EDA Project Documentation
(Traffic light controller)

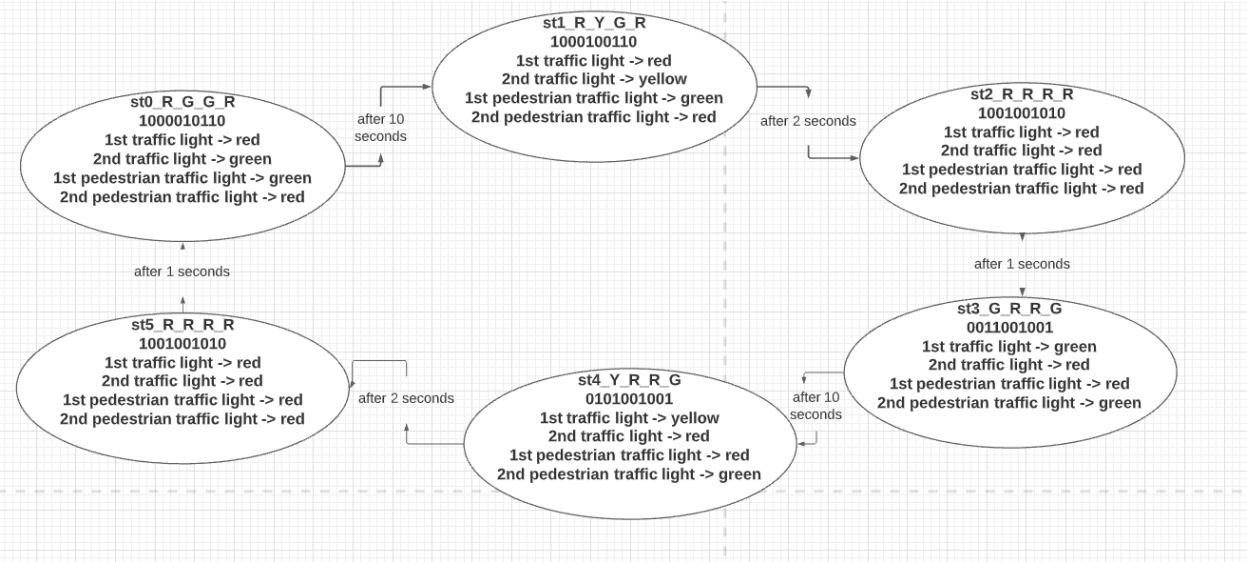
Mahmoud Abdalla Mohasseb Abdelrahim

20P2787

- **Introduction:**

  We have two traffic lights for cars. Each one of them has traffic light for pedestrian. When 1st traffic light of car is green / yellow, the 2nd traffic light will be red, 1st pedestrian traffic light will be red, 2nd pedestrian traffic light will be green and same for 2nd traffic light for cars. In some second all traffic lights will be red. Two pedestrian traffic lights have buttons for pedestrian to press them to be able to cross the street. When pedestrian press button two traffic lights of car will be red and two traffic lights for pedestrian will be green for two seconds. After two seconds state of traffic lights will continue from last state.

- **State Diagram:**



Code of Traffic light controller:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY TrafficLightController IS
  PORT(
    trafficLights :OUT STD_LOGIC_VECTOR(9 DOWNTO 0);      -- we need here 10 bits (3 for 1st traffic light, 3 for 2nd
    clk :IN STD_LOGIC;
    press :IN STD_LOGIC
  );
END TrafficLightController;

ARCHITECTURE Behavioral of TrafficLightController IS
  TYPE state_type IS(st0_R_G_G_R, st1_R_Y_G_R, st2_R_R_R_R, st3_G_R_R_G, st4_Y_R_R_G, st5_R_R_R_R, st6_R_R_G_G);
  SIGNAL state:state_type;
  SIGNAL state1:state_type;      -- to save state of two traffic lights so if pedestrain press button we have last st
  SIGNAL count:STD_LOGIC_VECTOR(4 DOWNTO 0);     -- to count seconds for traffic lights for cars
  SIGNAL countp:STD_LOGIC_VECTOR(4 DOWNTO 0);    -- to count seconds for traffic lights of pedestrain if pedestrain p
  CONSTANT sec10:STD_LOGIC_VECTOR(4 DOWNTO 0) := "01010";
  CONSTANT sec2:STD_LOGIC_VECTOR(4 DOWNTO 0) := "00010";
  CONSTANT sec1:STD_LOGIC_VECTOR(4 DOWNTO 0) := "00001";
  BEGIN
    PROCESS(clk, press)
      BEGIN
        IF press = '1' THEN  --pedestrain pressed button
          state <= st6_R_R_G_G;
          if countp < sec2 then
            state <= st6_R_R_G_G; --reset to red for two traffic lights and green for two pedestrain traffic lights
            countp <= countp + 1;
          else
            countp <= "00001";
          end if;
        ELSIF Clk' event and Clk = '1' then -- check for raising edge
          case (state1) is
            when st0_R_G_G_R =>
              if count < sec10 then
                state1 <= st0 R G G R;
```

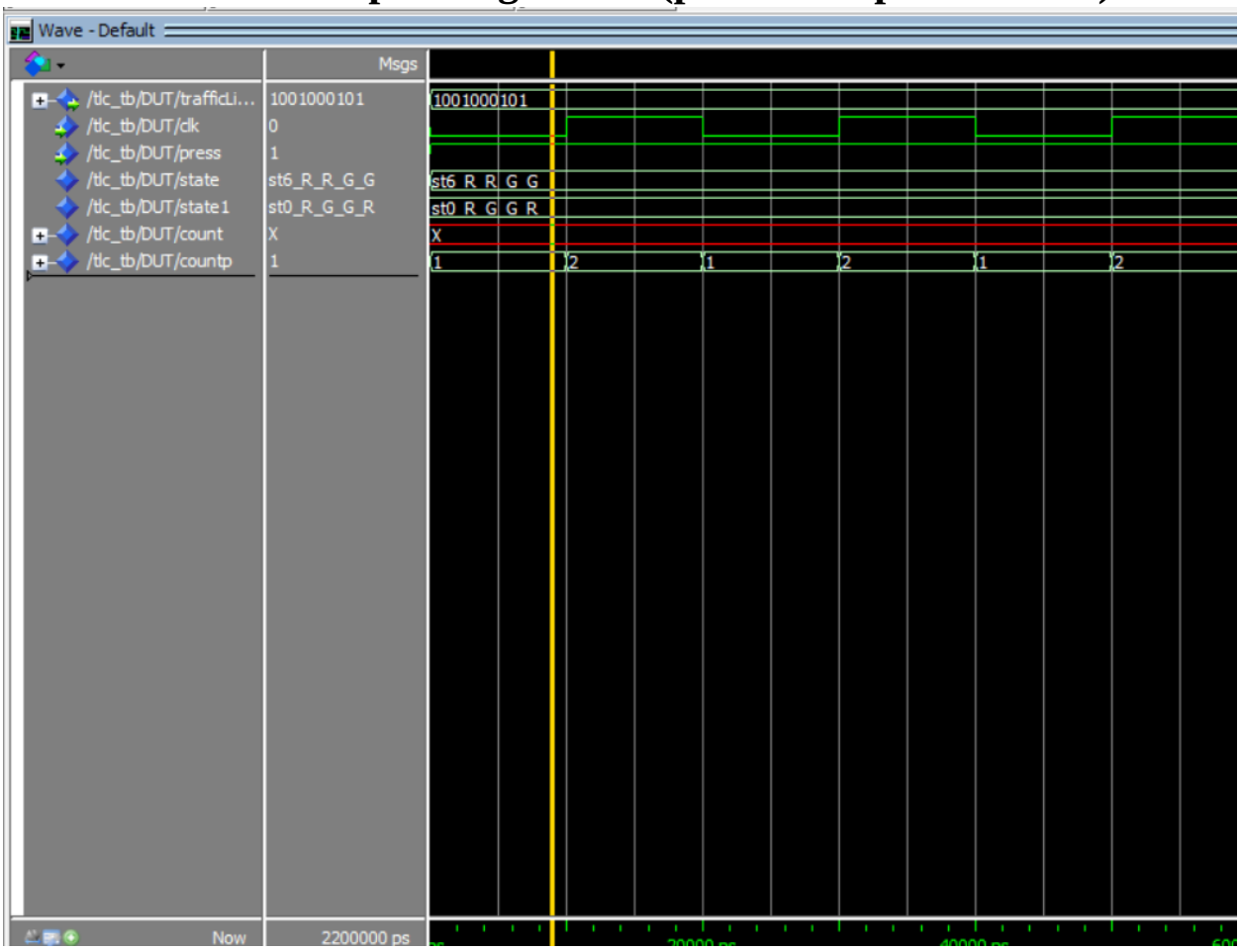
```
                state <= state1;
                count <= count + 1;
              else
                state1 <= st1_R_Y_G_R;
                state <= state1;
                count <= "00000";
              end if;
            when st1_R_Y_G_R =>
              if count < sec2 then
                state1 <= st1_R_Y_G_R;
                state <= state1;
                count <= count + 1;
              else
                state1 <= st2_R_R_R_R;
                state <= state1;
                count <= "00000";
              end if;
            when st2_R_R_R_R =>
              if count < sec1 then
                state1 <= st2_R_R_R_R;
                state <= state1;
                count <= count + 1;
              else
                state1 <= st3_G_R_R_G;
                state <= state1;
                count <= "00000";
              end if;
            when st3_G_R_R_G =>
              if count < sec10 then
                state1 <= st3_G_R_R_G;
                state <= state1;
                count <= count + 1;
              else
                state1 <= st4_Y_R_R_G;
                state <= state1;
                count <= "00000";
```

```
73              end if;
74            when st4_Y_R_R_G =>
75              if count < sec2 then
76                state1 <= st4_Y_R_R_G;
77                state <= state1;
78                count <= count + 1;
79              else
80                state1 <= st5_R_R_R_R;
81                state <= state1;
82                count <= "00000";
83              end if;
84            when st5_R_R_R_R =>
85              if count < sec1 then
86                state1 <= st5_R_R_R_R;
87                state <= state1;
88                count <= count + 1;
89              else
90                state1 <= st0_R_G_G_R;
91                state <= state1;
92                count <= "00000";
93              end if;
94            when st6_R_R_G_G =>
95              countp <= "00000";
96            END CASE;
97          END IF;
98        END PROCESS;
99
100   OUTPUT_DECODE: process (state)
101     BEGIN
102       case state is
103         when st0_R_G_G_R => Trafficlights <= "1000010110"; -- 1st traffic -> Red, 2nd traffic -> green, 1st peds traffic -
104         when st1_R_Y_G_R => Trafficlights <= "1000100110"; -- 1st traffic -> Red, 2nd traffic -> yellow, 1st peds traffic
105         when st2_R_R_R_R => Trafficlights <= "1001001010"; -- 1st traffic -> Red, 2nd traffic -> red, 1st peds traffic ->
106         when st3_G_R_R_G => Trafficlights <= "0011001001"; -- 1st traffic -> green, 2nd traffic -> red, 1st peds traffic -
107         when st4_Y_R_R_G => Trafficlights <= "0101001001"; -- 1st traffic -> yellow, 2nd traffic -> red, 1st peds traffic
108         when st5_R_R_R_R => Trafficlights <= "1001001010"; -- 1st traffic -> Red, 2nd traffic -> red, 1st peds traffic ->
109         when st6_R_R_G_G => Trafficlights <= "1001000101"; -- 1st traffic -> red, 2nd traffic -> red, 1st peds traffic ->
110       END CASE;
111   END process;
112 END Behavioral;
```

- **Code of test bench:**

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3   ENTITY TLC_tb is
4   END ENTITY;
5   ARCHITECTURE tb of TLC_tb IS
6   signal Trafficlights : std_logic_vector (9 downto 0);
7   signal clk,press: std_logic;
8   begin
9
10  DUT : ENTITY work.TrafficLightController
11  PORT MAP(Trafficlights=>Trafficlights,clk=>clk,press=>press);
12  Clock : process
13  begin
14  Clk <= '0';
15  wait for 10 ns;
16  Clk <= '1';
17  wait for 10 ns;
18  end process;
19  stimulis : process
20  begin
21   report("Starting simulation");
22   press <= '1';wait for 60 ns;
23   press <= '0';wait for 200000000 ns;
24   report("End simulation");
25  end process;
26  end architecture;
```

- **Wave Form of pressing button (pedestrian press button):**



- **Wave Form of traffic lights**