the other. During this safety state, all traffic lights are kept RED for 2 seconds.

Note: When Emergency2 or Emergency4 is triggered indicating an emergency case in Highway2, the present state is asynchronously switched to this state.

```
C1 <= "100"; C3 <= "100"; P2 <= "100"; P4 <= "100";     -- RED
C2 <= "100"; C4 <= "100"; P1 <= "100"; P3 <= "100";     -- RED
```

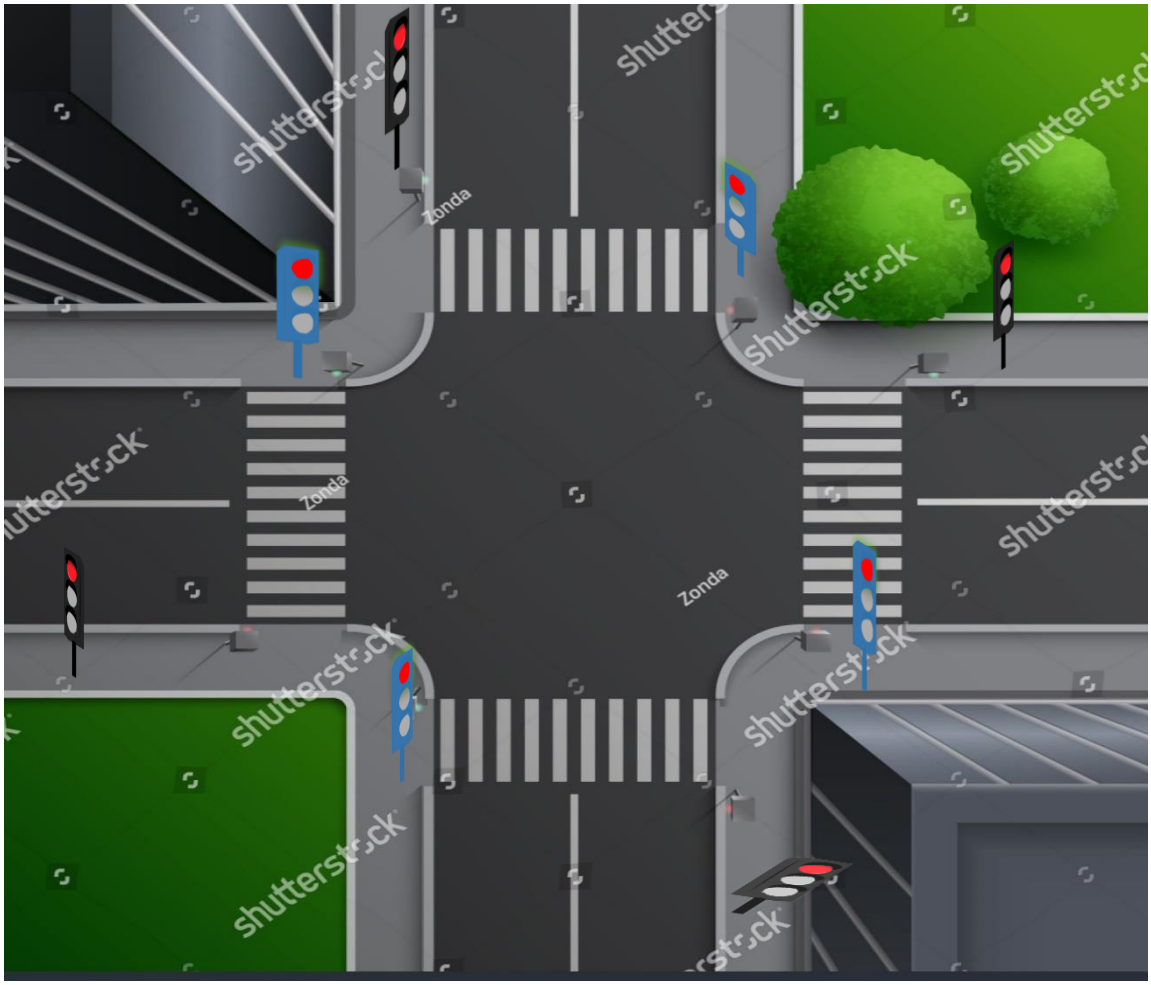

Figure 7: State 2

2.4. State 3 (St3_C2_C4_P1_P3_GREEN)

In this state, cars movement is switched to Highway2, so **C2** and **C4** are GREEN for 20 seconds. While pedestrians' movement is switched to move across Highway1, so **P1** and **P3** are GREEN for also 20 seconds.

```
C1 <= "100"; C3 <= "100"; P2 <= "100"; P4 <= "100";     -- RED
C2 <= "001"; C4 <= "001"; P1 <= "001"; P3 <= "001";     -- GREEN
```

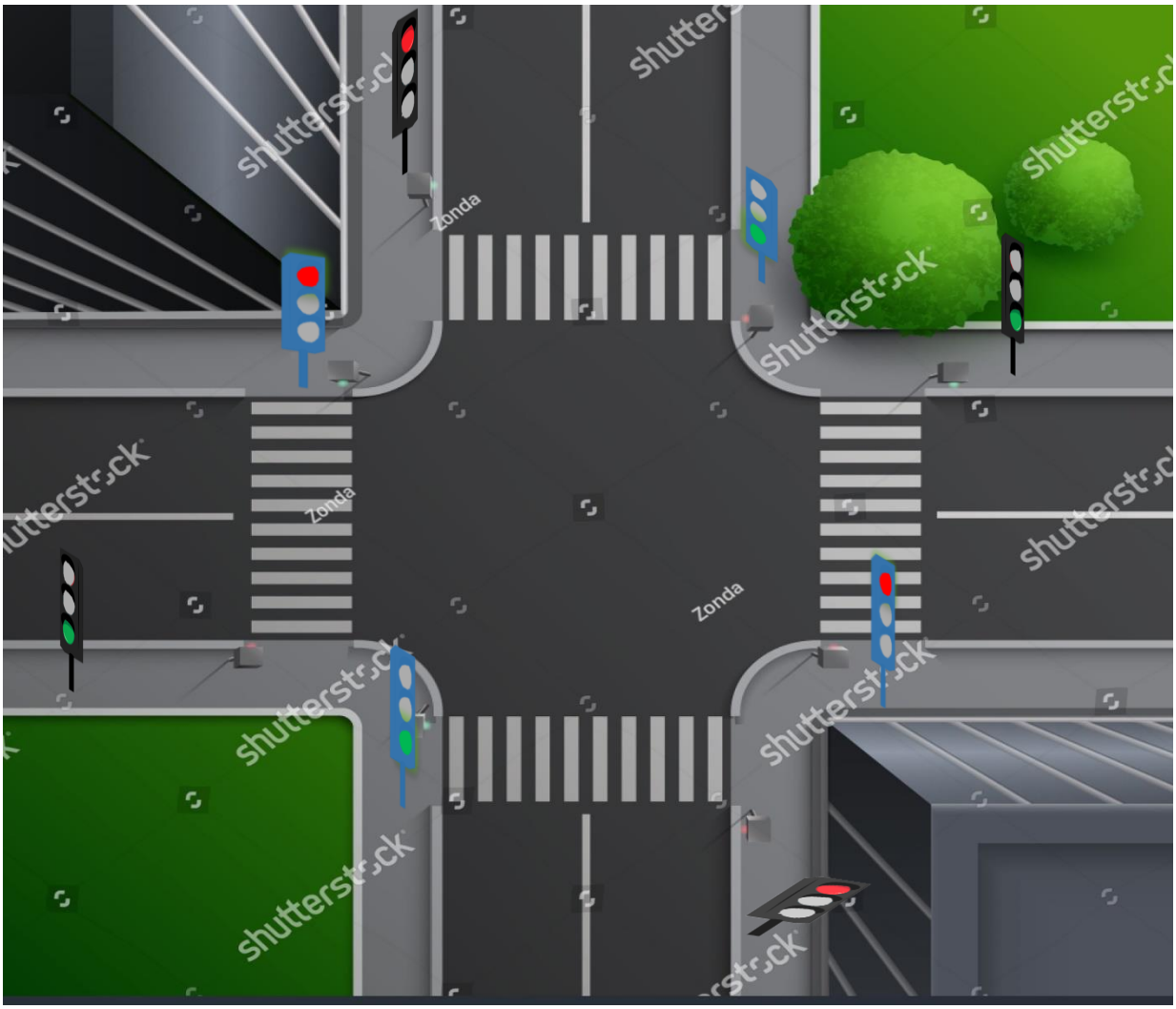

Figure 8: State 3

2.5. State 4 (St4_C2_C4_P1_P3_YELLOW)

In this state, cars movement is still allowed in Highway2, but **C2** and **C4** are changed to YELLOW for another 10 seconds. Also, pedestrians' movement is still allowed across Highway1, but **P1** and **P3** are changed to YELLOW for also 10 seconds.

```
C1 <= "100"; C3 <= "100"; P2 <= "100"; P4 <= "100";     -- RED
C2 <= "010"; C4 <= "010"; P1 <= "010"; P3 <= "010";     -- YELLOW
```

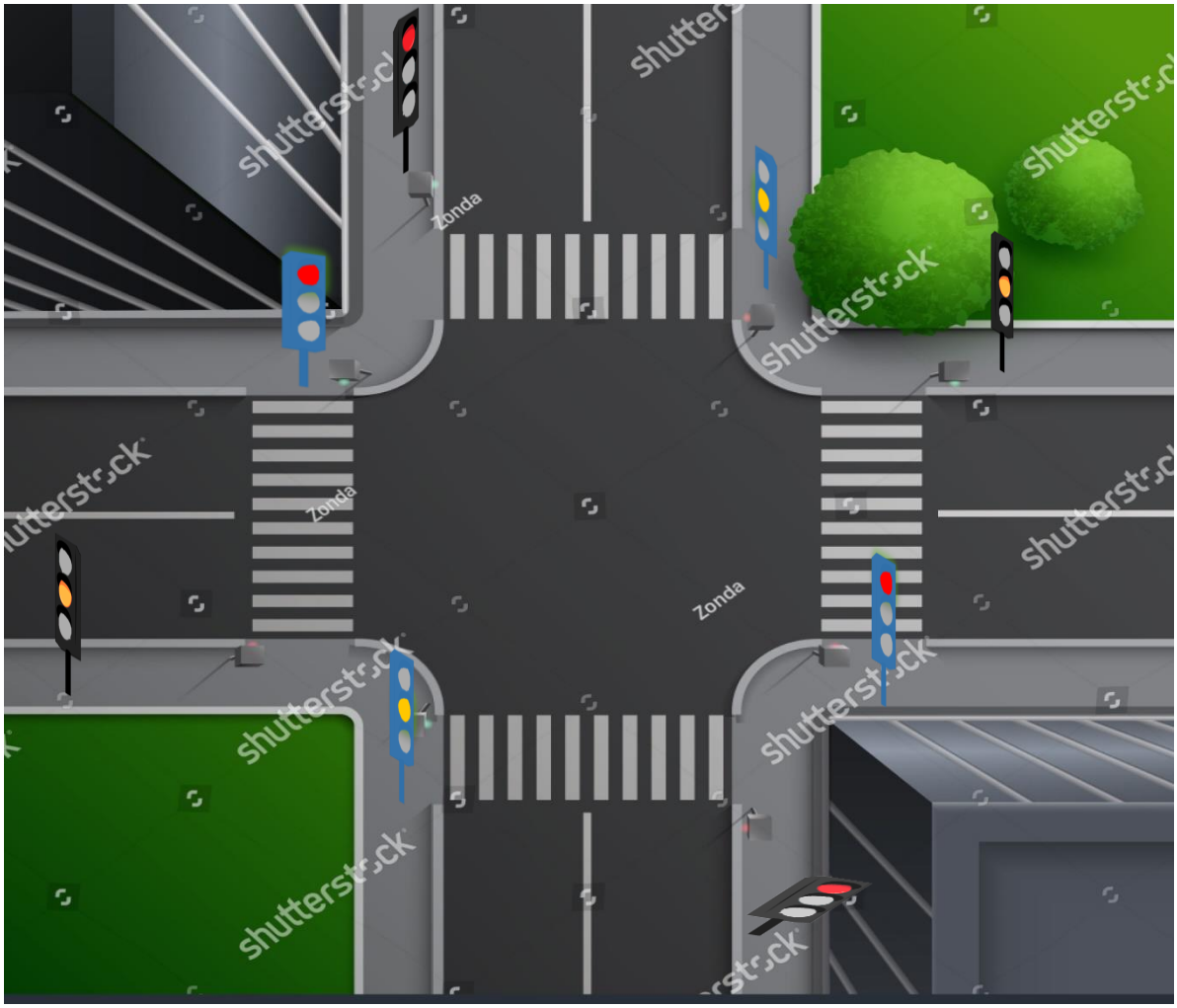

Figure 9: State 4

2.6. State 5 (st5_SafetyState)

This state is the same as State 2, where all traffic lights are kept RED for 2 seconds. However, it only differs from State 2 in being switched later to the initial state (State 0) instead of State 3.

Note: When Emergency1 or Emergency3 is triggered indicating an emergency case in Highway1, the present state is asynchronously switched to this state.

```
C1 <= "100"; C3 <= "100"; P2 <= "100"; P4 <= "100";     -- RED
C2 <= "100"; C4 <= "100"; P1 <= "100"; P3 <= "100";     -- RED
```

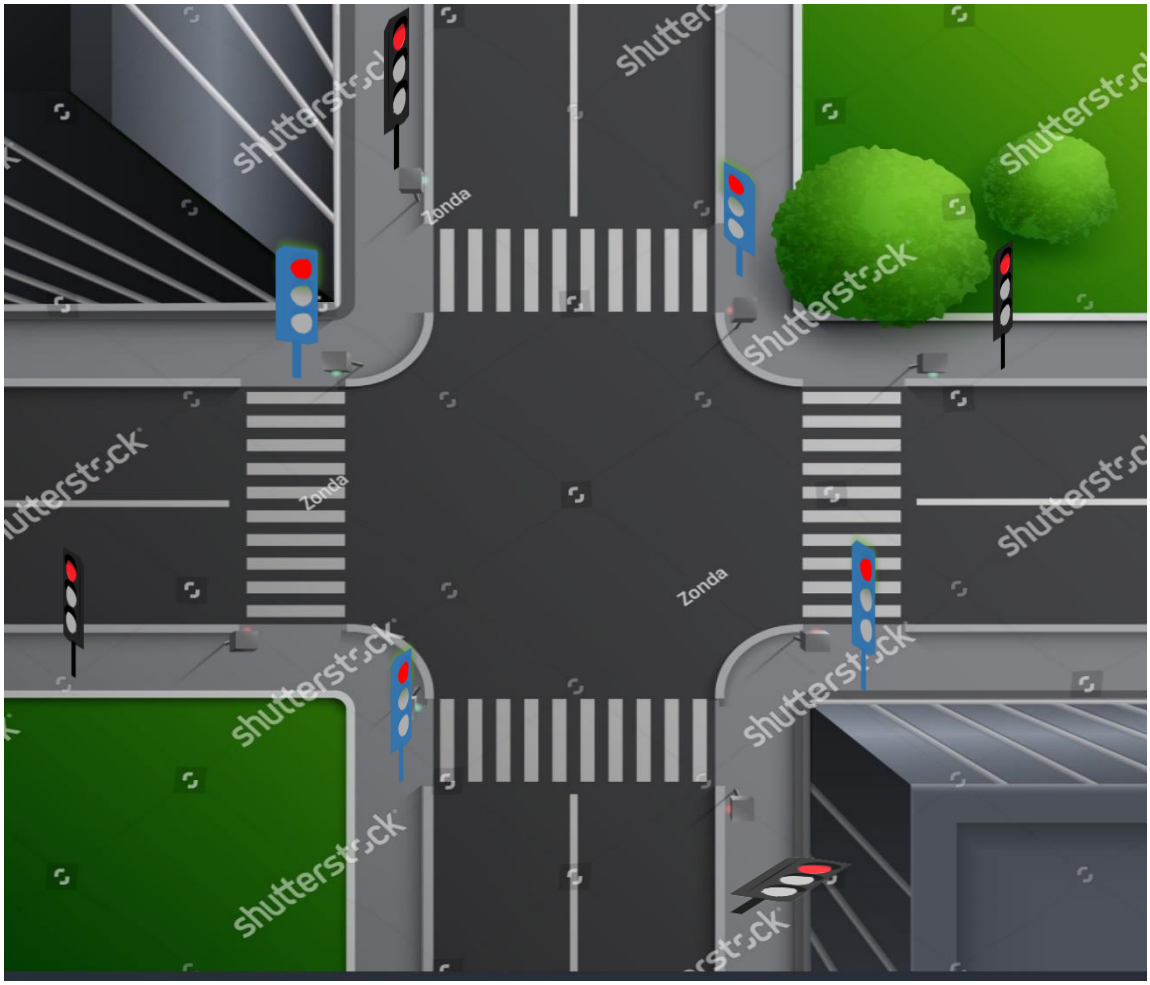

Figure 10: State 5

3. VHDL Code

3.1. Main Entity

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
entity TLC is
    Port (
        -- TL has 3 bits for RED, YELLOW, GREEN respectivley
        C1, C3, P1, P3 :out STD_LOGIC_Vector(2 downto 0);    -- Cars & Pedestrians traffic
lights in highway 1
        C2, C4, P2, P4 :out STD_LOGIC_Vector(2 downto 0);    -- Cars & Pedestrians traffic
lights in highway 2

        Emergency1, Emergency3 : in STD_LOGIC;        -- Emergrncy signals in highway 1
        Emergency2, Emergency4 : in STD_LOGIC;        -- Emergrncy signals in highway 2

        Timer1, Timer3 : out STD_LOGIC_Vector(4 downto 0);  -- Screen Countdown Timers in
highway 1
        Timer2, Timer4 : out STD_LOGIC_Vector(4 downto 0);  -- Screen Countdoen Timers in
highway 2

        clk, reset : in STD_LOGIC);

end TLC;


architecture Behavioral of TLC is

    type state_type is ( st0_C1_C3_P2_P4_GREEN,    -- Cars in Highway1 (C1,C3) &
Pedesterians in Highway2 (P2,P4) are GREEN
                         st1_C1_C3_P2_P4_YELLOW,   -- Cars in Highway1 (C1,C3) &
Pedesterians in Highway2 (P2,P4) are YELLOW
                         st2_SafetyState,          -- All RED before switching traffic
                         St3_C2_C4_P1_P3_GREEN,    -- Cars in Highway2 (C2,C4) &
Pedesterians in Highway1 (P1,P3) are GREEN
                         St4_C2_C4_P1_P3_YELLOW,   -- Cars in Highway2 (C2,C4) &
Pedesterians in Highway1 (P1,P3) are YELLOW
                         st5_SafetyState);         -- All RED before switching traffic

    signal state: state_type;
    signal counter: integer := 1;    -- no. of passed seconds
    signal t1, t2 : std_logic_vector(4 downto 0) := "00000";
    constant sec30 : std_logic_vector(4 downto 0) := "11110";
    constant sec0  : std_logic_vector(4 downto 0) := "00000";
```

```
begin

  Timer1 <= t1;
  Timer3 <= t1;
  Timer2 <= t2;
  Timer4 <= t2;

    process (clk, reset, Emergency1, Emergency2, Emergency3, Emergency4)
        begin
            if (reset='1') or (counter>1000000000) then
                state <= st0_C1_C3_P2_P4_GREEN;     -- reset to initial
state
                counter <= 1;                 -- reset counter
                t1 <= sec30;              -- reset Timer
                t2 <= sec30;

            elsif (Emergency1 = '1') or (Emergency3 = '1') then
              state <= st5_SafetyState;
              counter <= -1;
              t1 <= sec0;
                t2 <= sec0;

            elsif (Emergency2 = '1') or (Emergency4 = '1') then
              state <= st2_SafetyState;
              counter <= 30;
              t1 <= sec0;
                t2 <= sec0;

            elsif clk' event and clk = '1' then

                case (state) is

                    when st0_C1_C3_P2_P4_GREEN =>
                        if (counter mod 64) = 20 then   -- 20s of state0 passed
                            state <= st1_C1_C3_P2_P4_YELLOW ;
                        else
                            state <= st0_C1_C3_P2_P4_GREEN;
                        end if;
                            t1 <= t1 - 1;
                            t2 <= t2 - 1;

                    when st1_C1_C3_P2_P4_YELLOW =>
                        if (counter mod 64) = 30 then   -- 10s of state1 passed
                            state <= st2_SafetyState ;
                            t1 <= sec0;
                            t2 <= sec0;
```

```
                    else
                        state <= st1_C1_C3_P2_P4_YELLOW;
                        t1 <= t1 - 1;
                        t2 <= t2 - 1;
                    end if;

                when st2_SafetyState =>
                    if (counter mod 64) = 32 then   -- 2s of state2 passed
                        state <= St3_C2_C4_P1_P3_GREEN;
                        t1 <= sec30;
                        t2 <= sec30;
                    else
                        state <= st2_SafetyState;
                        t1 <= sec0;
                        t2 <= sec0;
                    end if;

                when St3_C2_C4_P1_P3_GREEN =>
                    if (counter mod 64)= 52 then    -- 20s of state3 passed
                        state <= St4_C2_C4_P1_P3_YELLOW;
                    else
                        state <= St3_C2_C4_P1_P3_GREEN;
                    end if;
                        t1 <= t1 - 1;
                        t2 <= t2 - 1;

                when St4_C2_C4_P1_P3_YELLOW =>
                    if (counter mod 64)= 62 then    -- 10s of state4 passed
                        state <= st5_SafetyState;
                        t1 <= sec0;
                        t2 <= sec0;
                    else
                        state <= St4_C2_C4_P1_P3_YELLOW;
                        t1 <= t1 - 1;
                        t2 <= t2 - 1;
                    end if;

                when st5_SafetyState =>
                    if (counter mod 64)=0 then      -- 2s of state5 passed (i.e. End
of cycle)
                        state <= st0_C1_C3_P2_P4_GREEN;
                        t1 <= sec30;
                        t2 <= sec30;
                    else
                        state <=st5_SafetyState;
                        t1 <= sec0;
                        t2 <= sec0;
```

```
                    end if;

                   when others =>
                    state <= st0_C1_C3_P2_P4_GREEN;
                    t1 <= sec30;
                    t2 <= sec30;
            end case;

        counter <= counter + 1;  -- increment the no. of passed seconds
        end if;
    end process;

    OUTPUT_DECODE: process (state)
        begin
            case state is

                when st0_C1_C3_P2_P4_GREEN =>
                    C1 <= "001"; C3 <= "001"; P2 <= "001"; P4 <= "001";    -- GREEN
                    C2 <= "100"; C4 <= "100"; P1 <= "100"; P3 <= "100";    -- RED

                when st1_C1_C3_P2_P4_YELLOW =>
                    C1 <= "010"; C3 <= "010"; P2 <= "010"; P4 <= "010";    -- YELLOW
                    C2 <= "100"; C4 <= "100"; P1 <= "100"; P3 <= "100";    -- RED

                when st2_SafetyState =>
                    C1 <= "100"; C3 <= "100"; P2 <= "100"; P4 <= "100";    -- RED
                    C2 <= "100"; C4 <= "100"; P1 <= "100"; P3 <= "100";    -- RED

                when St3_C2_C4_P1_P3_GREEN =>
                    C1 <= "100"; C3 <= "100"; P2 <= "100"; P4 <= "100";    -- RED
                    C2 <= "001"; C4 <= "001"; P1 <= "001"; P3 <= "001";    -- GREEN

                when St4_C2_C4_P1_P3_YELLOW =>
                    C1 <= "100"; C3 <= "100"; P2 <= "100"; P4 <= "100";    -- RED
                    C2 <= "010"; C4 <= "010"; P1 <= "010"; P3 <= "010";    -- YELLOW

                when st5_SafetyState =>
                    C1 <= "100"; C3 <= "100"; P2 <= "100"; P4 <= "100";    -- RED
                    C2 <= "100"; C4 <= "100"; P1 <= "100"; P3 <= "100";    -- RED

                when others =>
                    C1 <= "100"; C3 <= "100"; P2 <= "100"; P4 <= "100";    -- RED
                    C2 <= "100"; C4 <= "100"; P1 <= "100"; P3 <= "100";    -- RED
            end case;

        end process;
end Behavioral;
```

3.2. Test Bench

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

entity TLC_test is
end entity;

architecture tb of TLC_test is

signal C1, C3, P1, P3, C2, C4, P2, P4: STD_LOGIC_Vector(2 downto 0);
signal Emergency1, Emergency3, Emergency2, Emergency4 : STD_LOGIC;
signal Timer1, Timer3, Timer2, Timer4 : STD_LOGIC_Vector(4 downto 0);
signal Clk,Reset: STD_LOGIC;

begin

  DUT : ENTITY work.TLC PORT MAP(C1, C3, P1, P3, C2, C4, P2, P4,
                                 Emergency1, Emergency3, Emergency2, Emergency4,
                                 Timer1, Timer3, Timer2, Timer4,
                                 Clk, Reset);

  Clock : process
  begin
    Clk <= '0';
    wait for 500 ms;
    Clk <= '1';
    wait for 500 ms;
  end process;

  stimulis : process
  begin
    report("Starting simulation");
    Reset <= '1';   -- Enable reset
    Emergency1 <= '0'; Emergency2 <= '0';
    Emergency3 <= '0'; Emergency4 <= '0';
    wait for 1 sec;

    Reset <= '0';   -- Disable reset
    wait for 100 sec;


    Emergency1 <= '1';    -- Send Emergency signal in Highway1
    wait for 1 sec;
    Emergency1 <= '0';
    wait for 25 sec;
```

```
    Emergency2 <= '1';     -- Send Emergency signal in Highway2
    wait for 1 sec;
    Emergency2 <= '0';
    wait for 500 sec;

    report("End simulation");
  end process;

end architecture;
```

4. Simulation

At first, the present state in set initially to State 0 (**st0_C1_C3_P2_P4_GREEN**).

After 20 seconds (when the countdown reaches 10), the state is switched to State 1 (**st1_C1_C3_P2_P4_YELLOW**).

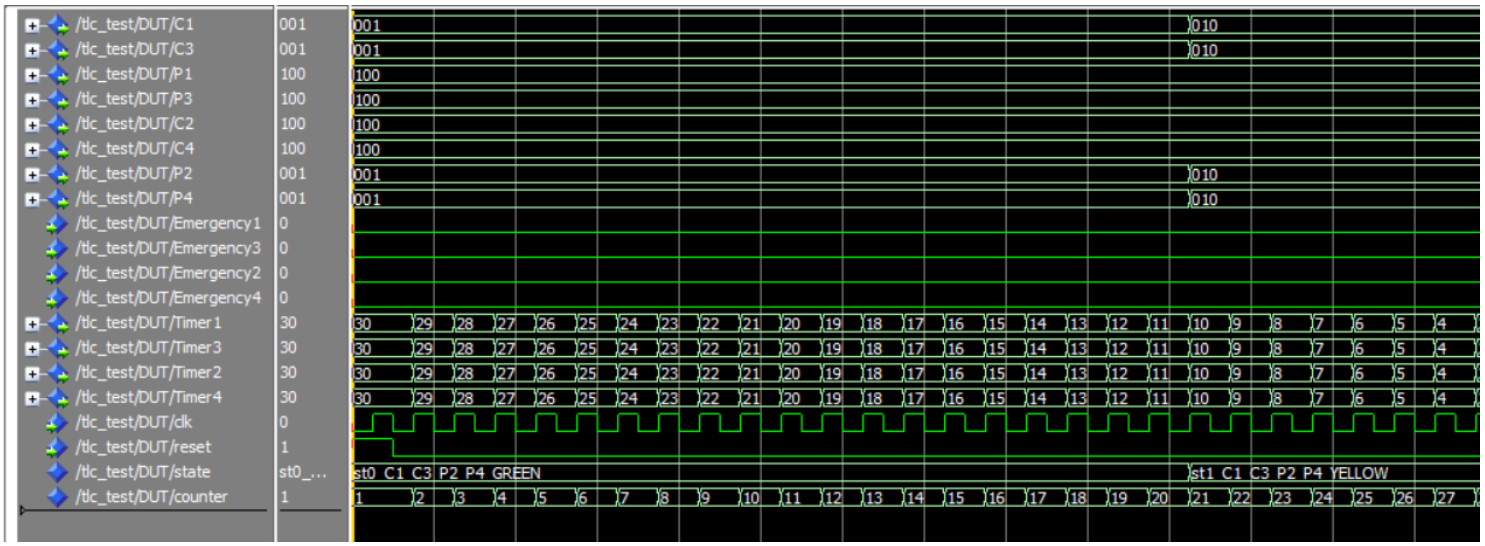

Figure 11: Simulation Screenshot 1

State 1 remains for another 10 seconds, and once the countdown reaches 0, the present state is changed to State 2 (**st2_SafetyState**) that lasts for only 2 seconds with the timers showing 0 on the screen.

After this safety period, the countdown is reset to 30 again. Also, the state is switched to State 3 (**st3_C2_C4_P1_P3_GREEN**) for 20 seconds, then to State 4 (**st4_C2_C4_P1_P3_YELLOW**) for 10 seconds.

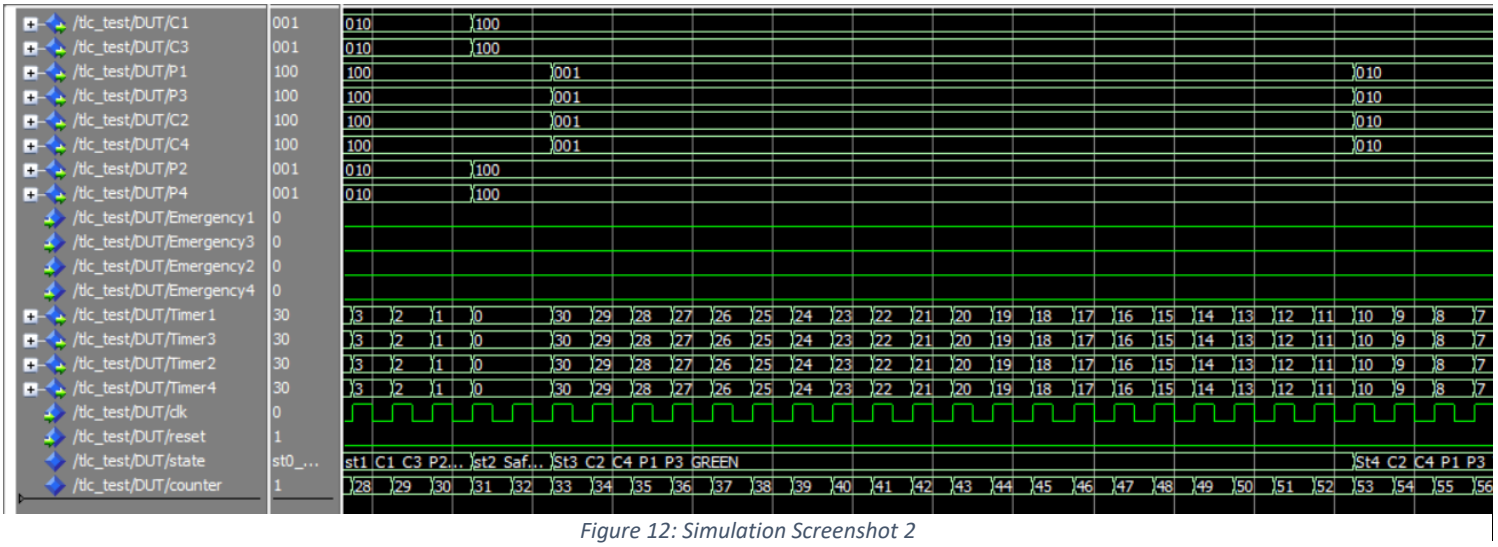

Figure 12: Simulation Screenshot 2

Once the countdown reaches 0 again, the present state is changed to State 5 (**st5_SafetyState**) that lasts for 2 seconds, reaching the end of this state cycle. Then, it returns back to the initial state (**st0_C1_C3_P2_P4_GREEN**) and repeats this state cycle again.

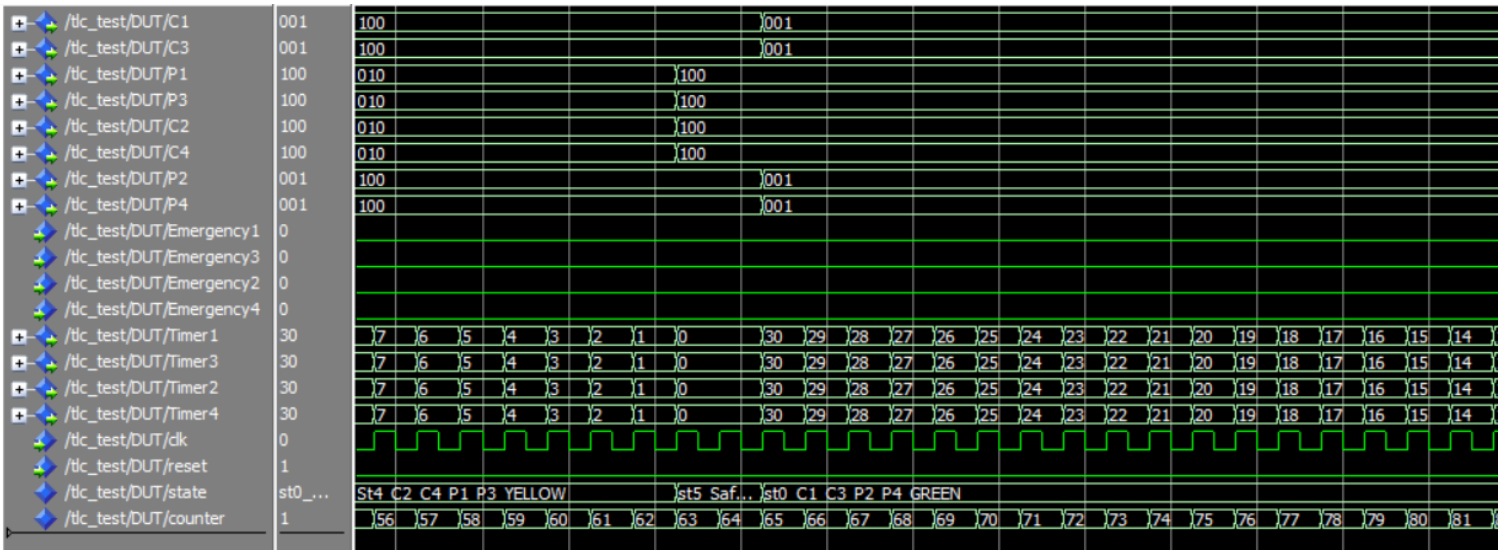

Figure 13: Simulation Screenshot 3

The normal flow of the state cycle is interrupted once any emergency inputs is triggered.

As shown below, once the **Emergency1** input is triggered HIGH (cars movement in Highway1 needs to be allowed), the present state is asynchronously changed to **st5_SafetyState** (to prevent sudden change of traffic) which will be then switched to **st0_C1_C3_P2_P4_GREEN**, allowing cars movement in Highway1.

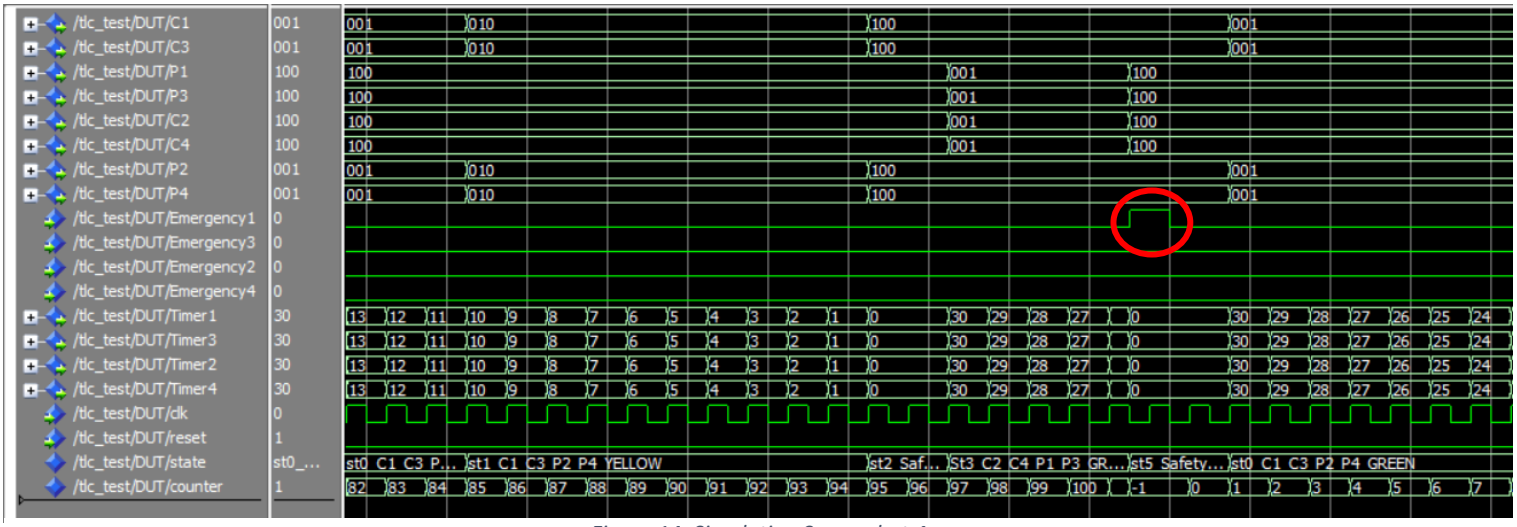

Figure 14: Simulation Screenshot 4

Similarly, once the **Emergency2** input is triggered HIGH (cars movement in Highway2 needs to be allowed), the present state is asynchronously changed to **st2_SafetyState** (to prevent sudden change of traffic) which will be then switched to **st3_C2_C4_P1_P3_GREEN**, allowing cars movement in Highway2.

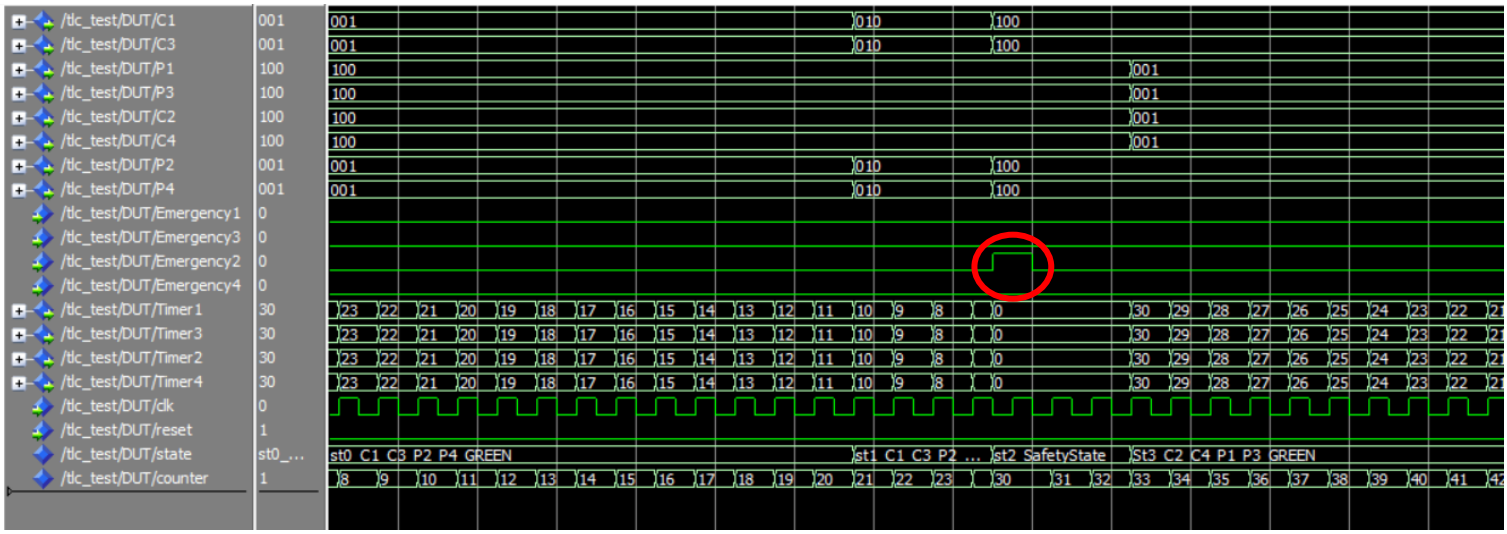

Figure 15: Simulation Screenshot 5

After that, the state cycle flows normally as described above unless any emergency signal is triggered.

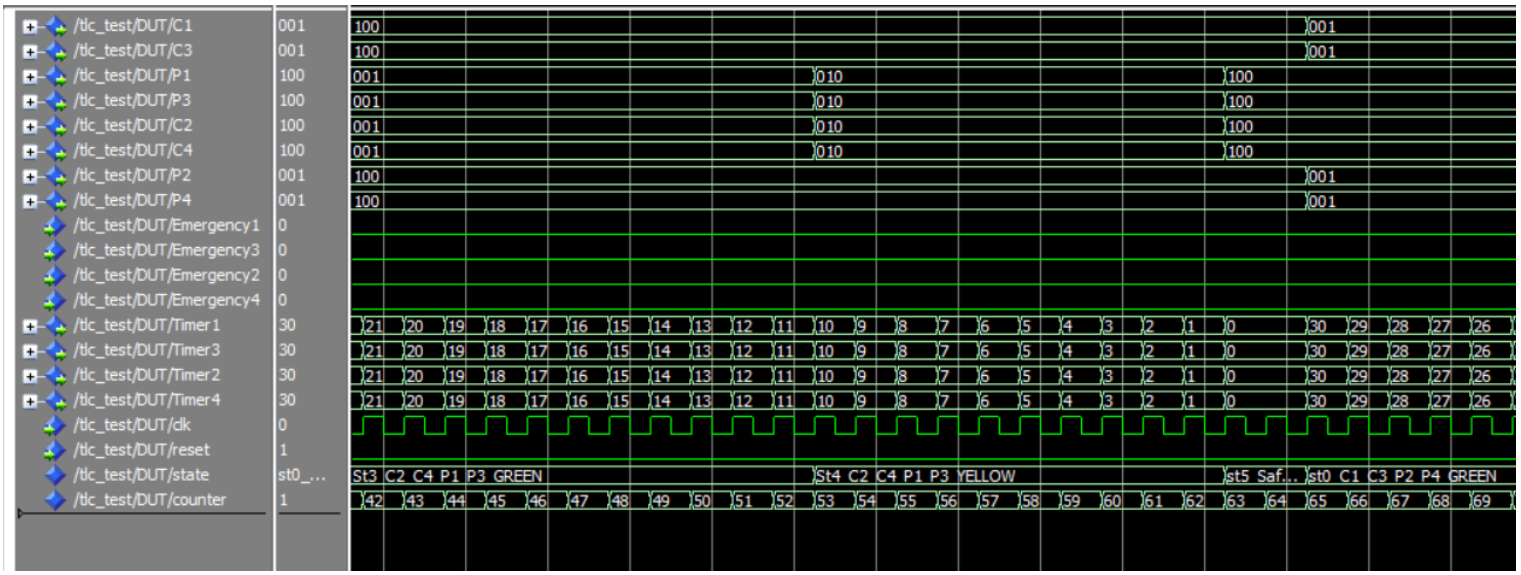

Figure 16: Simulation Screenshot 6