ELECTRONIC DESIGN AUTOMATION PROJECT

Project Documentation

FACULTY OF ENGINEERING
AIN SHAMS UNIVERSITY
International Credit Hours Engineering Programs (i.CHEP)

EDA Project

Submission date:

14/01/2022

Submitted to:

Dr. Hassan Ali Hassan

Eng. Ahmed Fawzy

Submitted by:

Omar Ashraf Mabrouk

Abdelrahman Mohamed Salah

Zakaria Sobhy Abd El-Salam

Mahmoud Mohamed Seddik

Hussein Ahmed Hassan

Group 2

Course Information

ASU Course Code CSE 312	***ASU Course Name*** Electronic Design Automation
UEL Course Code EG 7423	***UEL Course Name*** Engineering Systems
Semester Fall 2021	**Date of Submission** 14-10-2022

Participants Information

Full Name	**ASU ID**	**UEL ID**
Omar Ashraf Mabrouk	19P8102	U2140624
Abdelrahman Mohamed Salah	19P9131	U2140523
Zakaria Sobhy Abd El-Salam Soliman Madkour	19P2676	U2140654
Mahmoud Mohamed Seddik	19P3374	U2140584
Hussein Ahmed Hassan Selim	19P9614	U2140571

Table of Contents

0.0 Introduction 1
1.0. Traffic System 1 2
  1.1. Road Layout and System Description 2
  1.2. State Diagram 3
  1.3. Simulation results (Waveforms) 4
2.0. Traffic System 2 5
  2.1. Road layout and system description 5
  2.2. State Diagram 6
  3.3. Simulation results (Waveforms) 7
3.0. Traffic System 3 9
  3.1. Road layout and system description 9
  3.2. State diagram 11
  3.3. Simulation results (Waveforms) 12
Appendix 16
  Traffic Light System 1 Code 16
  Traffic Light System 1 Test Bench 20
  Traffic Light System 2 Code 22
  Traffic Light System 2 Test Bench 26
  Traffic Light System 3 Code 28
  Traffic Light System 3 Test Bench 32

0.0 Introduction

A traffic light system is an electronic device that assigns right of way at an intersection or crossing or street crossing by means of displaying the standard red, yellow and green colored indications. in addition, it also works in conjunction with pedestrian displays to assign pedestrian crossing right of way.

Main Modifications:

Code readability / safety delays / pedestrian button in system 1 / sensor for secondary road in system 2 / vertical and horizontal sensors for system 3

In our project we designed three traffic systems:

1. Traffic light system controlling a one-way road and a pedestrian crosswalk with a pedestrian button to toggle the pedestrians crossing on/off.

2. Traffic light system controlling an intersection between two roads, one is a busy road (highway) while the other is a not busy road (secondary road) with a sensor indicating an approach of a car in the secondary road. The system enables the car in the secondary road to either pass or safely merge into the highway.

3. Traffic light system controlling the intersection of a two-way directional roads (4-way junction) with horizontal and vertical sensors.

The project code is uploaded to the following drive with the full project and 6 files, 2 files for each system (Code – Simulation test). Also, the code is present in the appendix at the end of the project.

https://drive.google.com/drive/folders/1HQLN09_UOo7Wlgxc18qluJRUs3imloqM?usp=sharing

1.0. Traffic System 1

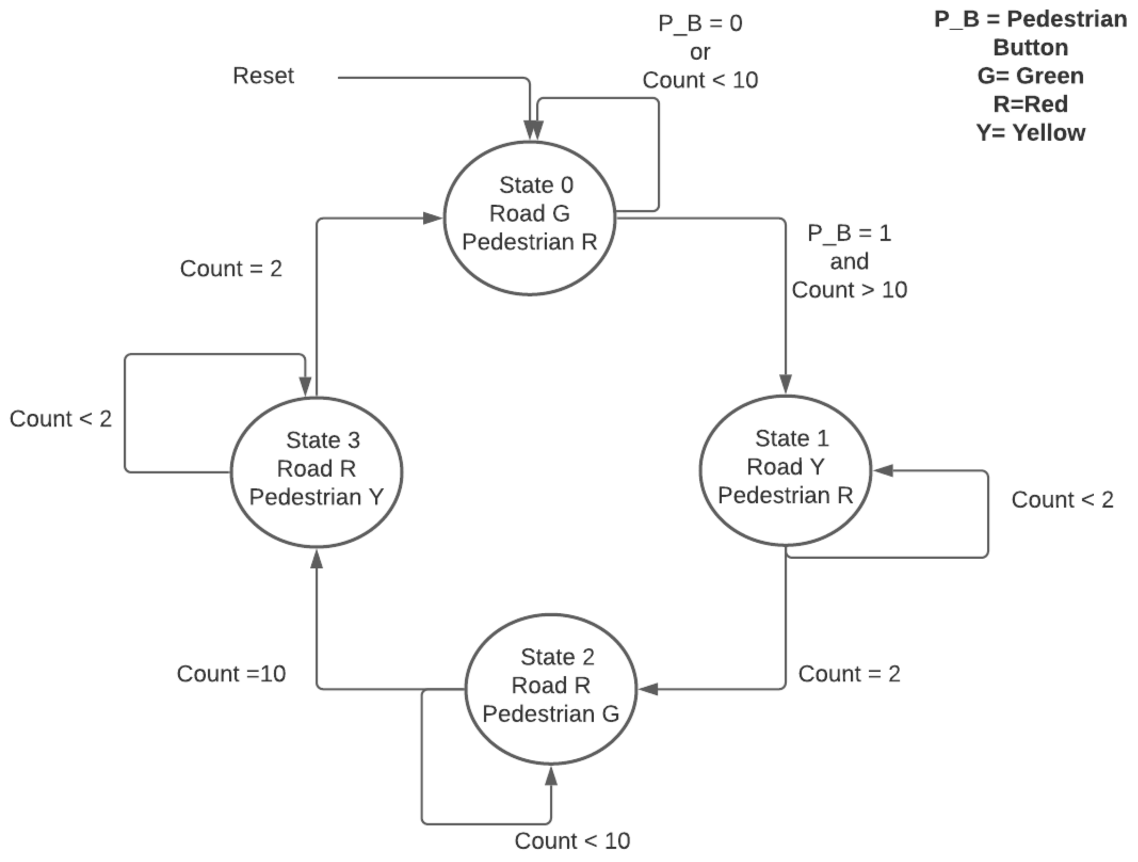
1.1. Road Layout and System Description

Since there exist times in the day where it is not necessary to regularly stop the cars for pedestrians to cross the street, we added a pedestrian button. When this button is clicked, the road is stopped, and pedestrians are allowed to cross street **on regular time intervals**. However, if the pedestrian button is disabled; the road won't stop until the pedestrian button is re-clicked.

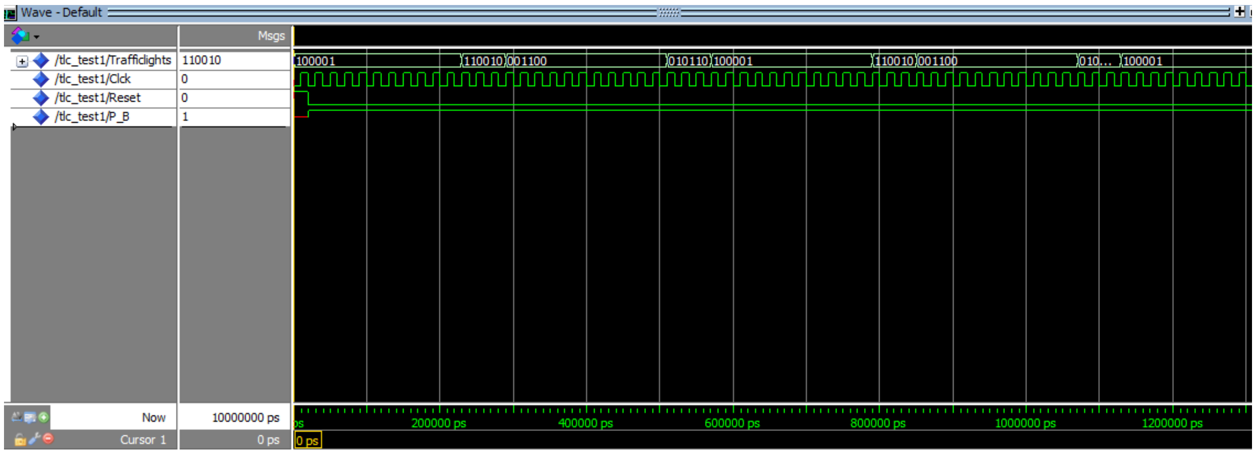
1.2. State Diagram

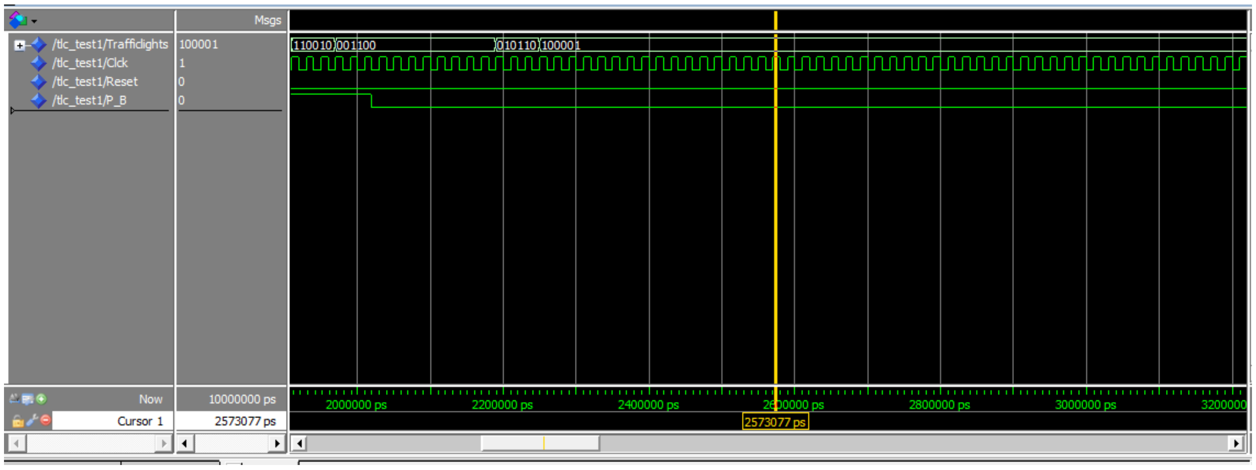
States:

0. Road traffic light is green, pedestrian traffic light is red, traffic light="100_001".
1. Road traffic light is yellow, pedestrian traffic light is red, traffic light= "110_010".
2. Road traffic light is red, pedestrian traffic light is green, traffic light="001_100".
3. Road traffic light is red, pedestrian traffic light is yellow, traffic light="010_110".



1.3. Simulation results (Waveforms)

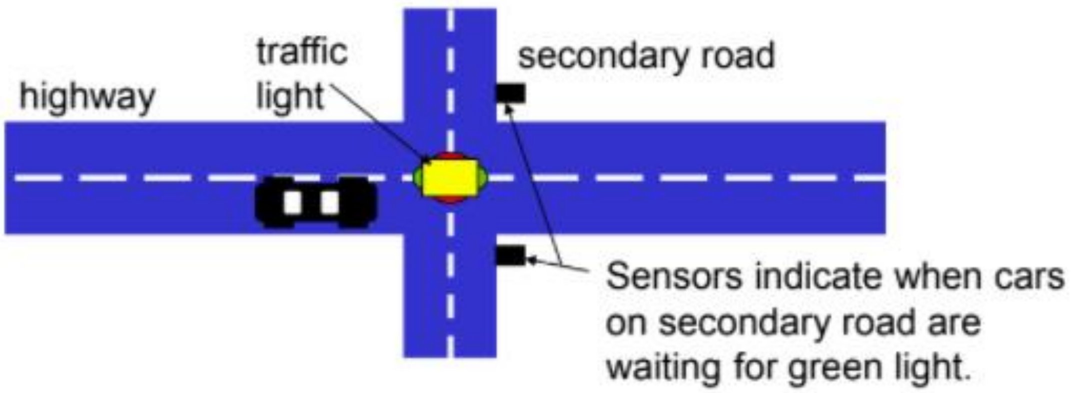




2.0. Traffic System 2

2.1. Road layout and system description

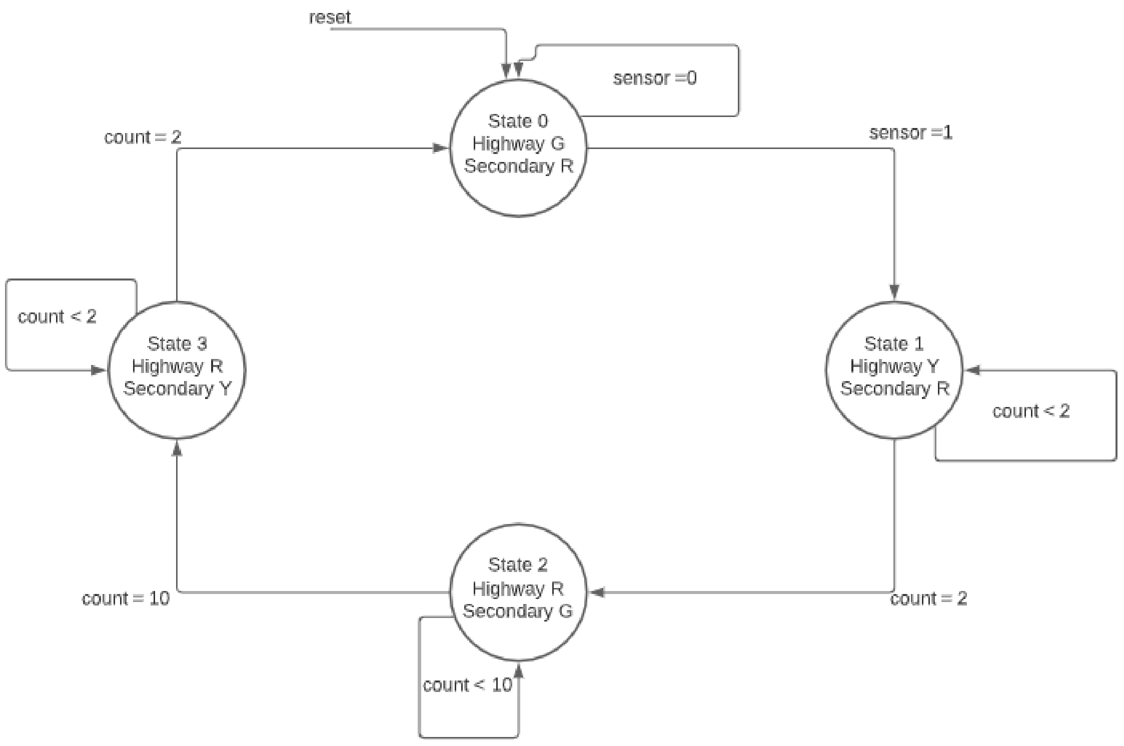
This system is used for an intersection between a highway (busy road) and a secondary road (not busy road) so that a car approaching from the secondary road can safely merge into the highway. Rarely a car would pass through the secondary road therefore the traffic flow of the highway remains moving unless the sensor on the secondary road notices a car approaching the traffic is stopped in the highway and the car on the secondary road passes then if there is no other car in the secondary road the highway resumes the traffic flow.


traffic light
highway
secondary road
Sensors indicate when cars on secondary road are waiting for green light.

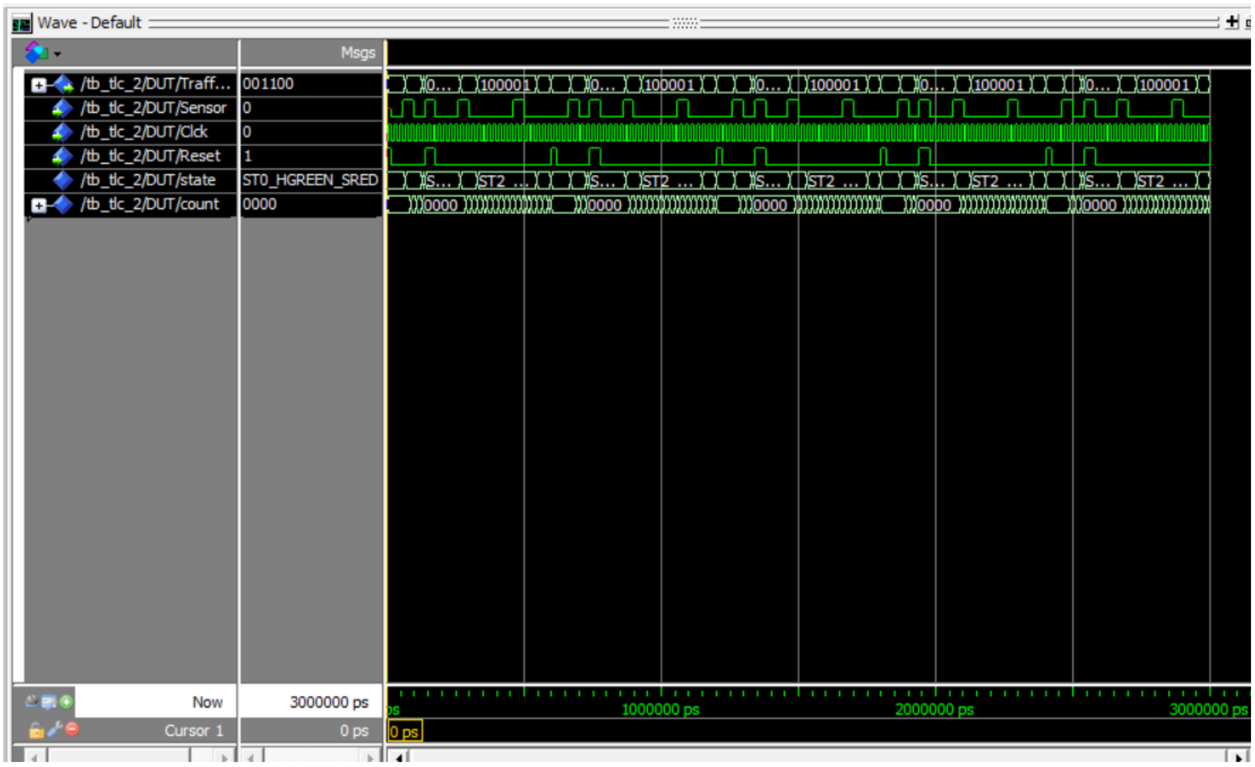
2.2. State Diagram

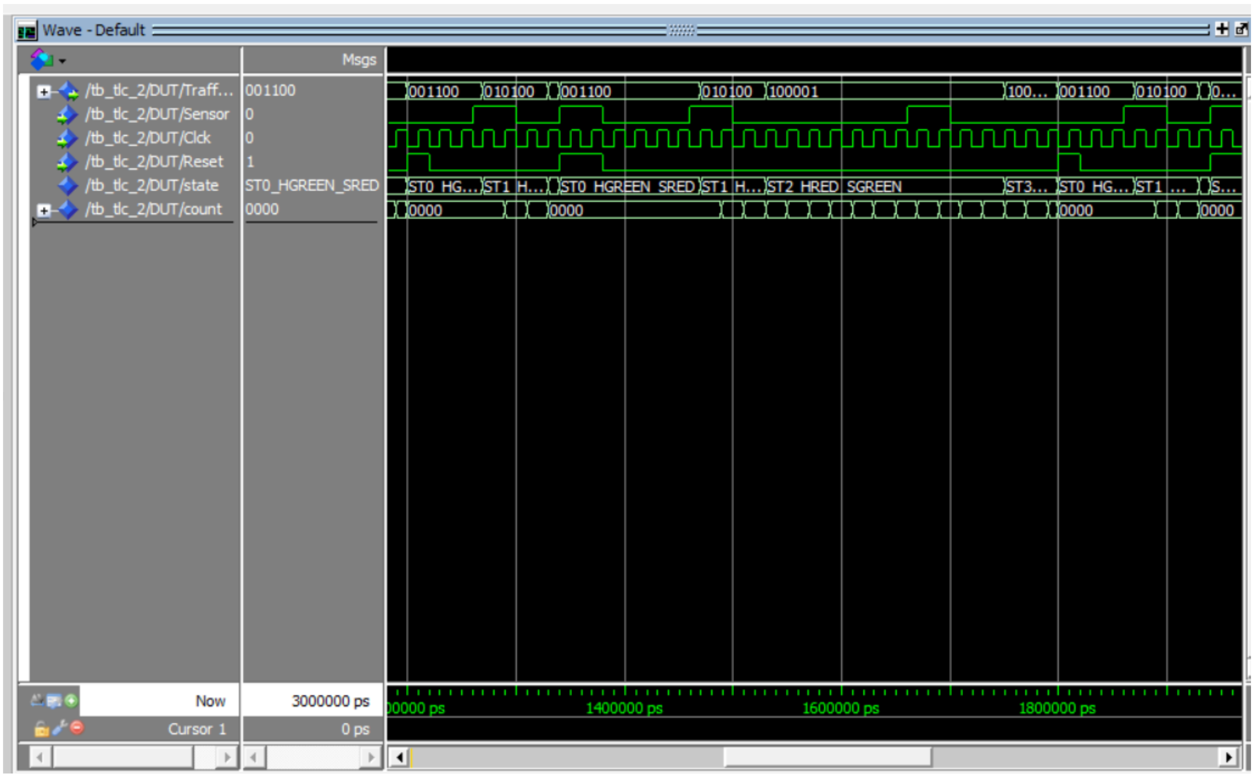
States:

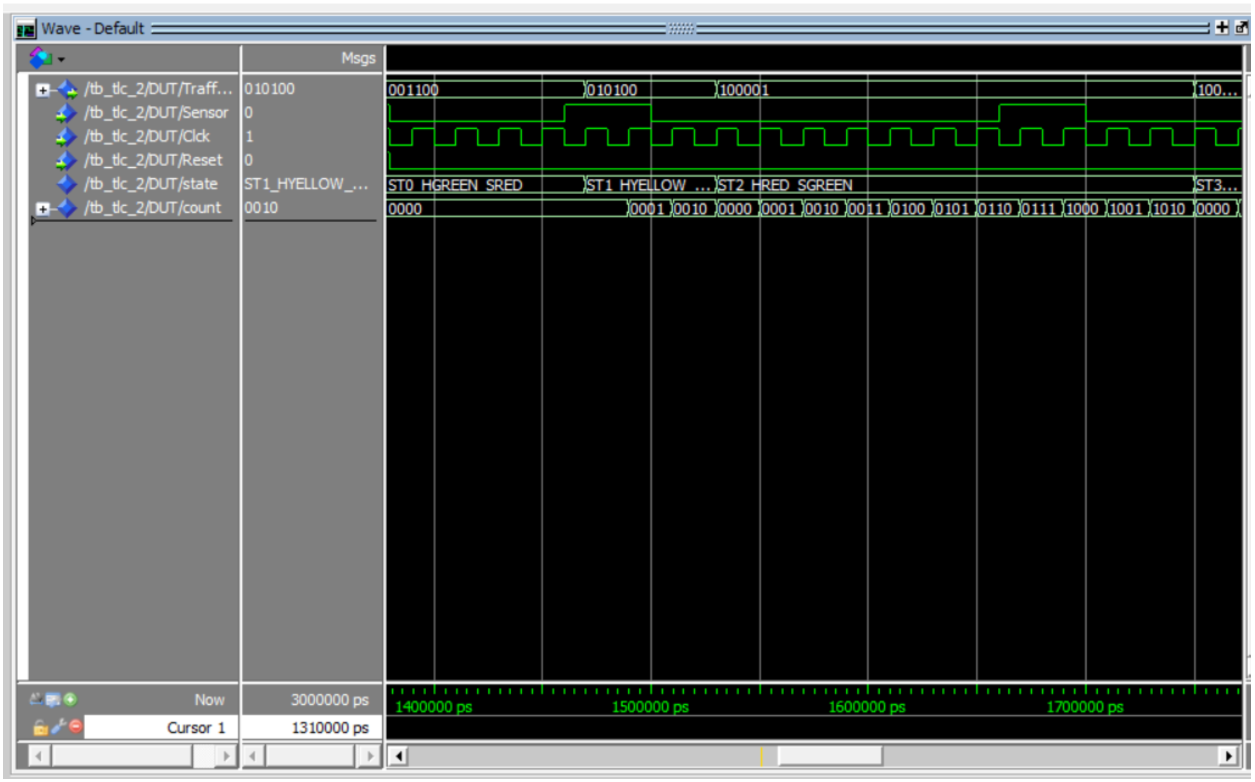
0. Highway traffic light is green, Secondary Road traffic light is red, traffic light="001_100".
1. Highway traffic light is yellow, Secondary Road traffic light is red, traffic light= "010_100".
2. Highway traffic light is red, Secondary Road traffic light is green, traffic light="100_001".
3. Highway traffic light is red, Secondary Road traffic light is yellow, traffic light="100_010".

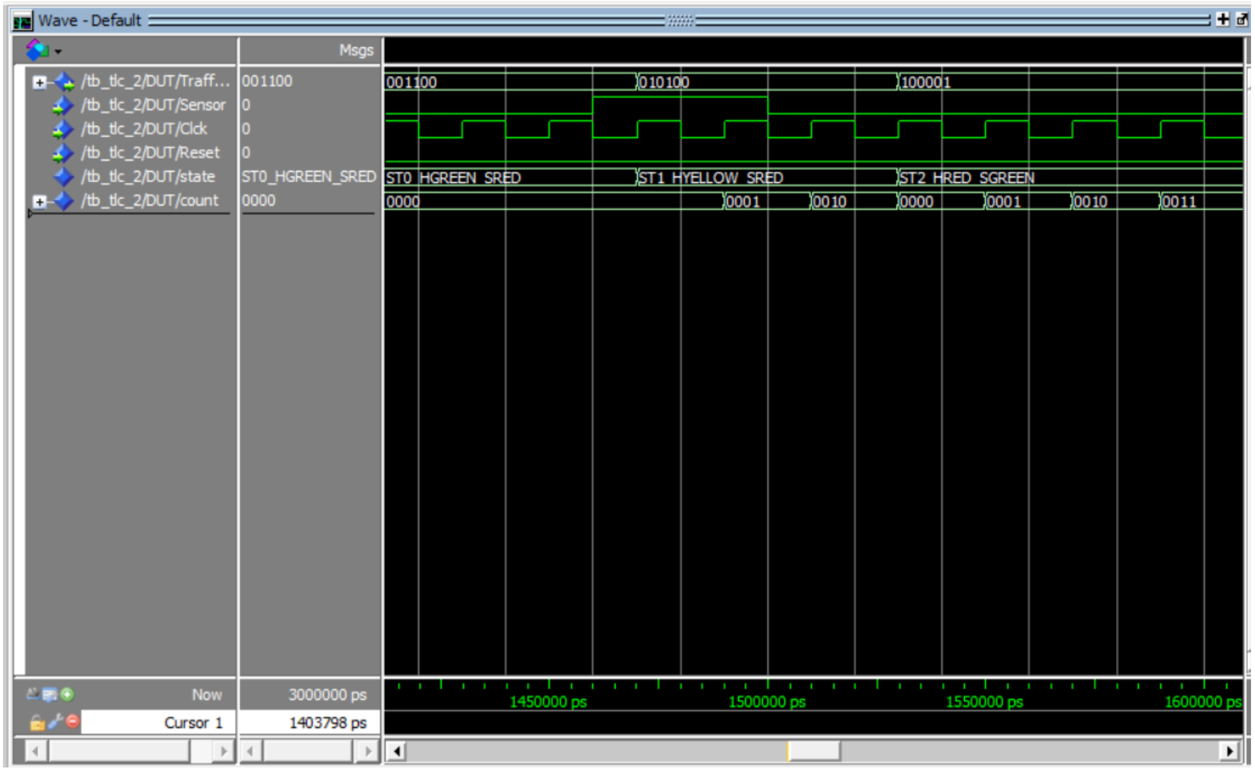


3.3. Simulation results (Waveforms)





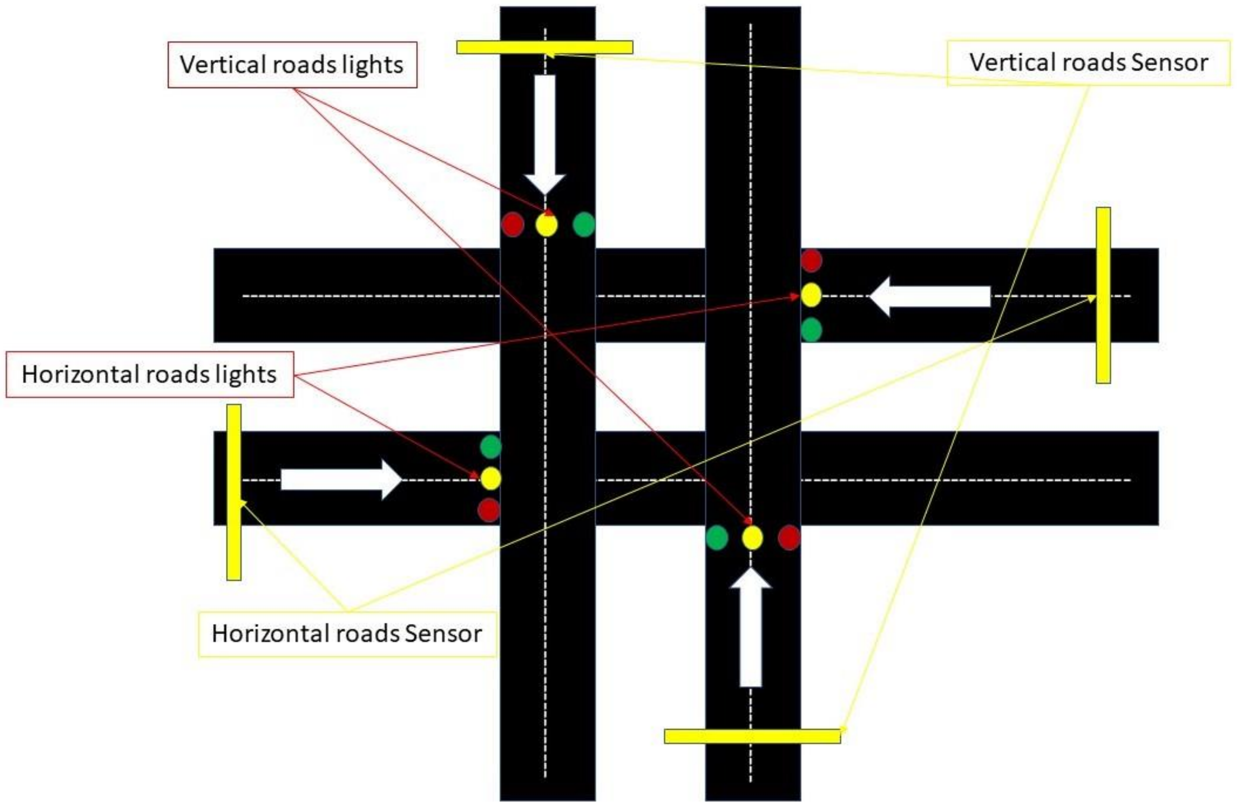

Wave - Default
Msgs
/tb_tlc_2/DUT/Traff... 010100
/tb_tlc_2/DUT/Sensor 0
/tb_tlc_2/DUT/Clck 1
/tb_tlc_2/DUT/Reset 0
/tb_tlc_2/DUT/state ST1_HYELLOW_...
/tb_tlc_2/DUT/count 0010
001100 010100 100001 100...
ST0_HGREEN_SRED ST1_HYELLOW_... ST2_HRED_SGREEN ST3...
0000 0001 0010 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 0000
Now 3000000 ps
Cursor 1 1310000 ps
1400000 ps
1500000 ps
1600000 ps
1700000 ps


Wave - Default
Msgs
/tb_tlc_2/DUT/Traff... 001100
/tb_tlc_2/DUT/Sensor 0
/tb_tlc_2/DUT/Clck 0
/tb_tlc_2/DUT/Reset 0
/tb_tlc_2/DUT/state ST0_HGREEN_SRED
/tb_tlc_2/DUT/count 0000
001100 010100 100001
ST0_HGREEN_SRED ST1_HYELLOW_SRED ST2_HRED_SGREEN
0000 0001 0010 0000 0001 0010 0011
Now 3000000 ps
Cursor 1 1403798 ps
1450000 ps
1500000 ps
1550000 ps
1600000 ps

3.0. Traffic System 3

3.1. Road layout and system description

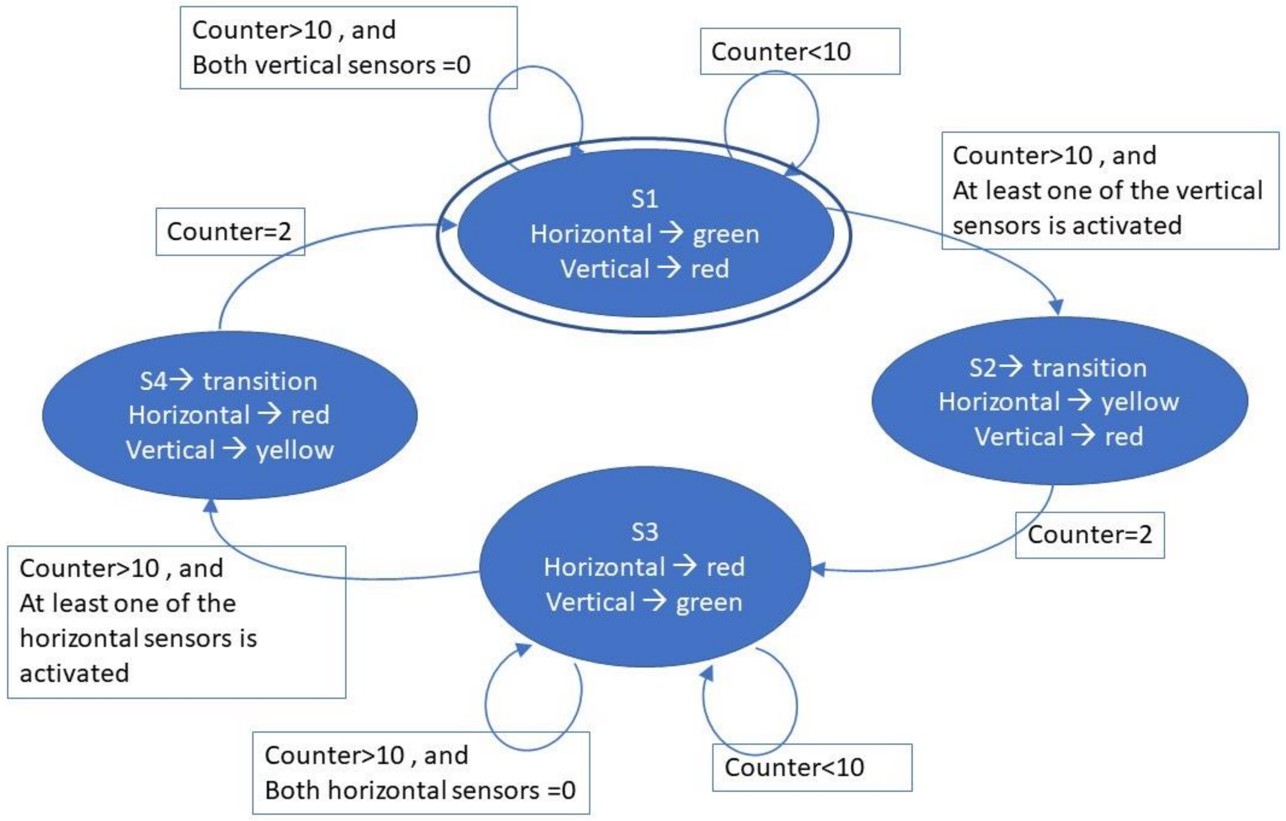
This traffic control system controls the intersection of a two-way directional roads as depicted in the figure below. Each road has its own traffic lights and its own sensor. In this design we intend to allow the two parallel roads to flow with each other and stop with each other. In other words, the traffic lights of each two parallel roads will have the same colors at the same instant. Of course, each of them will be assigned different output port but in the back-end logic of the system we can treat the lights of parallel roads as one signal.

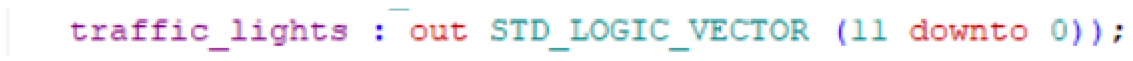


The system will work as follows, whenever the light converts to green for a certain direction it has to wait for a certain amount of time say 100 counter clicks before it may consider shifting to another state. If the direction has consumed its time then we check the sensors of the two other roads – which are perpendicular on there two roads – if one or both of them are activated this means that the other two roads are full in this case we convert the state to make the lights of the other two roads green. But if the sensors arent asserted then we continue in the current state as this means that the other roads arent full yet.

To easily understand how the system works imagine having two buckets, you are currently emptying a bucket after some time you check if the other bucket is full. If the other bucket isnt full, you continue emptying the one in your hands. If the other bucket is full then you leave the one in your hands and start emptying the other. The next state diagram will help understand how the sysytem works.

3.2. State diagram



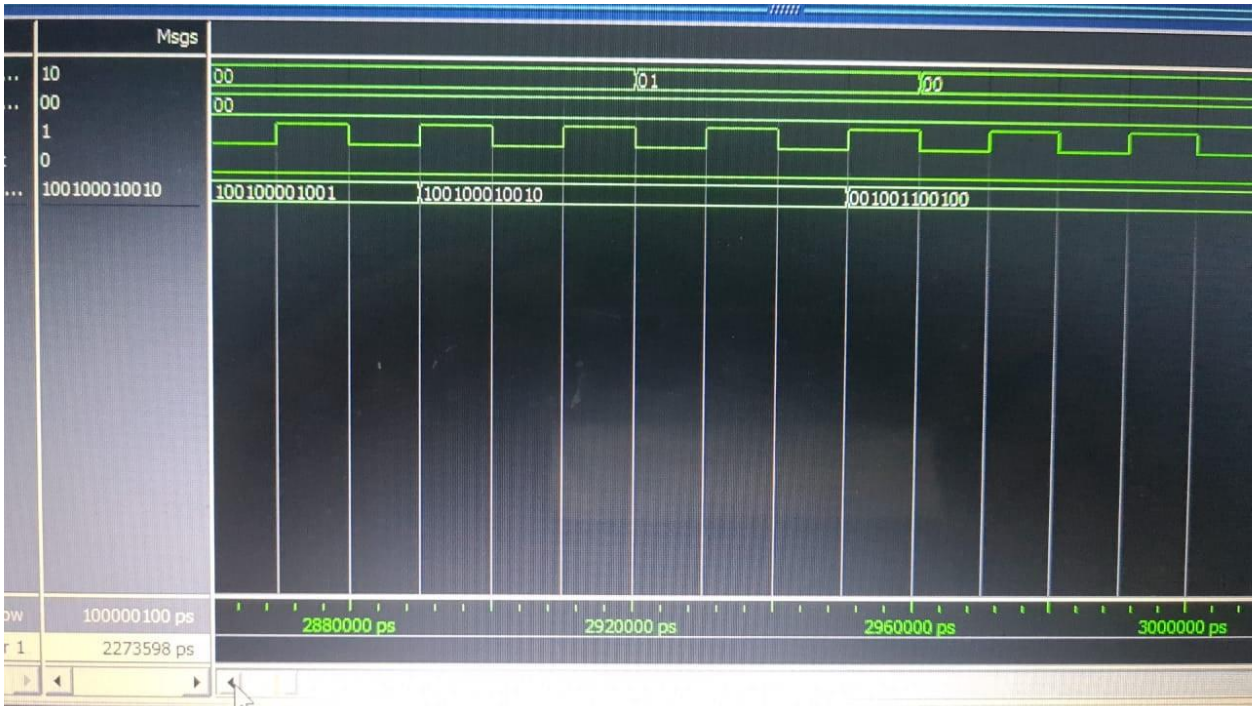


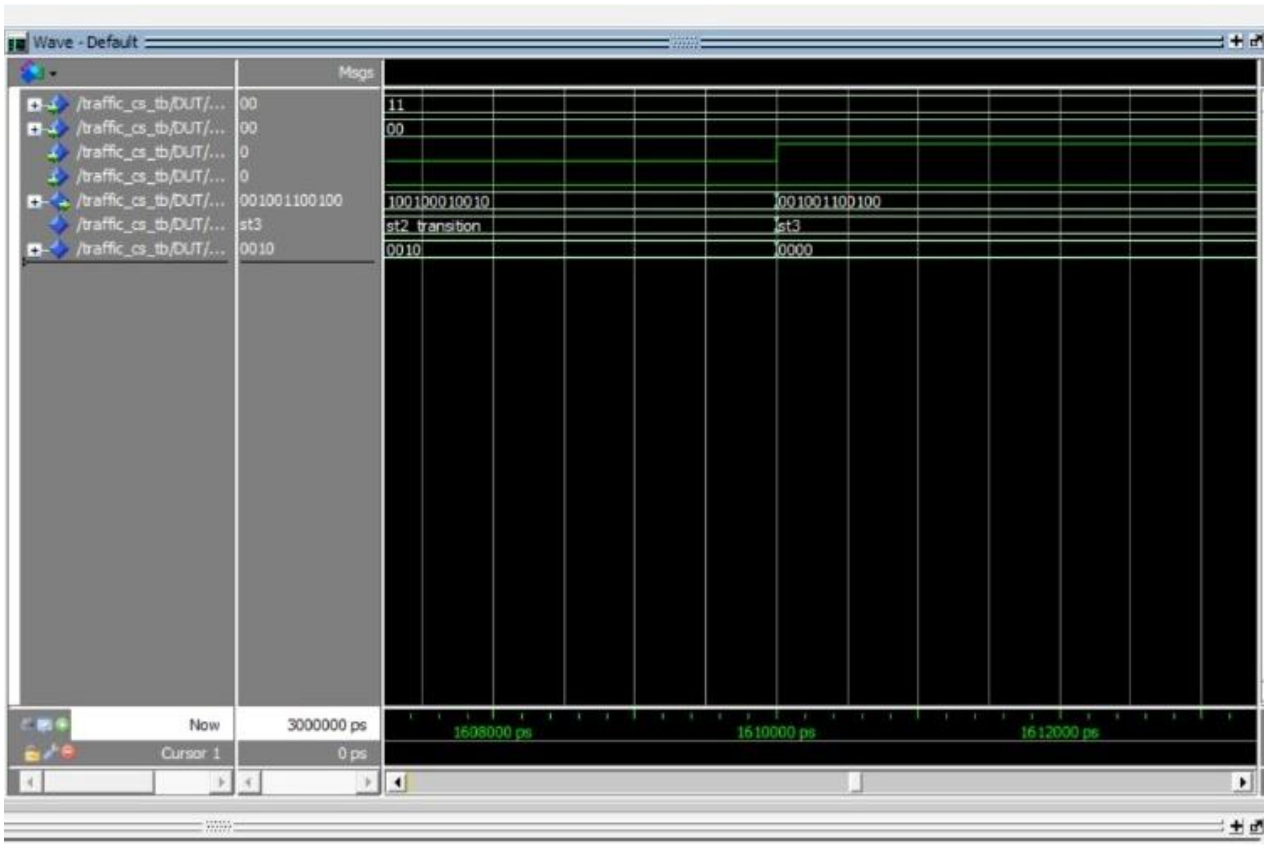
```
traffic_lights : out STD_LOGIC_VECTOR (11 downto 0));
```

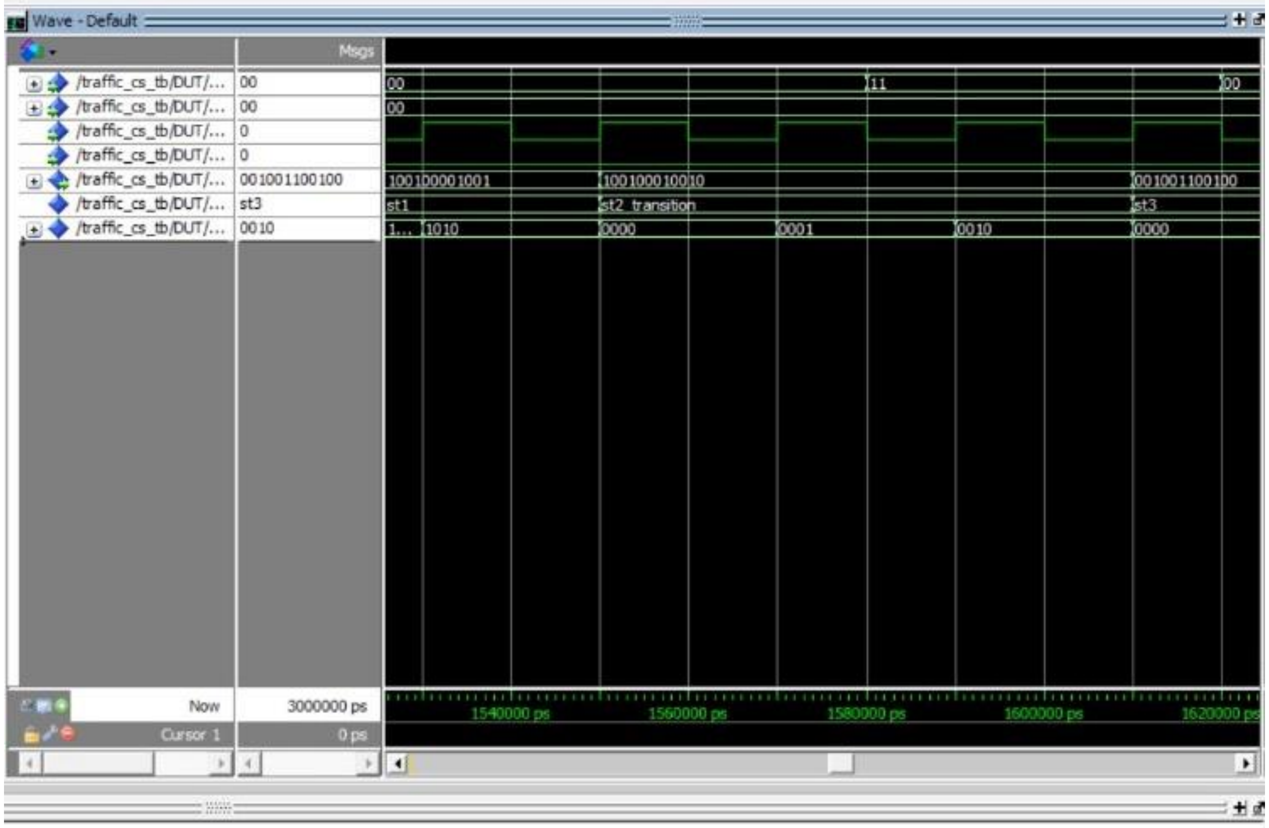
The output is in the form of vector having 12 bits it is divided as follow

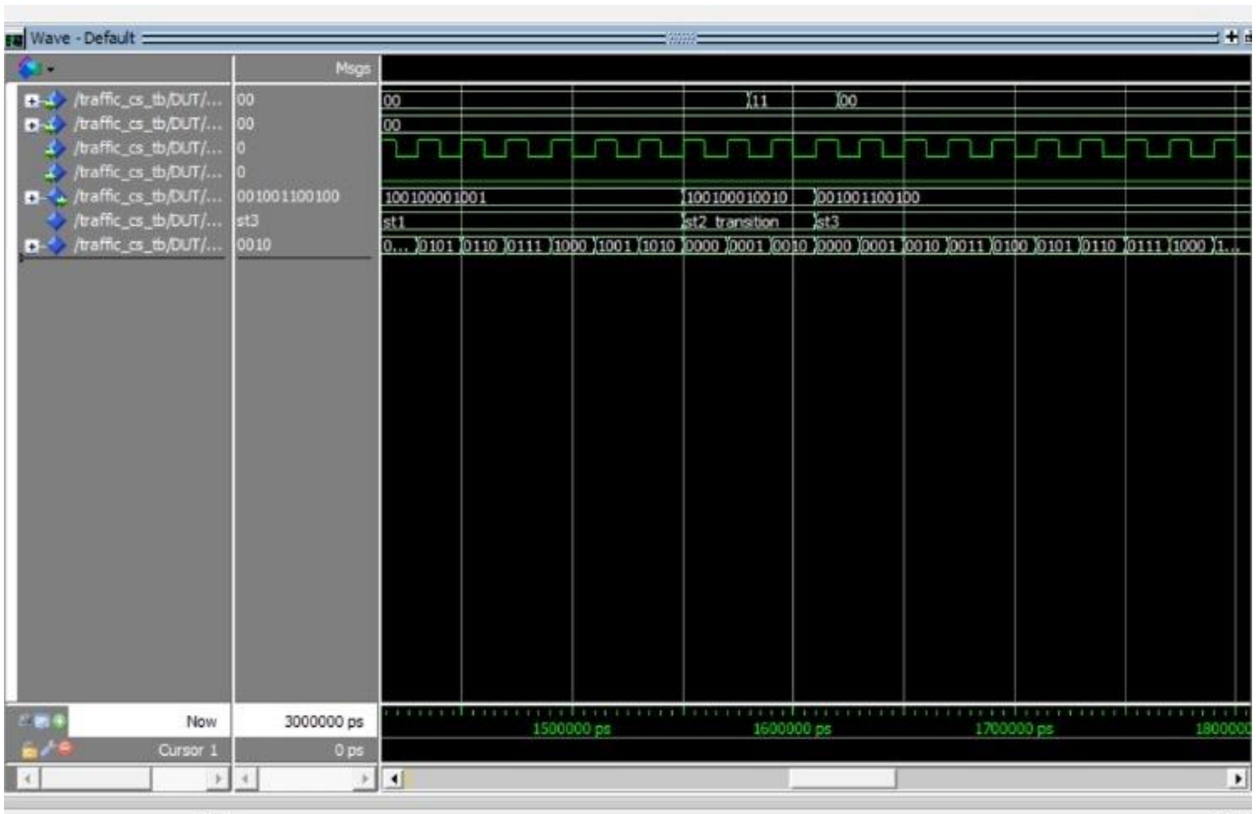
Vertical road 2			Vertical road 1			Horizontal road 2			Horizontal road 1		
11	10	9	8	7	6	5	4	3	2	1	0
RED	YELLOW	GREEN	RED	YELLOW	GREEN	RED	YELLOW	GREEN	RED	YELLOW	GREEN

3.3. Simulation results (Waveforms)

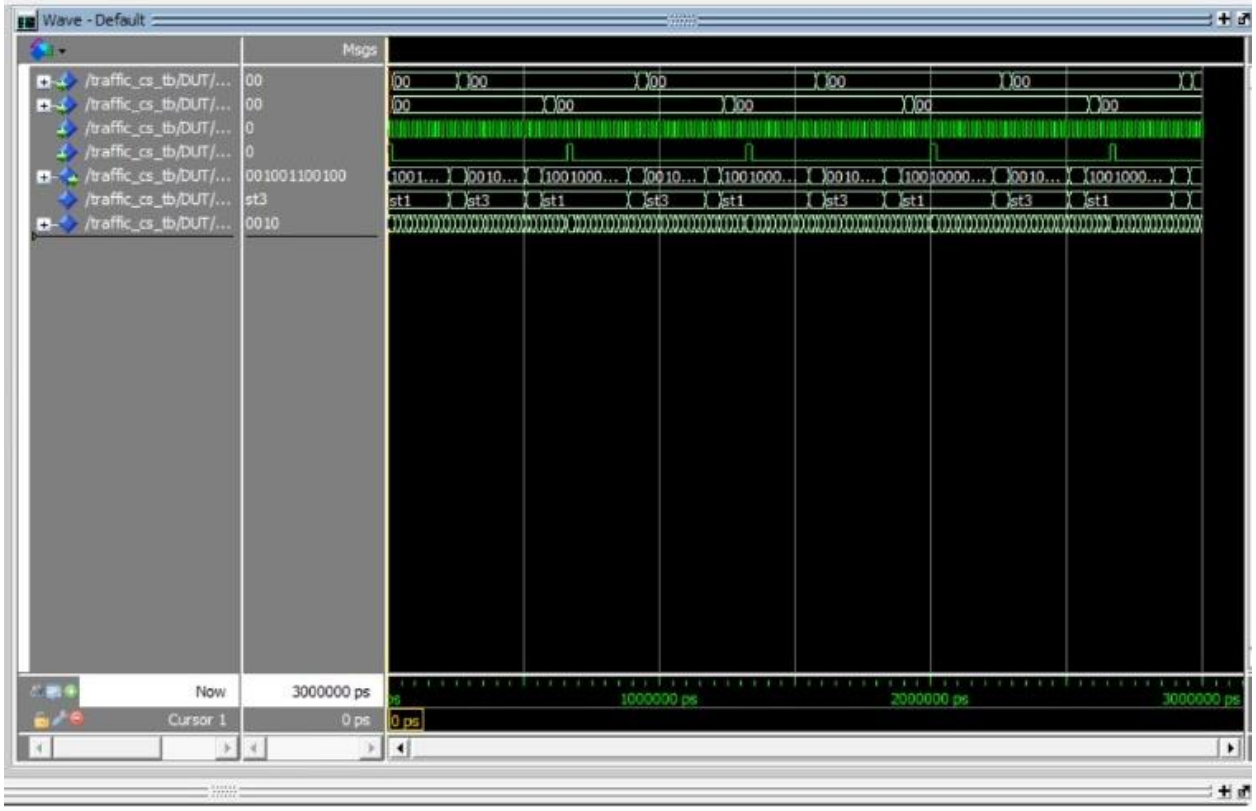



Wave - Default
Msgs
/traffic_cs_tb/DUT/... 00
/traffic_cs_tb/DUT/... 00
/traffic_cs_tb/DUT/... 0
/traffic_cs_tb/DUT/... 0
/traffic_cs_tb/DUT/... 001001100100
/traffic_cs_tb/DUT/... st3
/traffic_cs_tb/DUT/... 0010
11
00
100100010010
001001100100
st2_transition
st3
0010
0000
Now 3000000 ps
Cursor 1 0 ps
1608000 ps
1610000 ps
1612000 ps

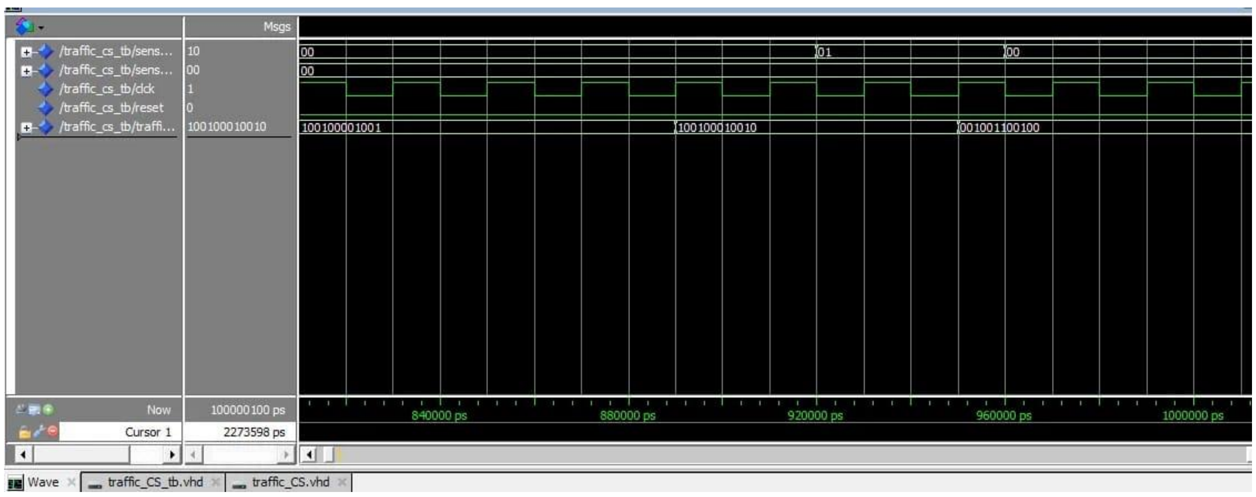

Wave - Default
Msgs
/traffic_cs_tb/DUT/... 00
/traffic_cs_tb/DUT/... 00
/traffic_cs_tb/DUT/... 0
/traffic_cs_tb/DUT/... 0
/traffic_cs_tb/DUT/... 001001100100
/traffic_cs_tb/DUT/... st3
/traffic_cs_tb/DUT/... 0010
00
11
00
00
100100001001
100100010010
001001100100
st1
st2_transition
st3
1...
1010
0000
0001
0010
0000
Now 3000000 ps
Cursor 1 0 ps
1540000 ps
1560000 ps
1580000 ps
1600000 ps
1620000 ps



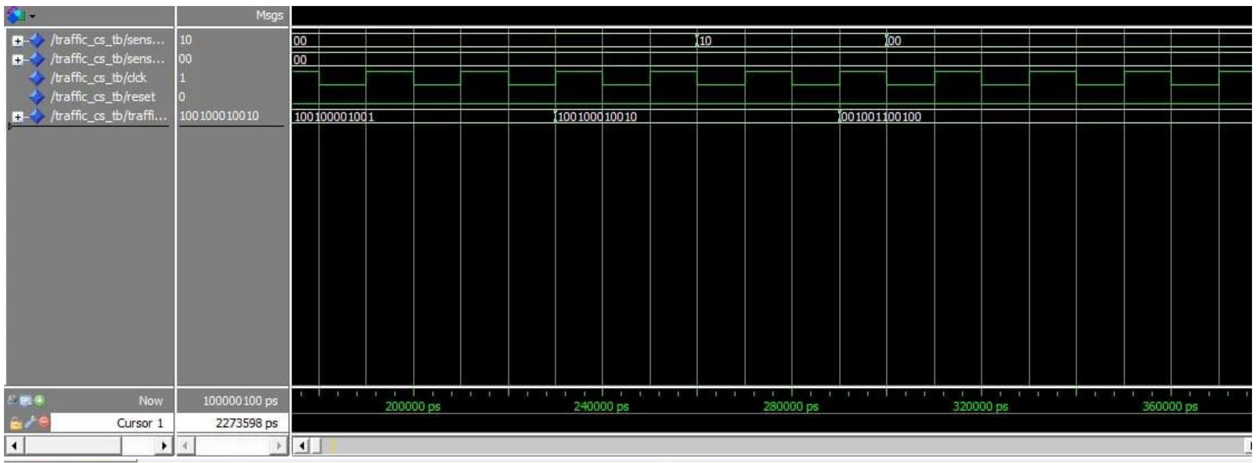
```
Wave - Default
                              Msgs
/traffic_cs_tb/DUT/...   00             00  11  00
/traffic_cs_tb/DUT/...   00             00
/traffic_cs_tb/DUT/...   0
/traffic_cs_tb/DUT/...   0
/traffic_cs_tb/DUT/...   001001100100   100100001001  100100010010  001001100100
/traffic_cs_tb/DUT/...   st3            st1  st2_transition  st3
/traffic_cs_tb/DUT/...   0010           0... 0101 0110 0111 1000 1001 1010 0000 0001 0010 0000 0001 0010 0011 0100 0101 0110 0111 1000 1...

Now       3000000 ps        1500000 ps   1600000 ps   1700000 ps   180000
Cursor 1  0 ps
```



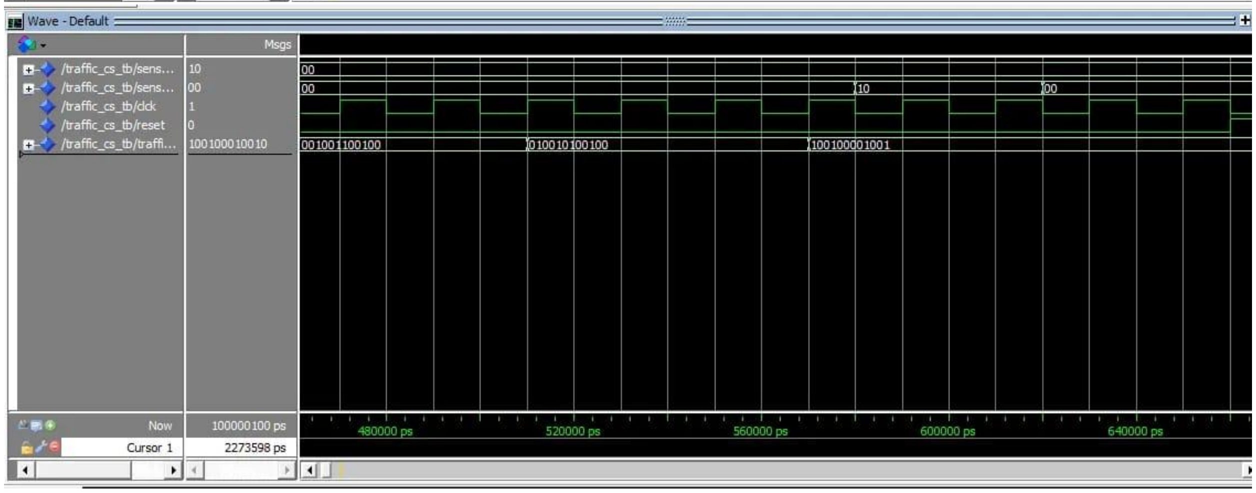
```
Wave - Default
                              Msgs
/traffic_cs_tb/DUT/...   00             00  00  00  00  00
/traffic_cs_tb/DUT/...   00             00  00  00  00  00
/traffic_cs_tb/DUT/...   0
/traffic_cs_tb/DUT/...   0
/traffic_cs_tb/DUT/...   001001100100   1001...  0010...  1001000...  0010...  1001000...  0010...  10010000...  0010...  1001000...
/traffic_cs_tb/DUT/...   st3            st1  st3  st1  st3  st1  st3  st1  st3  st1
/traffic_cs_tb/DUT/...   0010

Now       3000000 ps    ps   1000000 ps   2000000 ps   3000000 ps
Cursor 1  0 ps          0 ps
```



```
/traffic_cs_tb/sens...   10            00            01            00
/traffic_cs_tb/sens...   00            00
/traffic_cs_tb/clk       1
/traffic_cs_tb/reset     0
/traffic_cs_tb/traffi... 100100010010  100100001001  100100010010  001001100100
Now       100000100 ps
Cursor 1  2273598 ps
840000 ps   880000 ps   920000 ps   960000 ps   1000000 ps
```



```
/traffic_cs_tb/sens...   10            00            10            00
/traffic_cs_tb/sens...   00            00
/traffic_cs_tb/clk       1
/traffic_cs_tb/reset     0
/traffic_cs_tb/traffi... 100100010010  100100001001  100100010010  001001100100
Now       100000100 ps
Cursor 1  2273598 ps
200000 ps   240000 ps   280000 ps   320000 ps   360000 ps
```



```
/traffic_cs_tb/sens...   10            00
/traffic_cs_tb/sens...   00            00            10            00
/traffic_cs_tb/clk       1
/traffic_cs_tb/reset     0
/traffic_cs_tb/traffi... 100100010010  001001100100  010010100100  100100001001
Now       100000100 ps
Cursor 1  2273598 ps
480000 ps   520000 ps   560000 ps   600000 ps   640000 ps
```

Appendix

Traffic Light System 1 Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity TLC_1 is Port (

Trafficlights:out STD_LOGIC_Vector (5 downto 0);
Clck : in STD_LOGIC;
Reset : in STD_LOGIC;
P_B : in STD_LOGIC);

end TLC_1;



architecture Behavioral of TLC_1 is

type state_type is (st0_G1_R2, st1_Y1_R2, st2_R1_G2, st3_R1_Y2);
signal state: state_type;
signal count : std_logic_vector (3 downto 0);
constant sec10 : std_logic_vector ( 3 downto 0) := "1010";
constant sec2 : std_logic_vector (3 downto 0 ) := "0010";
constant sec16: std_logic_vector (3 downto 0 ) := "1111";

begin
```

```
process (Clck,Reset) begin



if Reset='1' then
state <= st0_G1_R2; --reset to initial state
count <= X"0"; -- reset counter



elsif Clck' event and Clck = '1' then --


case (state) is
when st0_G1_R2 =>


if count < sec10 then
state <= st0_G1_R2;
count <= count + 1;


else
if (P_B='0') then
state <= st0_G1_R2;


else
state <= st1_Y1_R2;
count <= X"0";
end if;
end if;
```

```
when st1_Y1_R2 =>
if count < sec2 then
state <= st1_Y1_R2;
count <= count + 1;

else
state <= st2_R1_G2;
count <= X"0";
end if;

when st2_R1_G2 =>
if count < sec10 then
state <= st2_R1_G2;
count <= count + 1;

else
state <= st3_R1_Y2;
count <= X"0";
end if;

when st3_R1_Y2 =>
if count < sec2 then
state <= st3_R1_Y2;
count <= count + 1;

else
state <=st0_G1_R2;
count <= X"0";
```

```
end if;

when others =>
state <= st0_G1_R2;
end case;
end if;
end process;

OUTPUT_DECODE: process (state)
begin
case state is
when st0_G1_R2 => Trafficlights <= "100001";
when st1_Y1_R2 => Trafficlights <= "110010";
when st2_R1_G2 => Trafficlights <= "001100";
when st3_R1_Y2 => Trafficlights <= "010110";
when others => Trafficlights <= "100001";
end case;
end process;
end Behavioral;
```

Traffic Light System 1 Test Bench

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

entity TB_TLC_1 is
end entity;

architecture tb of TB_TLC_1 is

signal Trafficlights : std_logic_vector (5 downto 0);
signal Clck,Reset,P_B: std_logic;

begin
DUT : ENTITY work.TLC_1

PORT MAP(Trafficlights=>Trafficlights,Clck=>Clck,Reset=>Reset,P_B=>P_B);

Clock : process
begin
Clck <= '0';
wait for 10 ns;
Clck <= '1';
wait for 10 ns;
end process;

stimulis : process
begin
```

```
report("Starting simulation");

Reset <= '1';
wait for 20 ns;
Reset <= '0';
P_B <= '1';
wait for 2000 ns;
Reset <= '0';
P_B <= '0';
wait for 2000 ns;

report("End simulation");
end process;
end architecture;
```

Traffic Light System 2 Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity TLC_2 is
 Port (
 Traffic_lights:out STD_LOGIC_Vector (5 downto 0);--(RED-YELLOW-GREEN//RED-
YELLOW-GREEN) HIGHWAY/SECONDARY
 Sensor:in STD_LOGIC;
 Clck : in STD_LOGIC;
 Reset : in STD_LOGIC);
end TLC_2;

architecture Behavioral of TLC_2 is
 type state_type is (ST0_HGREEN_SRED, ST1_HYELLOW_SRED, ST2_HRED_SGREEN,
ST3_HRED_SYELLOW);
 signal state: state_type;
 signal count : std_logic_vector (3 downto 0);
 constant sec10 : std_logic_vector ( 3 downto 0) := "1010";
 constant sec2 : std_logic_vector (3 downto 0 ) := "0010";
 constant sec16: std_logic_vector (3 downto 0 ) := "1111";

begin
  process (clck,reset)
  begin
    if Reset='1'  then
     state  <=  ST0_HGREEN_SRED;     --reset to initial state
     count <=  X"0";     --  reset counter elsif Clck'
    elsif Clck'  event and Clck =  '1'  then

        case  (state) is

 when ST0_HGREEN_SRED => --when green light on highway and red light on secondary road
  if(sensor = '1') then -- if vehicle is detected on secondary road by sensors
  state <= ST1_HYELLOW_SRED;  -- High way turns to yellow light and secondary stays red
 else
  state <= ST0_HGREEN_SRED; -- otherwise, no change in state
 end if;

 when ST1_HYELLOW_SRED =>  -- When yellow light on highway and red light on secondary
road
 if count < sec2 then -- delay not done state, remains the same
 state <= ST1_HYELLOW_SRED;
 count <= count + 1; -- increment counter
 else
 state <= ST2_HRED_SGREEN; -- otherwise, highway turns red and secondary road turns green
 count <= X"0";   -- reset count
 end if;

 when ST2_HRED_SGREEN => -- when red light on highway and green light on secondary
road
 if count < sec10 then   -- if secondary road time is less than 10 cycles no change in state
 state <= ST2_HRED_SGREEN;
 count <= count + 1;     -- increment counter
 else
```

```
 state <= ST3_HRED_SYELLOW; -- otherwise highway stays red and secondary road turns
yellow
 count <= X"0"; -- reset count
 end if;

 when ST3_HRED_SYELLOW => -- when highway is red and secondary road is yellow
 if count < sec2 then -- delay not finished, state remains the same
 state <= ST3_HRED_SYELLOW;
 count <= count + 1; --increment counter
 else
 state <=ST0_HGREEN_SRED; -- otherwise highway turns green and secondary road turns red
 count <= X"0"; -- reset count
 end if;

 when others =>
 state <= ST0_HGREEN_SRED;
 end case;
 end if;
 end process;


 OUTPUT_DECODE: process (state)
 begin
 case state is
      when ST0_HGREEN_SRED  => traffic_lights <= b"001_100";  -- highway green   and
secondary red
      when ST1_HYELLOW_SRED => traffic_lights <= b"010_100";  -- highway yellow  and
secondary red
      when ST2_HRED_SGREEN  => traffic_lights <= b"100_001";  -- highway red     and
secondary green
```

```
      when ST3_HRED_SYELLOW => traffic_lights <= b"100_010";  -- highway red     and
secondary yellow

      when others            => traffic_lights <= b"100_100"; -- set both roads to red to avoid
accidents

     end case;

 end process;

end Behavioral;
```

Traffic Light System 2 Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY TB_TLC_2 IS
END TB_TLC_2;

ARCHITECTURE tb of TB_TLC_2 is

 signal Traffic_lights :  STD_LOGIC_VECTOR (5 downto 0);
 signal Sensor :  STD_LOGIC;
 signal clck :   STD_LOGIC;
 signal reset : STD_LOGIC;

begin
  DUT : ENTITY work.TLC_2

  PORT MAP(sensor=>sensor,
           traffic_lights=>traffic_lights,
           clck=>clck,
           reset=>reset)  ;

Clock : process
begin
clck <= '0';
wait for 10 ns;
clck <= '1';
```

```
wait for 10 ns;
end process;
-- clck has a clock cycle of 20 ns



stimulis : process
begin
report("Starting simulation");

   reset <= '1';                    wait for 20 ns;
   reset <= '0';   sensor <= '0';  wait for 40 ns;
   reset <= '0';   sensor <= '1';  wait for 40 ns;
   reset <= '0';   sensor <= '0';  wait for 40 ns;
   reset <= '1';   sensor <= '1';  wait for 40 ns;
   reset <= '0';   sensor <= '0';  wait for 40 ns;
   reset <= '0';   sensor <= '0';  wait for 40 ns;
   reset <= '0';   sensor <= '1';  wait for 40 ns;
   reset <= '0';   sensor <= '0';  wait for 40 ns;
   reset <= '0';   sensor <= '0';  wait for 40 ns;
   reset <= '0';   sensor <= '0';  wait for 40 ns;
   reset <= '0';   sensor <= '0';  wait for 40 ns;
   reset <= '0';   sensor <= '1';  wait for 40 ns;
   reset <= '0';   sensor <= '0';  wait for 40 ns;
   reset <= '0';   sensor <= '0';  wait for 40 ns;
 wait for 20 ns;
report("End simulation");
end process;
end architecture;
```

Traffic Light System 3 Code

```
library  IEEE;
use  IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity  TLC_3  is
 Port (
  sensors_vertical : in STD_LOGIC_VECTOR(1 downto 0);
  sensors_horizontal : in STD_LOGIC_VECTOR(1 downto 0);
  clck : in  STD_LOGIC;
  reset :  in  STD_LOGIC;
  traffic_lights : out STD_LOGIC_VECTOR (11 downto 0));
end TLC_3;



architecture  Behavioral of TLC_3  is
  type  state_type  is ( st1 , st2_transition , st3 , st4_transition );

  signal state: state_type;
  signal count :  std_logic_vector (3 downto 0);
  constant sec10 : std_logic_vector (  3 downto 0)  := "1010"; -- minimum waiting time for one
road to be green
  constant sec2 : std_logic_vector  (3 downto 0 )  :=  "0010"; -- transition time

begin

  process (clck,reset)
```

```
begin
  if Reset='1'  then
   state  <=  st1;     --reset to initial state
   count <=  X"0";     --  reset counter elsif Clck'
  elsif Clck'  event and Clck =  '1'  then

      case  (state) is

        when st1 =>

         if count < sec10 then
           state  <=  st1;
           count <=  count  +  1;
         elsif   (count > sec10) and (sensors_vertical = "00") then
           state  <=  st1;
           count <=  count  +  1;
         else
           state  <=  st2_transition;
           count <=  X"0";
        end if;

        when st2_transition =>

           if count < sec2 then
             state <=  st2_transition;
             count <= count + 1;
           else
             state <= st3;
```

```
    count <= X"0";
  end if;

when st3 =>

  if count < sec10 then
    state <= st3;
    count <= count + 1;
  elsif (count > sec10) and (sensors_horizontal="00") then
    state <= st3;
    count <= count + 1;
  else
    state <= st4_transition;
    count <= X"0";
  end if;

when st4_transition =>

  if count < sec2 then
    state <=  st4_transition;
    count <= count + 1;
  else
    state <=st1;
    count <= X"0";
  end if;

when others =>
   state <= st1;
```

```
        end case;

    end if;

  end process;



 OUTPUT_DECODE: process (state)
    begin

     case state is
      when st1            => traffic_lights <= b"100_100_001_001";  -- horizontal green   and
vertical red
      when st2_transition => traffic_lights <= b"100_100_010_010";  -- horizontal yellow  and
vertical red
      when st3            => traffic_lights <= b"001_001_100_100";  -- horizontal red     and   vetical
green
      when st4_transition => traffic_lights <= b"010_010_100_100";  -- horizontal red     and
vetical yellow
      when others         => traffic_lights <= b"100_100_100_100"; -- set all road to red to avoid
accidents
     end case;

    end process;


end Behavioral;
```

Traffic Light System 3 Test Bench

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



entity TB_TLC_3 is
end entity;




architecture tb of TB_TLC_3 is

 signal sensors_vertical :  STD_LOGIC_VECTOR(1 downto 0);
 signal sensors_horizontal :  STD_LOGIC_VECTOR(1 downto 0);
 signal clck :   STD_LOGIC;
 signal reset : STD_LOGIC;
 signal traffic_lights :  STD_LOGIC_VECTOR (11 downto 0);
begin

  DUT : ENTITY work.TLC_3

  PORT MAP(sensors_vertical=>sensors_vertical,
           sensors_horizontal=>sensors_horizontal,
           clck=>clck,
           reset=>reset,
           traffic_lights=>traffic_lights);
```

```
Clock : process
begin
clck <= '0';
wait for 10 ns;
clck <= '1';
wait for 10 ns;
end process;
-- clck has a clock cycle of 20 ns



stimulis : process
begin
report("Starting simulation");


  reset <= '1'; wait for 20 ns;
  reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 200 ns;-- wait until the 10 cycles finish
  reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns;-- wait until we exceed 10 cycles but dont chane state
  reset <= '0';   sensors_vertical <= "10";   sensors_horizontal <= "00";   wait for 40 ns;-- supposed to go to S2
  reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns; -- supposed to go to S3 whatever the value of sensors is


  reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 200 ns;-- wait until the 10 cycles finish
  reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns;-- wait until we exceed 10 cycles but dont chane state
  reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "10";   wait for 40 ns;-- supposed to go to S4
```

```
   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns; --
supposed to go to S1 whatever the value of sensors is


   reset <= '1'; wait for 20 ns;

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 200 ns;-- wait
until the 10 cycles finish

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns;-- wait
until we exceed 10 cycles but dont chane state

   reset <= '0';   sensors_vertical <= "01";   sensors_horizontal <= "00";   wait for 40 ns;--
supposed to go to S2

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns; --
supposed to go to S3 whatever the value of sensors is


   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 200 ns;-- wait
until the 10 cycles finish

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns;-- wait
until we exceed 10 cycles but dont chane state

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "01";   wait for 40 ns;--
supposed to go to S4

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns; --
supposed to go to S1 whatever the value of sensors is


   reset <= '1'; wait for 20 ns;

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 200 ns;-- wait
until the 10 cycles finish

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns;-- wait
until we exceed 10 cycles but dont chane state

   reset <= '0';   sensors_vertical <= "11";   sensors_horizontal <= "00";   wait for 40 ns;--
supposed to go to S2

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns; --
supposed to go to S3 whatever the value of sensors is
```

```
   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 200 ns;-- wait
until the 10 cycles finish

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns;-- wait
until we exceed 10 cycles but dont chane state

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "11";   wait for 40 ns;--
supposed to go to S4

   reset <= '0';   sensors_vertical <= "00";   sensors_horizontal <= "00";   wait for 40 ns; --
supposed to go to S1 whatever the value of sensors is


 wait for 20 ns;



report("End simulation");

end process;

end architecture;
```