Lab 10

Traffic Light Controller

CPE 272 Lab
Fall 2022

Name: Azain Uqaily

Lab Partner: Dhyan Patel

10/3/22

Introduction
The final lab uses the DE10 Lite board in a way that we have not used before. This lab uses the board's GPIO pins to interact with a breadboard that is set up to imitate a road intersection with traffic lights to control the flow of traffic. We program our board using the idea of finite state machines where different states correspond to different behavior of the traffic lights.

Part I

Experiment:
For part I, the traffic light should continue being green as long as there are vehicles on that road. Meaning the input (sensor) Ta or Tb is '1'. Once this condition is false it will turn yellow on the next clock cycle and will give the green light to the other lane. The Moore state machine diagram is drawn below. The slow flip flop code is used and implemented with the help of port map statements as well.

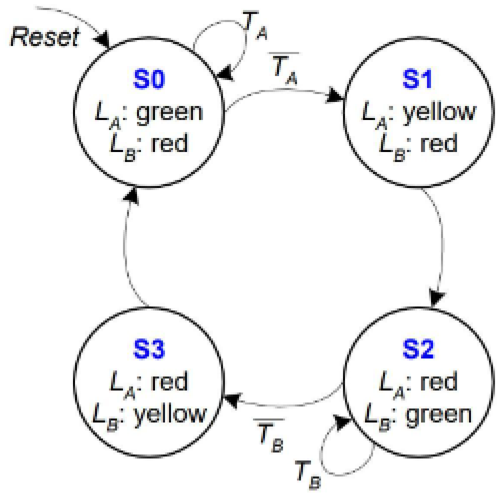
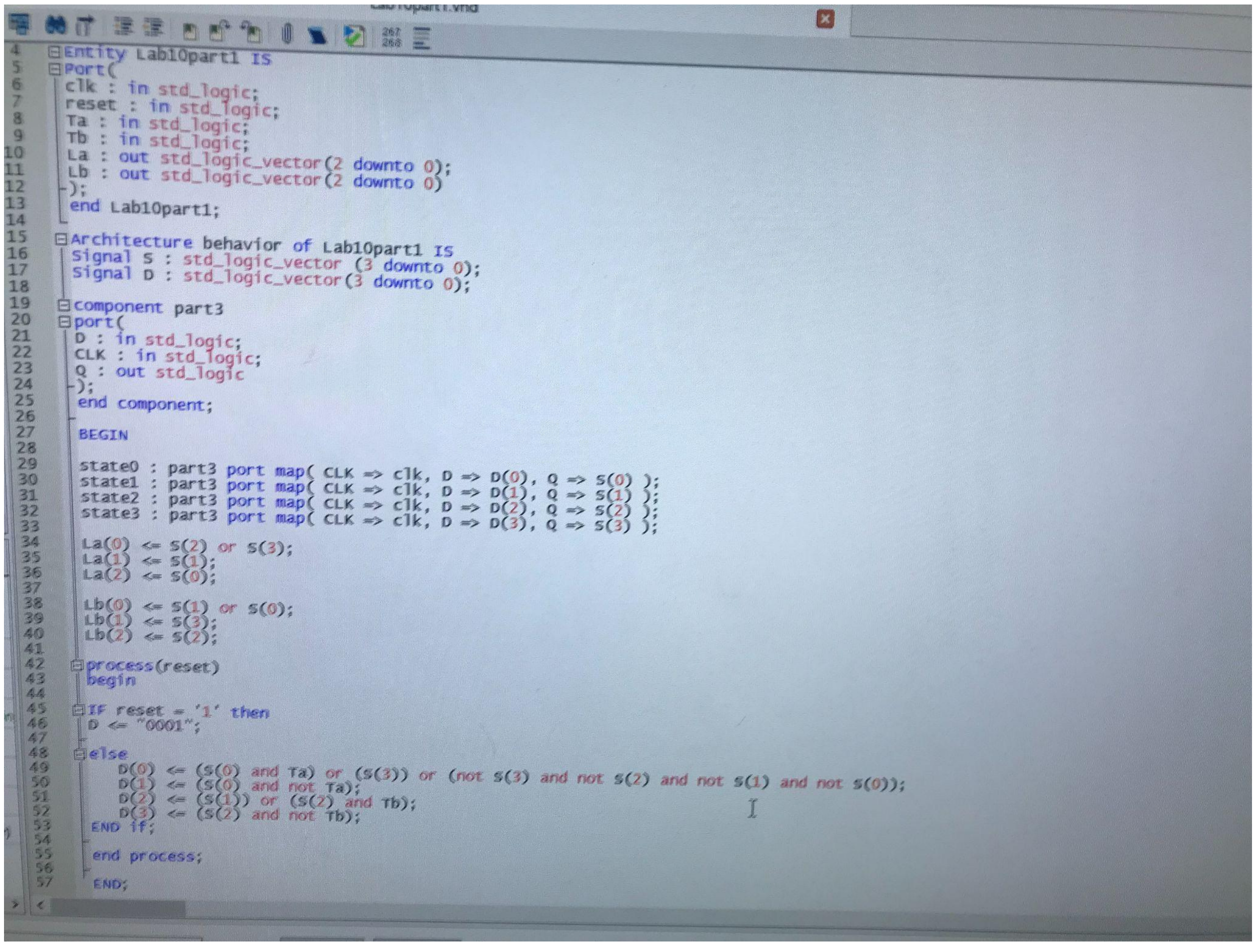


Figure: State Transition Diagram Part1



```
Entity Lab10part1 IS
Port(
 clk : in std_logic;
 reset : in std_logic;
 Ta : in std_logic;
 Tb : in std_logic;
 La : out std_logic_vector(2 downto 0);
 Lb : out std_logic_vector(2 downto 0)
);
end Lab10part1;

Architecture behavior of Lab10part1 IS
Signal S : std_logic_vector (3 downto 0);
Signal D : std_logic_vector(3 downto 0);

component part3
port(
D : in std_logic;
CLK : in std_logic;
Q : out std_logic
);
end component;

BEGIN

state0 : part3 port map( CLK => clk, D => D(0), Q => S(0) );
state1 : part3 port map( CLK => clk, D => D(1), Q => S(1) );
state2 : part3 port map( CLK => clk, D => D(2), Q => S(2) );
state3 : part3 port map( CLK => clk, D => D(3), Q => S(3) );

La(0) <= S(2) or S(3);
La(1) <= S(1);
La(2) <= S(0);

Lb(0) <= S(1) or S(0);
Lb(1) <= S(3);
Lb(2) <= S(2);

process(reset)
begin

IF reset = '1' then
D <= "0001";

else
    D(0) <= (S(0) and Ta) or (S(3)) or (not S(3) and not S(2) and not S(1) and not S(0));
    D(1) <= (S(0) and not Ta);
    D(2) <= (S(1)) or (S(2) and Tb);
    D(3) <= (S(2) and not Tb);
END if;

end process;

END;
```

Figure: VHDL Code for part1

Results:

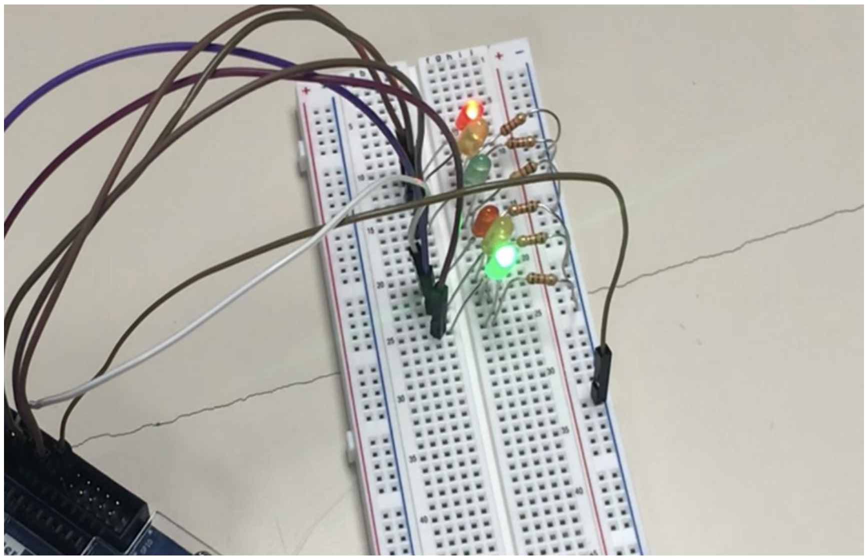


Figure: Part 1 BredBoard Output

As we can see in the image above, the bottom set of traffic lights is showing green meaning theoretically its lane still has traffic on it. Once this traffic is cleared up, it will switch to yellow and then show green on the other traffic light.

PART II

Experiment:

In part II we improved our design from part I by showing both lanes a red light for some time before showing either of them a green light. For this part we are allowed to use enumerated types and case statements to implement the new design.

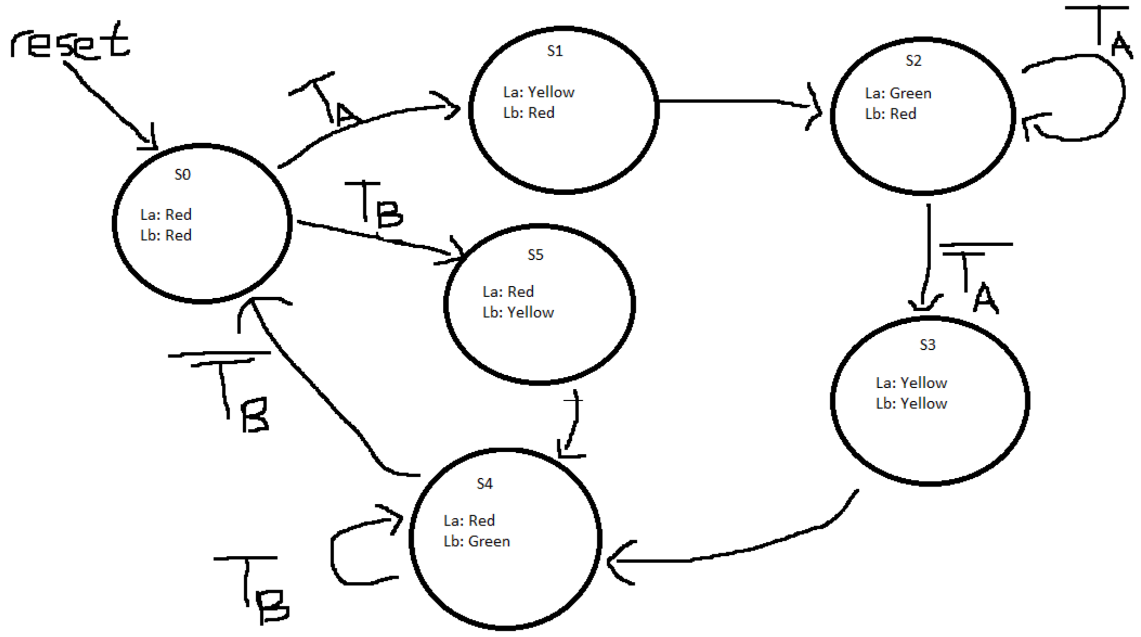
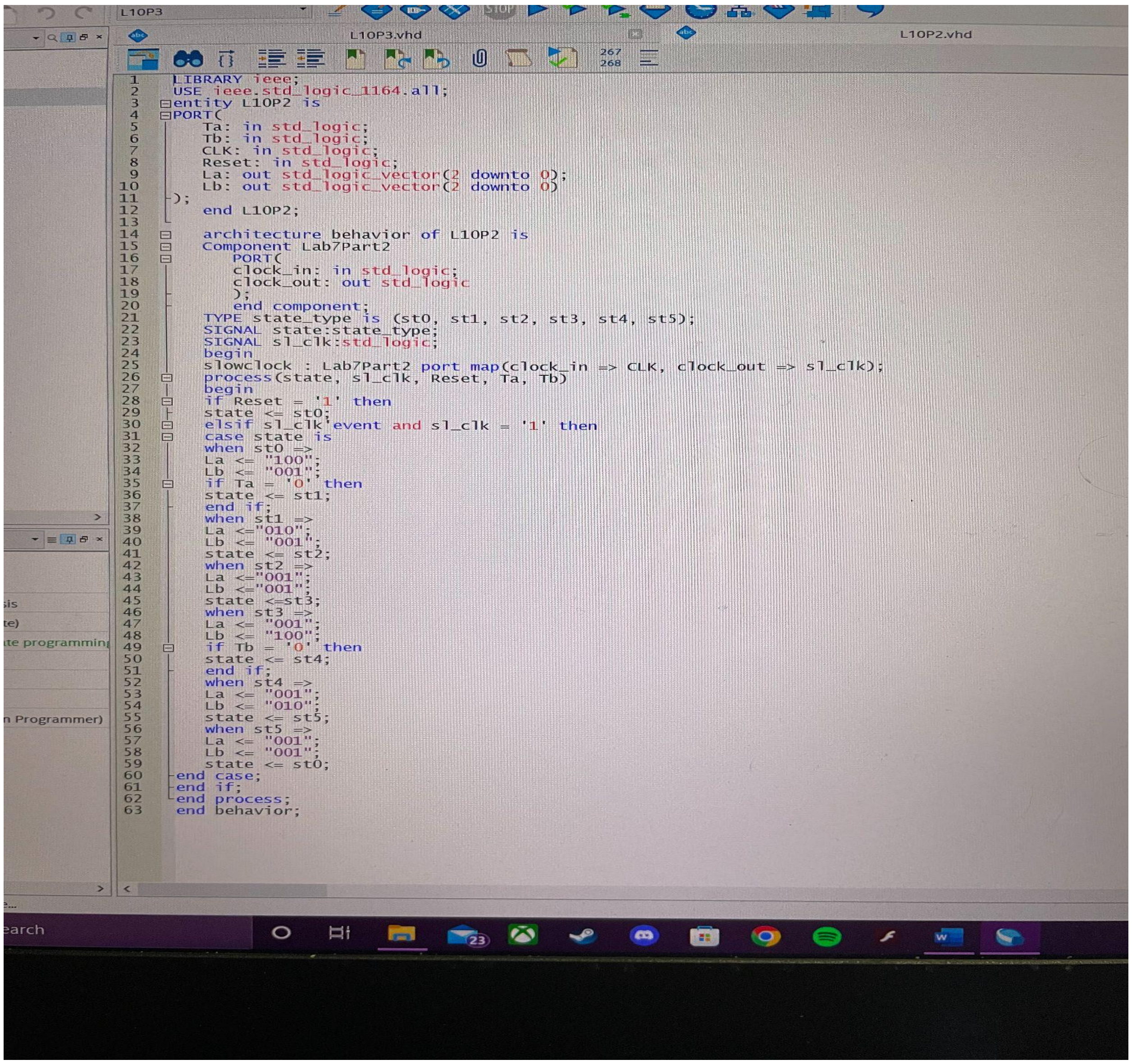


Figure: State Transition Diagram for Part 2

This state transition diagram shows a red light to both lanes for some time and after that if there is traffic on A it will show Lane A a green light till that traffic is cleared and then allow traffic to pass on B. After either cycle of green light it comes back to state 0 where both lanes are shown a red light.

Current State	Inputs		Next State
S	Ta	Tb	S’
S0	1	x	S1
S1	X	x	S2
S2	1	x	S2
S2	0	X	S3
S3	x	X	S4
S4	x	1	S4
S4	X	0	S0
S0	x	1	S5
S5	x	x	S4

Figure: Transition Table for part 2



```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
entity L10P2 is
PORT(
    Ta: in std_logic;
    Tb: in std_logic;
    CLK: in std_logic;
    Reset: in std_logic;
    La: out std_logic_vector(2 downto 0);
    Lb: out std_logic_vector(2 downto 0)
);
    end L10P2;

    architecture behavior of L10P2 is
    Component Lab7Part2
        PORT(
        clock_in: in std_logic;
        clock_out: out std_logic
        );
        end component;
    TYPE state_type is (st0, st1, st2, st3, st4, st5);
    SIGNAL state:state_type;
    SIGNAL sl_clk:std_logic;
    begin
    slowclock : Lab7Part2 port map(clock_in => CLK, clock_out => sl_clk);
    process(state, sl_clk, Reset, Ta, Tb)
    begin
    if Reset = '1' then
    state <= st0;
    elsif sl_clk'event and sl_clk = '1' then
    case state is
    when st0 =>
    La <= "100";
    Lb <= "001";
    if Ta = '0' then
    state <= st1;
    end if;
    when st1 =>
    La <="010";
    Lb <= "001";
    state <= st2;
    when st2 =>
    La <="001";
    Lb <="001";
    state <=st3;
    when st3 =>
    La <= "001";
    Lb <= "100";
    if Tb = '0' then
    state <= st4;
    end if;
    when st4 =>
    La <= "001";
    Lb <= "010";
    state <= st5;
    when st5 =>
    La <= "001";
    Lb <= "001";
    state <= st0;
end case;
end if;
end process;
end behavior;
```

Figure: VHDL Code for part 2

Results:

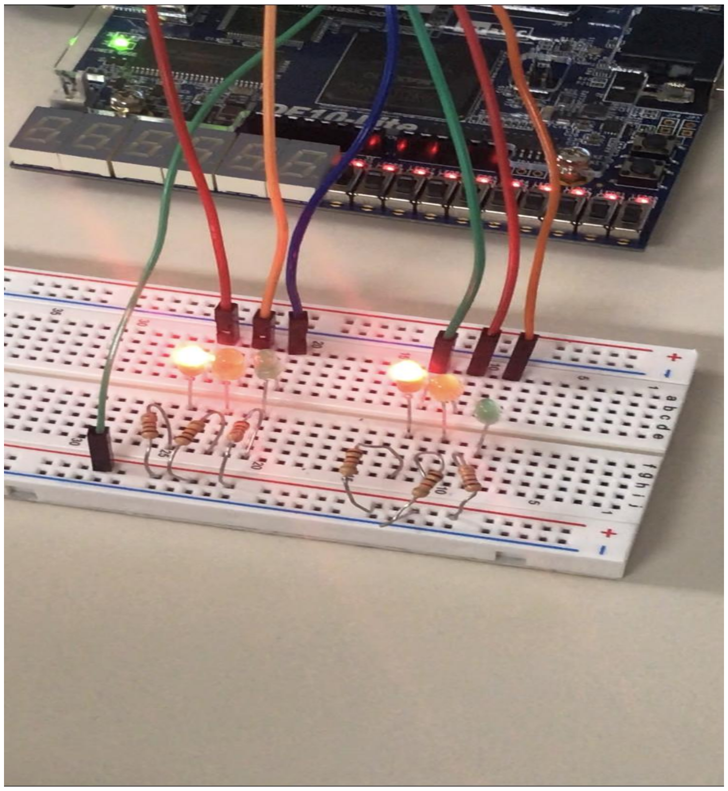


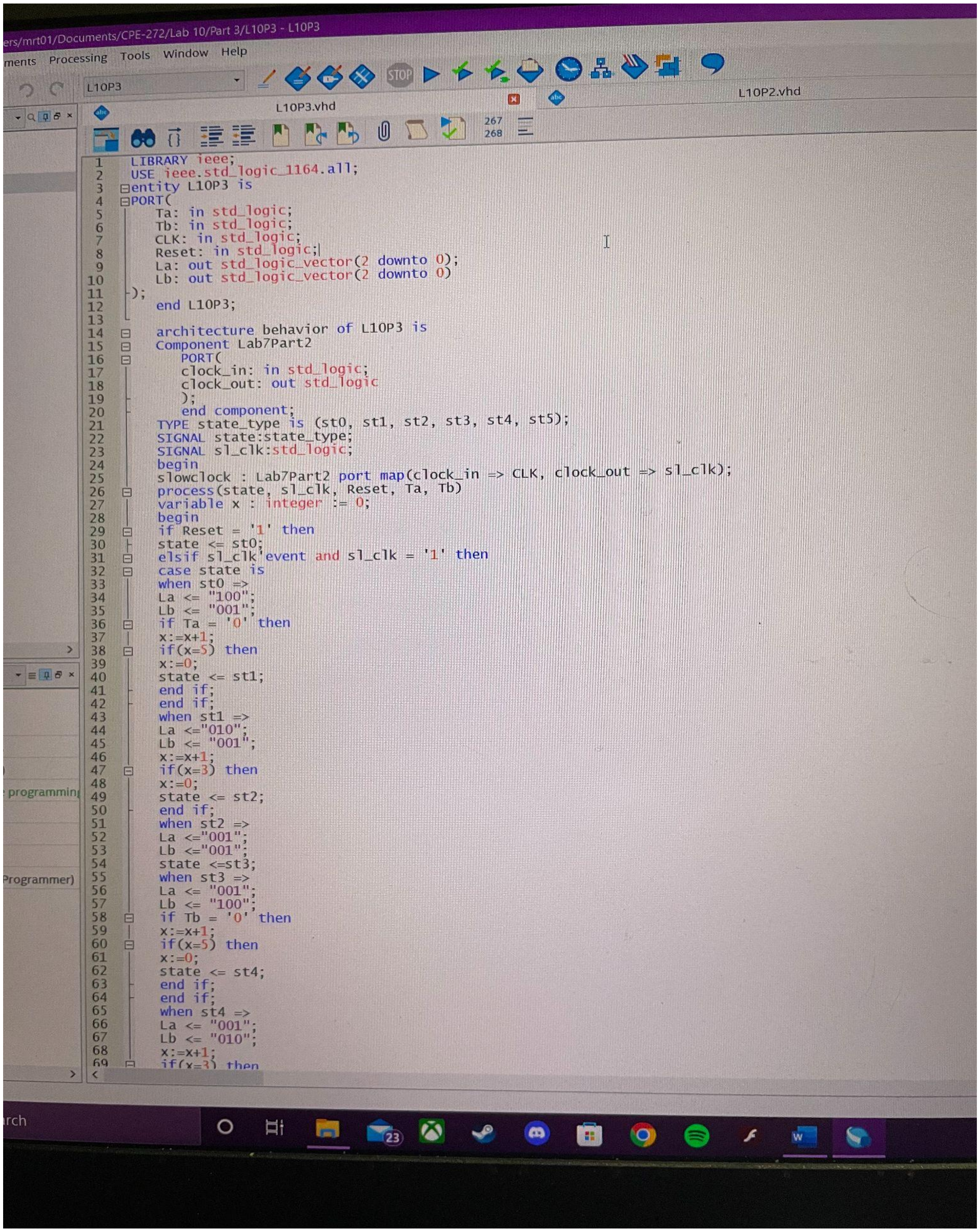
Figure: BredBoard output for part 2

As we can see in our results, we are able to show both the lanes a red light at the same time. This is better than part1 because this is how traffic lights in the real world function.

Part III

Experiment:

In this last part the goal is to focus on the timing of our transitions. In the real world our application may stay in different states for varied amounts of time. This part asks us to improve our code so that the green light is shown for at least 4 times longer than the time a red light is shown for and the yellow light is shown for twice the time of the red light. With the help of the slow clock, this is implemented in the code shown below.



```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
entity L10P3 is
PORT(
    Ta: in std_logic;
    Tb: in std_logic;
    CLK: in std_logic;
    Reset: in std_logic;
    La: out std_logic_vector(2 downto 0);
    Lb: out std_logic_vector(2 downto 0)
);
    end L10P3;

    architecture behavior of L10P3 is
    Component Lab7Part2
        PORT(
        clock_in: in std_logic;
        clock_out: out std_logic
        );
        end component;
    TYPE state_type is (st0, st1, st2, st3, st4, st5);
    SIGNAL state:state_type;
    SIGNAL sl_clk:std_logic;
    begin
    slowclock : Lab7Part2 port map(clock_in => CLK, clock_out => sl_clk);
    process(state, sl_clk, Reset, Ta, Tb)
    variable x : integer := 0;
    begin
    if Reset = '1' then
    state <= st0;
    elsif sl_clk'event and sl_clk = '1' then
    case state is
    when st0 =>
    La <= "100";
    Lb <= "001";
    if Ta = '0' then
    x:=x+1;
    if(x=5) then
    x:=0;
    state <= st1;
    end if;
    end if;
    when st1 =>
    La <="010";
    Lb <= "001";
    x:=x+1;
    if(x=3) then
    x:=0;
    state <= st2;
    end if;
    when st2 =>
    La <="001";
    Lb <="001";
    state <=st3;
    when st3 =>
    La <= "001";
    Lb <= "100";
    if Tb = '0' then
    x:=x+1;
    if(x=5) then
    x:=0;
    state <= st4;
    end if;
    end if;
    when st4 =>
    La <= "001";
    Lb <= "010";
    x:=x+1;
    if(x=3) then
```

Figure: VHDL Code for part 3

Results:

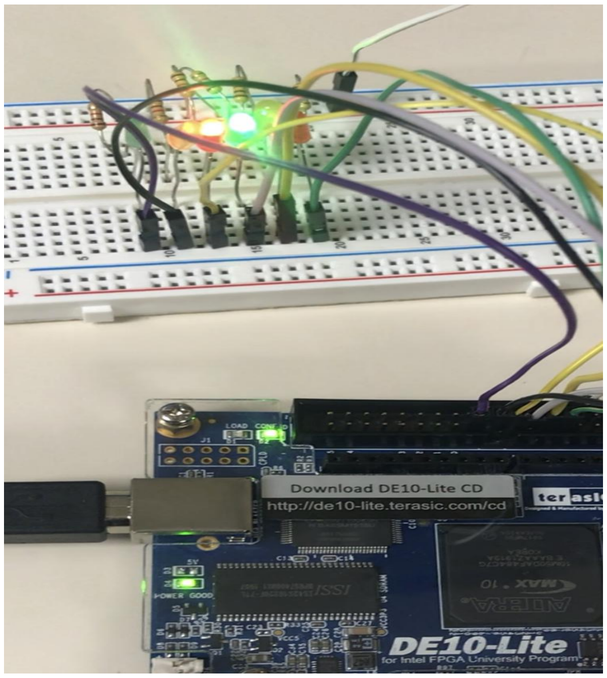

Download DE10-Lite CD
http://de10-lite.terasic.com/cd
DE10-Lite
for Intel FPGA University Program

Figure: BredBoard output for part 3

Although the timing aspect of the breadboard is not able to be shown, our traffic light was set up such that a green light shown would last about 8 seconds, the yellow light is around 4 seconds and red is around 2 seconds.

Part IV

I would like to enhance my design from part 3 by adding a simple pedestrian crossing button. Similar to a sensor this would just be a button in real life but when activated and after any green light is turned to red, this state would ensure that both lights are red and the pedestrian is able to cross. This state is achieved when an input of Bp is true (Button pressed). After some time maybe 10 seconds this light will turn off without logic and the arrows are drawn to return back to maybe the reset state where both the lights are red. And then from there whichever lane has traffic can be shown a green light.
For simplicity sake, I drew the part of the state transition diagram that pertains to this feature. Button pressed can be a condition that is being checked on every normal state.

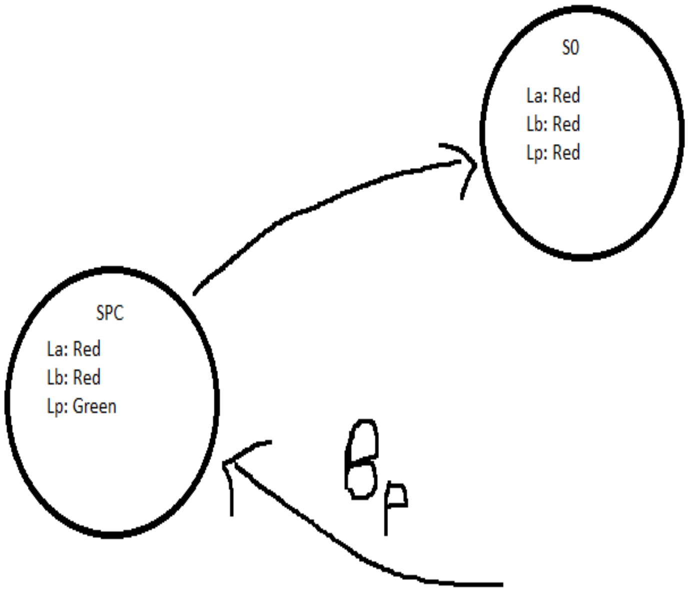


Figure: SPC (state pedestrian crossing)

Conclusion

After the seven segment display lab, this lab would probably be my second favorite lab. It was very fun and applicable but definitely challenging. As the final lab it was a great lab to conclude the semester by implementing almost all of our knowledge into building this traffic light controller. It was fun collaborating with different groups and seeing how they each implemented their versions of the traffic lights.