ECE 410

Lab 3 – Traffic Intersection Controller

Names: Madeeha Anjum, abuni gaiya

ID:1514645, 1544553

Date: December 6, 2019

Table of Contents

Click the requirement you would like to view

Abstract

The purpose of this lab was to help the students gain digital experience by designing a traffic light intersection controller. We implemented this traffic light intersection controller using the concept of finite state machines learned in lab 2. We designed the finite state machine with 4 states which were as follows;

State 0 -> When the Green light was on in the East West direction and the Red light was on in the North South direction.

State 1 -> When the Blue(yellow) light was on in the East West direction and the Red light was on in the North South direction.

State 2 -> When the Red light was on in the East West direction and the Green light was on in the North South direction.

State 3 -> When the Red light was on in the East West direction and the Blue(yellow) light was on in the North South direction.

INTRODUCTION

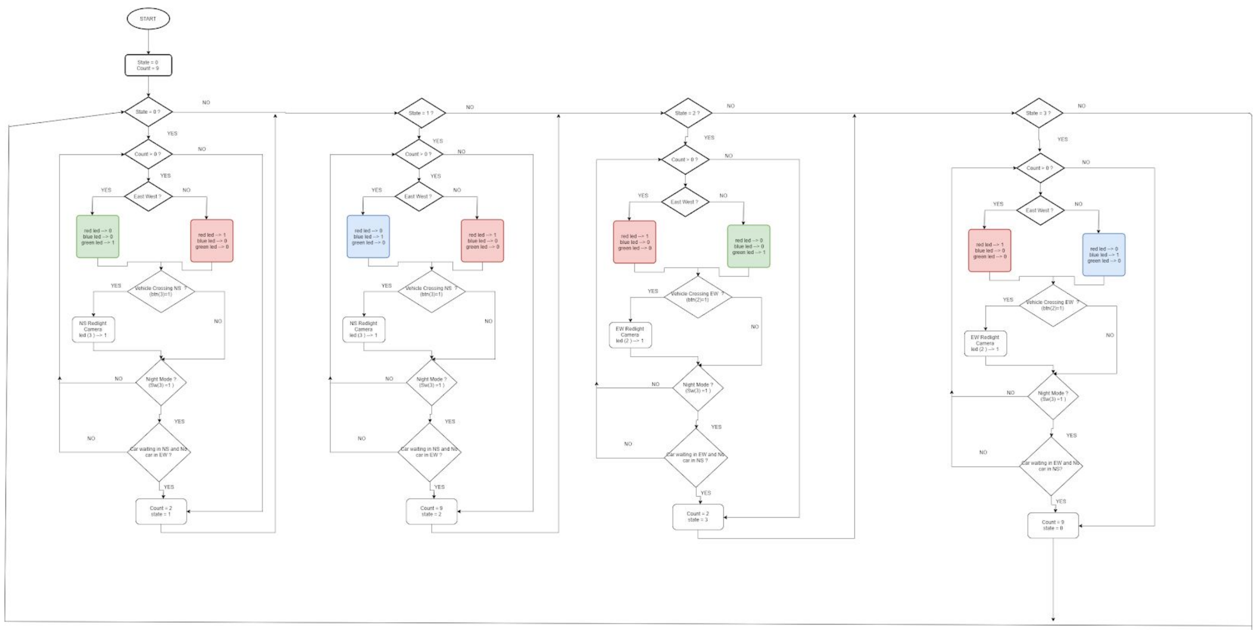
A traffic controller system is a big part of transportation systems in every big city. From an early age children are taught that Red light means Stop , Green light means Go and Orange or Yellow Light means get ready to Stop. In this lab we designed a replica of what most traffic controllers in big cities would do. We had to look out for different cases like when we are in night mode(traffic is usually lower at night) and a car is waiting in one direction while the are no cars waiting or approaching in the other direction. In this case we would have to let the car waiting pass, since there is no other car waiting in the other direction, there would be no need of making the car waiting to wait for seconds. We have implemented this traffic controller system with many other features which we will discuss more in the chapters below.

Design/Testing/Discussion:

Design Flow-Chart

In this section we have attached the flow chart that we used to implement our design for the traffic controller system below; We would also include a clearer copy in our zip file.

Figure 1: Flow chart for requirement 1 - 4



From our flow chart we were able to detect a sequence where the light goes from green to blue to red. We also observed that the steps to change from state to state in each state where similar.

Requirement #1:

In this part of the lab we verified the correct operation of the given design. The seven-segment display only showed the values of 00, 01 and 02; so, we updated the relevant sections to be able to see a complete countdown happen. In requirement 2 we tested the design.

As we did in lab 2 , when the count delay was greater than 9 we divided the count delay by 10 to get the most significant digit(the leftmost digit to be displayed seven segment display) and we just subtracted the most significant digit multiplied by 10 from the count delay to get the least significant digit. When the count delay was less than 9 we displayed it on the rightmost side of the seven segment display and displayed zero on the leftmost side as the most significant digit. The code for displaying the digits for the seven segment was already partially given to us so we just had to complete it. In requirement 2 we tested the design.

Design code

```
Decoder_4to7Segment: process (clk)
begin

    -- Update following case statement to display complete range of digit_7seg_display
values on 7-segments.
        case digit_7seg_display is
--digit (_) display on segment #1 when CC='0' on segment #2 when CC='1'
            when 0 =>
                          out_7seg<="0111111";
            when 1 =>
                          out_7seg<="0110000";
            when 2 =>
                          out_7seg<="1011011";
            when 3 =>
                          out_7seg<="1111001";
            when 4 =>
                          out_7seg<="1110100";
            when 5 =>
                          out_7seg<="1101101";
            when 6 =>
                          out_7seg<="1101111";
            when 7 =>
                          out_7seg<="0111000";
            when 8 =>
                          out_7seg<="1111111";
            when 9 =>
                          out_7seg<="1111101";

            when others =>

    end case;



    -- End of your design lines.
    end process;
```

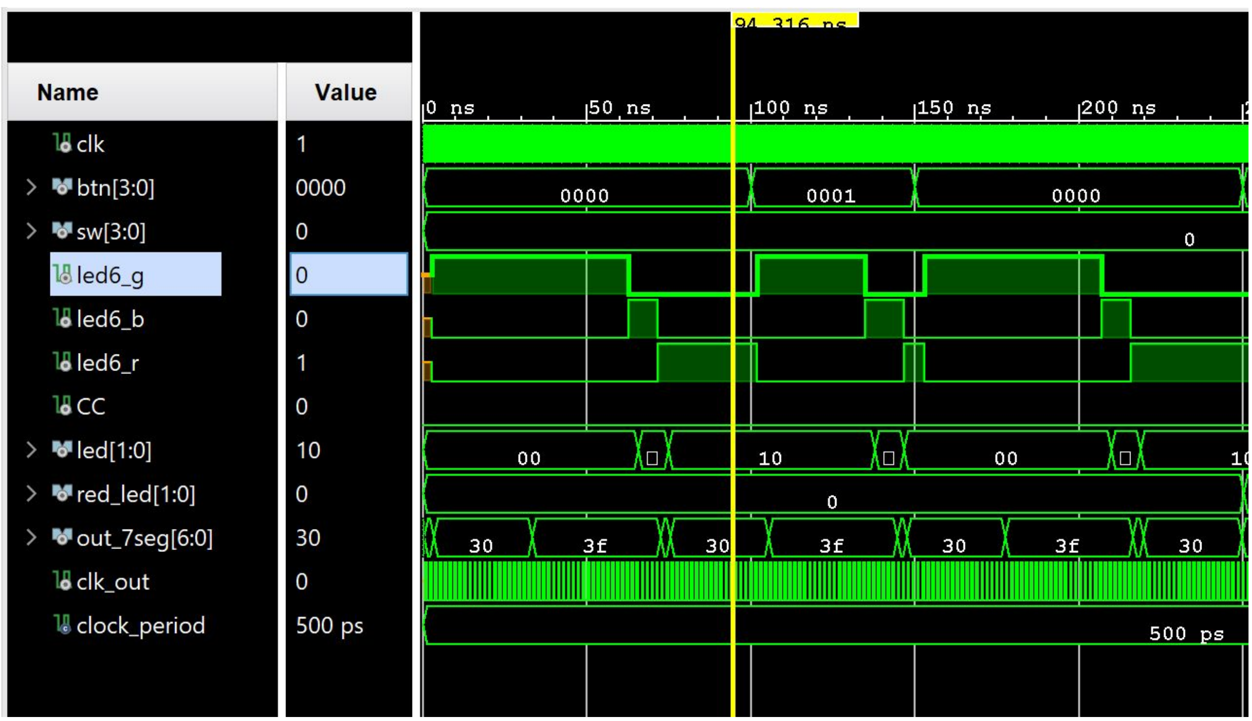
```
Select_7Segment: process (clk,clk_1Hz,select_segment)
begin

    if(Count_OneSecDelay>9) then
        --Write your design lines here
        Count_OneSecDelay_MSD <= Count_OneSecDelay/10;
        Count_OneSecDelay_LSD <= Count_OneSecDelay - Count_OneSecDelay_MSD*10;
        --End of your design lines.
    else
        Count_OneSecDelay_MSD<=0;
        Count_OneSecDelay_LSD<=Count_OneSecDelay;
    end if;

    if select_segment='1' then
        digit_7seg_display<= Count_OneSecDelay_LSD;
    else
        digit_7seg_display<= Count_OneSecDelay_MSD;
    end if;

    CC<=select_segment;

end process;
```

Requirement #2: Normal Operation

In this part of the lab we wrote a simulation test bench of the design. The way we designed the testbench was we created a simulation source named traffic_intersection_tb and created an entity for it and added a clock divider to the testbench adding it as a component to our traffic_intersection_tb and creating an instance of it. The various signals needed were declared in the architecture while initializing some to 0 or vectors of zero. Lastly, we mapped the various declared signals using portmap. The way we tested the design was the same way we would physically change the values on the board, for example holding down a button to see the other direction. For this part of the lab we tested how the states changed in response to btn(1) being pushed. For our simulation when the system is in State 0 (displayed by the led[0:1]) and btn[0] is not pressed meaning the leds are showing the lights in east-west direction, then led6_g is on, continuing to State 1 (displayed by the led[0:1] = 1) then finally to State 2 (led[0:1] = 2) and as soon as the btn[0] is pressed, the led6_r is off and led6_g turns on.

Simulation Results

Figure 2: Simulation results for requirement 2



Test bench code

```
--------------------------Requirement #2 Normal Simulation-----------------------------
---------

        wait for 100ns; --then continue

        btn<= "0001";

        wait for 50ns; --holding the input

        btn<= "0000";

        wait for 100ns;
```

1) As you can see right when the simulation begins, we have the following test case:

    bnt = : “0000”(indicating E/W direction)

    These values produce the outputs in order:

    led_g,b,r = ‘100’ , led = “00” --indicating a green light and state 0

    led_g,b,r = ‘010’ , led = “01” --indicating a blue light and state 1

    led_g,b,r = ‘001’ , led = “10” --indicating a red light and state 2

2) Nest when we change the input; while in state 2 to

    bnt = “0001” (indicating N/S direction)

    These values produce the outputs in order:

    led_g,b,r = ‘100’ , led = “10” --indicating a green light and state 2

    led_g,b,r = ‘010’ , led = “11” --indicating a blue light and state 3

    led_g,b,r = ‘001’ , led = “10” --indicating a red light and state 0

Requirement #3: Red Light Camera feature

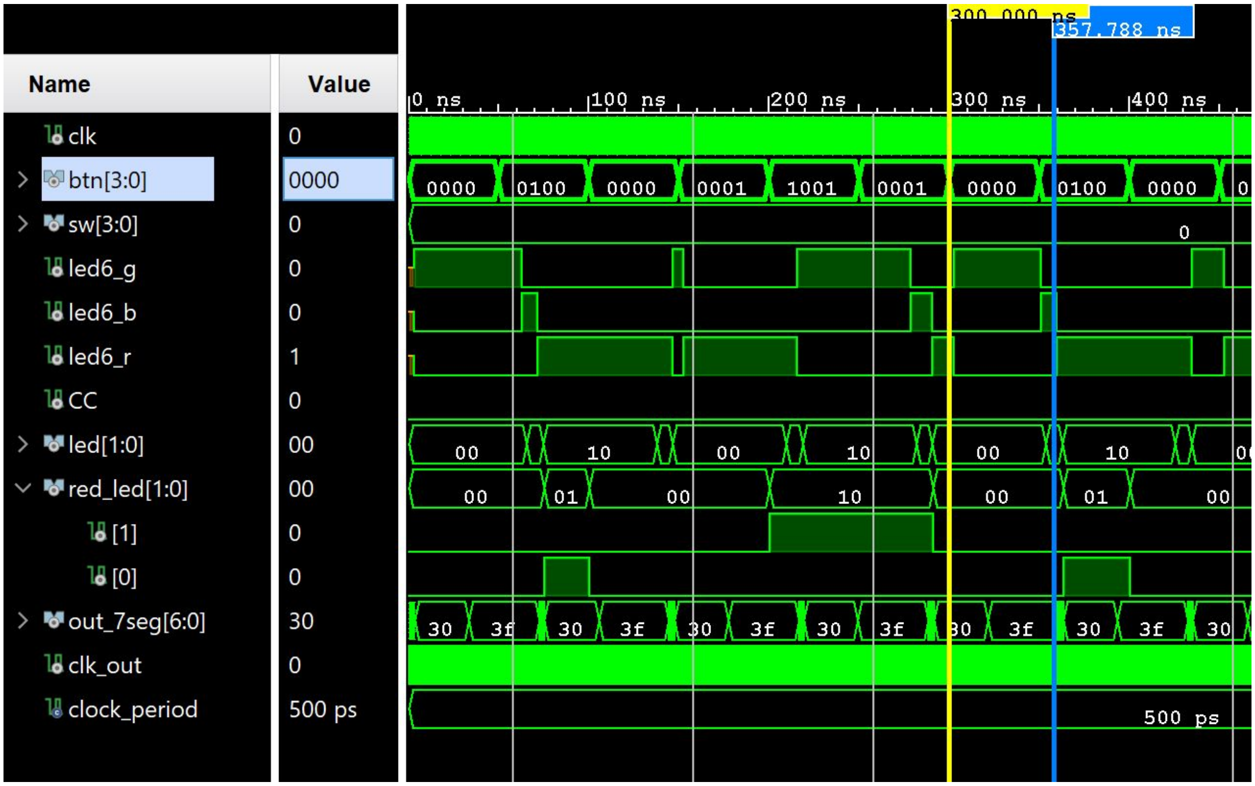
In this part of the lab we wrote code that would flash a red light (LED (2) E/W LED (3), (N/W) every time a vehicle approached the intersection encountered a red light.We implemented our design for this part with two processes one that detects the NorthSouth direction and one to detect the EastWest direction. For both processing we first checked that a car was approaching intersection i.e button 2 or button 3 was pressed. If a car was approaching intersection we would check if there is a red light in that direction by checking what state we are currently in. We finally flash the appropriate LED light if all the previous conditions were met. For the simulation we turn on the respective led when a vehicle approaching the intersection encounters a red light. In State 0 (led[0:1] = 0), east-west lights are green, when a vehicle approaches the intersection from north south direction (btn(3) = 1). The red-light camera led in the north-south direction turns on i.e. red_led(1)

Design

```
----------------------------------------------------------------
-Requirement #3 When a vehicle approaching intersection encounters a red light

-- Write a process for Red Light Camera feature at the intersection.
Red_light_Flash_WE: process(btn(2)) --EAST WEST
begin
case btn(2) is
  when '1'  =>
    if (state = S2 and btn(0)='0') or ( state = S3 and btn(0)='0') then
        red_led(0)<= '1';
    end if;
  when '0'  =>
    red_led(0)<= '0';
  when others =>
end case;
end process;

Red_light_Flash_NS: process(btn(3))-- NORTH SOUTH
begin
    case btn(3) is
      when '1'  =>
        if (state = S0 and  btn(0)='1') or  (state = S1 and btn(0)='1') then
            red_led(1)<= '1';
        end if;
      when '0'  =>
        red_led(1)<= '0';
      when others =>
    end case;
end process;
```

Simulation Results

Figure 3: Simulation for requirement 3



Testbench code

```
------------------------- Requirement #3 Red Light Flash------------------------------

-        Vehicle approaches intersection so flash_red if state is 2 or 3(red light)
       btn<= "0000";

       wait for 50ns;

       btn(2) <= '1'; -- Vehicle crossing on East/West (Output: red_led(0)=>1, at stat
es: 2, 3)

       wait for 50ns;

       btn(2) <= '0';

       wait for 50ns;

--        Vehicle approaches intersection so flash_red if state is 0 or 1(red light)

       btn<= "0001";

       wait for 50ns;

       btn(3) <= '1'; -- Vehicle crossing on North West (Output: red_led(1)=>1, at sta
tes: , 0, 1)

       wait for 50ns;

       btn(3) <= '0';

       wait for 50ns;
```

For the four test cases written in the testbench, we toggled the bnt(2) for East/West and btn(3) for North/South:

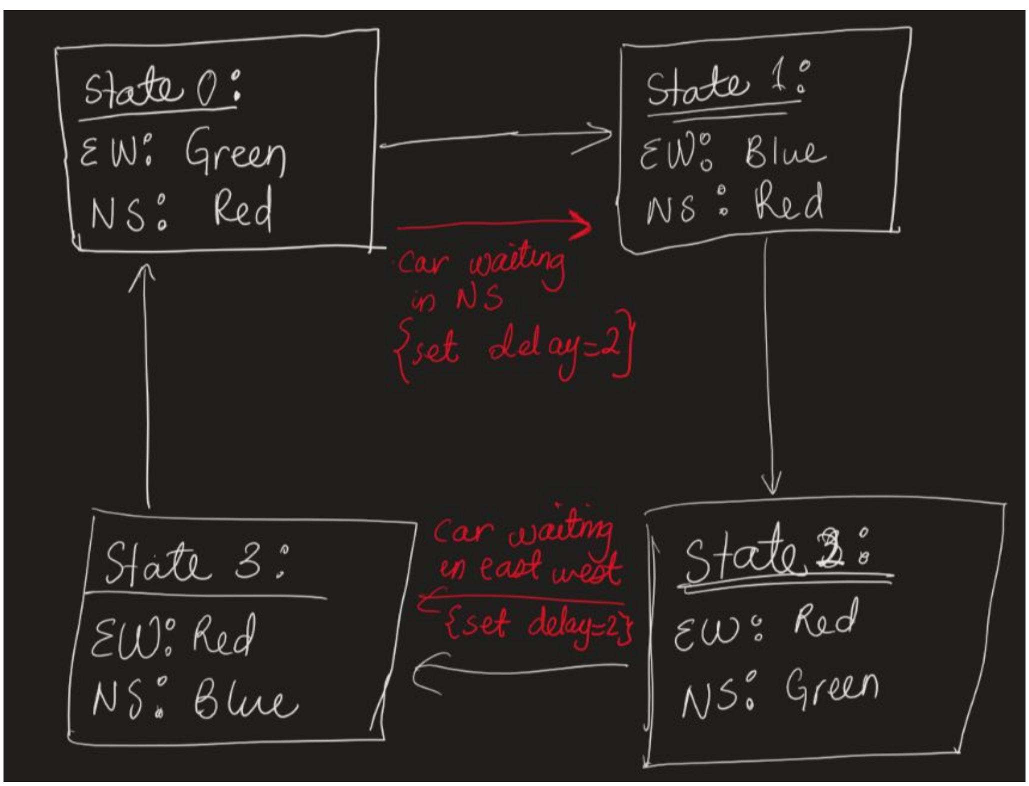
1) As we can see when btn(2) = ‘0’ nothing happens but as soon as btn (2) = ‘1’ and we encounter a red light (Led6_r = ‘1’) the red_led[0] = ‘1’
2) Next we can also see this happen when btn(3) = ‘0’ changes to btn(3) = ‘1’ , the red_led[1] = “1’ when we encounter a red light(Led6_r = ‘1’).

Requirement #4:Night time Operation

In this part of the lab we considered a situation where a vehicle encountered a red light while there weren't any vehicles passing or waiting in the perpendicular direction of travel. In that situation we implement our design to have the feature of properly turning the other direction of traffic lights to red, while vehicles waiting or approaching intersection traffic lights turned green. To accomplish this we enabled night mode by a holding SW(3). When this switch was enabled, the traffic lights are to change with approaching cars. We implemented this by altering the provided Traffic Intersection process. We designed it so that we did not have to consider the counter when transitioning to the next state. We used the given Ntswich to indicate whether we were in night mode.

The design for this part of the lab was more like an FSM(Finite State Machine) with Four different states. Initially we had four states that rotated in a sequential manner if State 0 and State 2 were more like the dominant states where the red light and green light where shown for each direction. State 1 and State 3 we saw as the transition state that allowed the green light from the previous to change red after a to Red by changing to blue(yellow) first before changing to red and changing the light in the other direction to green. This made sense as in an actual traffic system the light never changes immediately from green to red even in the night mode. For our implementation we checked if there was a if we had a green light in one direction and a red light in another direction, we checked to see if we in night mode and if there was a car waiting or approaching in the other direction. We would immediately change the delay to 2secs and change to the transition state. In our simulation we got a red light in east and west and no were present in the North/South direction so we made the light green changing the perpendicular light to red.

Figure 4: Rough sketch that was used to implement this requirement



Design

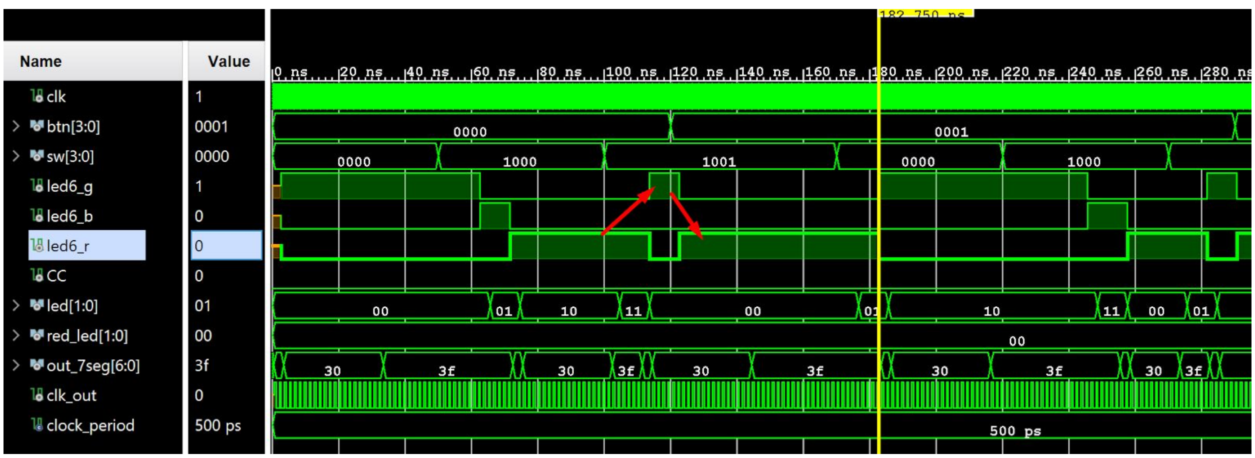
```
----------------------------------------------------------------
  --Requirement #4 Night Time Operation You can write design lines here to capture veh
icles presence and Night Time input (LDR).
  -- Write your design line here to update VehiclesPresence(0)
  VehiclesPresence(0) <= sw(0);
  -- Write your design line here to update VehiclesPresence(1)
  VehiclesPresence(1) <= sw(1);
  -- Write your design line here to update NTSwitch
  NTSwitch <= sw(3);
  --End of design lines.
----------------------------------------------------------------

      if btn(1)='1' then
          state<=S0;
      end if;

      if rising_edge(clk_1Hz) then
          Count_OneSecDelay<=Count_OneSecDelay-1;     --Increment one second count. ~1
.84 sec delay here

          case state is
              when S0 =>                             --East/West direction light gree
n
                      if Count_OneSecDelay>0 then
                          if btn(0)='0' then              --Since only have one RGB li
ght, else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                              led6_r<='0';
                              led6_b<='0';
                              led6_g<='1';
                          else
                              led6_r<='1';
                              led6_b<='0';
                              led6_g<='0';
                          end if;
              ----------------------------------------------------------------
              --    Requirement #4 Night Time Operation
                          if (NTSwitch = '1' and VehiclesPresence(1) = '1' and btn(2)
= '0') then
                              state<= S1;
                              Count_OneSecDelay <= 2;
                           end if;
              ----------------------------------------------------------------

                      else
                          state<=S1;
                          Count_OneSecDelay<=2;
                          if btn(0)='0' then         --Since only have one RGB light,
else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                              led6_r<='0';
                              led6_b<='1';
                              led6_g<='0';
                          else
                              led6_r<='1';
                              led6_b<='0';
```

```
                              led6_g<='0';
                          end if;
                      end if;

                  states_mon<="00";
                  -- ~1.7 sec delay here

              when S1 =>                            --East/West direction light yello
w=>blue on board
                      if Count_OneSecDelay>0 then
                          if btn(0)='0' then          --Since only have one RGB light,
else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                              led6_r<='0';
                              led6_b<='1';
                              led6_g<='0';
                          else
                              led6_r<='1';
                              led6_b<='0';
                              led6_g<='0';
                          end if;
                      else
                          state<=S2;
                          Count_OneSecDelay<=9;
                          if btn(0)='0' then           --Since only have one RGB light,
 else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                              led6_r<='1';
                              led6_b<='0';
                              led6_g<='0';
                          else
                              led6_r<='0';
                              led6_b<='0';
                              led6_g<='1';
                          end if;
                      end if;
                  states_mon<="01";

              when S2 =>                             -- East/West direction light red
 and North/South direction green.
                      if Count_OneSecDelay>0 then
                          if btn(0)='0' then           --Since only have one RGB light,
 else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                              led6_r<='1';
                              led6_b<='0';
                              led6_g<='0';
                          else
                              led6_r<='0';
                              led6_b<='0';
                              led6_g<='1';
                          end if;
              ----------------------------------------------------------------
              --    Requirement #4 Night Time Operation
                          if (NTSwitch = '1' and VehiclesPresence(0) = '1' and btn(3)
= '0') then
                              state<= S3;
                              Count_OneSecDelay <= 2;
                           end if;
              ----------------------------------------------------------------

                      else
```

```
                          state<=S3;
                          Count_OneSecDelay<=2;
                          if btn(0)='0' then            --Since only have one RGB ligh
t, else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                              led6_r<='1';
                              led6_b<='0';
                              led6_g<='0';
                          else
                              led6_r<='0';
                              led6_b<='1';
                              led6_g<='0';
                          end if;
                      end if;

                  states_mon<="10";
```

Simulation Results

Figure 5-Simulation part 1 for requirement 4



Testbench part 1

```
--        ---------------------Requirement #4 Night Time Operation--------------------
-----------------------
        --1)
        btn<= "0000";
        sw<= "0000";

        wait for 50ns;

        --SW(3) to simulate the light sensor
        sw(3) <= '1';

        wait for 50ns;

        sw(0) <= '1'; -- vehicle's presence in the East or West direction

        --should encounter a red light, we should then change it to green sw(1) is not
```

```
1(no vehical in N/S)

        --cheking theat the other direction is red
        wait for 20ns;
        btn<= "0001";

        wait for 50ns;
```

For the first test case written in the testbench, we activated night time operation by:

Sw = “ 1000”

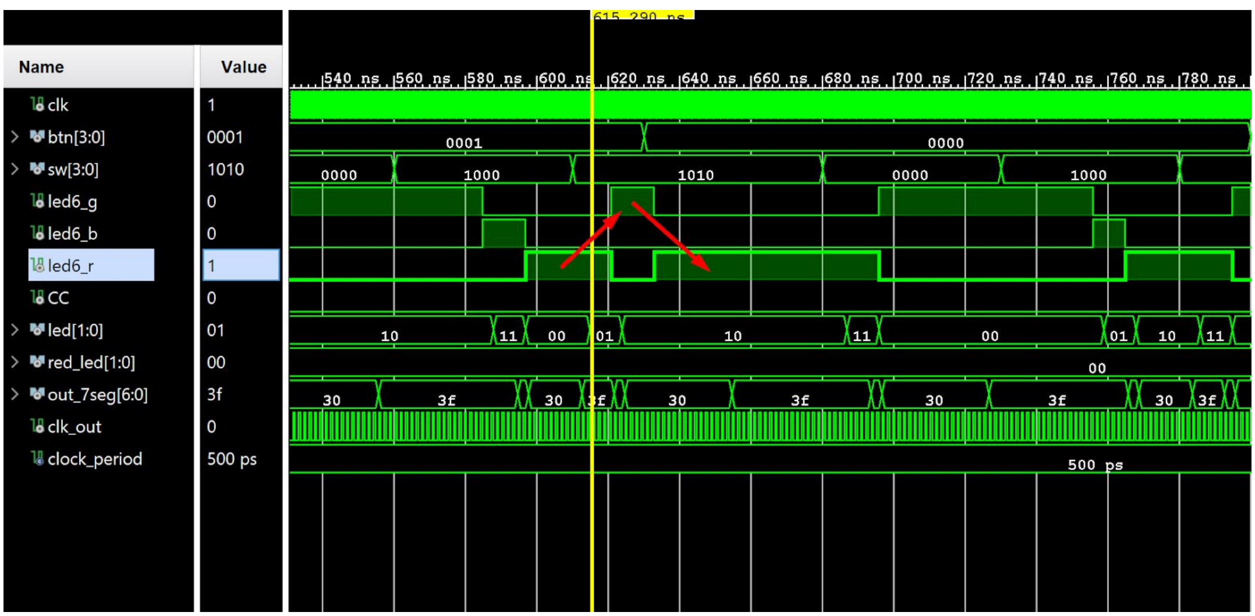
Then we simulated a vehicles presence in the East/ West direction by:

sw(0) <= '1001'

As we can see we got a red light and no cars in the North/South direction we made the light green and then we made the North/South light red.

Simulation 2

Figure 6: Simulation part 2 for requirement 4



Testbench part 2

```
        --2)
        btn<= "0001";
        sw<= "0000";

       wait for 50ns;

        --SW(3) to simulate the light sensor
        sw(3) <= '1';

        wait for 50ns;

        sw(1) <= '1'; -- vehicle's presence in the North or South direction

        --should encounter a red light, we should then change it to green sw(0) is not
 1(no vehical in E/W)

        --cheking theat the other direction is red
        wait for 20ns;

        btn<= "0000";

        wait for 50ns;
```

For the second test case after resetting we activated night time operation by:

Sw = " 1000"

Then we simulated a vehicles presence in the North or South direction by:

sw(1) <= '1010'

As we can see once we got a red light and no cars in the East/West direction we made the light green and then we made the East/West light red.

REQUIREMENT #5: CHANGING THE DISPLAY AND COUNT DOWN

Design

In this part of the lab we reduced the green to yellow light countdown from 20 to 9 seconds and then we displayed the time remaining on right hand side 7-Segment. IN addition we also displayed the current system state on LED0 and LED1, ad well as the left hand side 7-segment. this part of the lab we reduced the green to blue(yellow) light countdown from 20 to 9 seconds and then we displayed the time remaining on right hand side 7-Segment. IN addition we also displayed the current system state on LED0 and LED1, ad well as the left hand side 7-segment.To change the countdown from 20secs to 9 seconds we change the initial initialization for theCount_OneSecDelay to from 20 to 09 as shown below:

```
Signal Count_OneSECDelay: natural:=9;
```

To display the states on the right-hand side of the 7-segment we assigned the states_monsignal, which stored the current state number to Count_OneSecDelay_MSD and

```
   ----------------------------------------------------------------
Select_7Segment: process (clk,clk_1Hz,select_segment)
begin
   if(Count_OneSecDelay>9) then
--Write your design lines here
        Count_OneSecDelay_MSD <= Count_OneSecDelay/10;
Count_OneSecDelay_LSD <= Count_OneSecDelay - Count_OneSecDelay_MSD*10;
--End of your design lines.
   else
      Count_OneSecDelay_MSD<= to_integer(unsigned(states_mon));
      Count_OneSecDelay_LSD<=Count_OneSecDelay;
   end if;
   if select_segment='1' then
      digit_7seg_display<= Count_OneSecDelay_LSD;
   else
      digit_7seg_display<= Count_OneSecDelay_MSD;
  end if;
CC<=select_segment;
end process;
      ----------------------------------------------------------------
```

Requirement #6 : Pedestrian Crosswalk

In this part of the lab we displayed the pedestrian crosswalk signal as '0' for stop and '1' for walking on the seven segments. We also used one segment for the East-West direction and other for the North-South direction pedestrian crosswalk signals, if a button is pressed on keypad, the led displays the signal for pedestrian walk. The left digit displays the signal for east-west direction and the right digit displays the signal for north-south direction. If the system is in State 1 or State 3 do nothing. If State 0 green for east and west, then display '10' meaning east and west can walk. If State 2 east and west are red lights then display '01' meaning North and South are allowed to walk. This required us to implement a toggle feature between viewing the state and the countdown or the pedestrian signals by using the keypad keys as inputs to toggle and display on seven segments, the default is to display the pedestrian crosswalk signal when keypad keys are held pressed.

We included the keypads by using implementing the JE ports within constraint file to take the value from the row and produce the values of the columns and check to see if the keypad has been pressed as shown below:

```
##Pmod Header JE

set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Col[0
] }]; #IO_L4P_T0_34 Sch=je[1]
set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Col[1
] }]; #IO_L18N_T2_34 Sch=je[2]
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Col[2
] }]; #IO_25_35 Sch=je[3]
set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Col[3
] }]; #IO_L19P_T3_35 Sch=je[4]
set_property -dict { PACKAGE_PIN V13   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Row[0
] }]; #IO_L3N_T0_DQS_34 Sch=je[7]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Row[1
] }]; #IO_L9N_T1_DQS_34 Sch=je[8]
set_property -dict { PACKAGE_PIN T17   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Row[2
] }]; #IO_L20P_T3_34 Sch=je[9]
set_property -dict { PACKAGE_PIN Y17   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Row[3
] }]; #IO_L7N_T1_34 Sch=je[10]
```

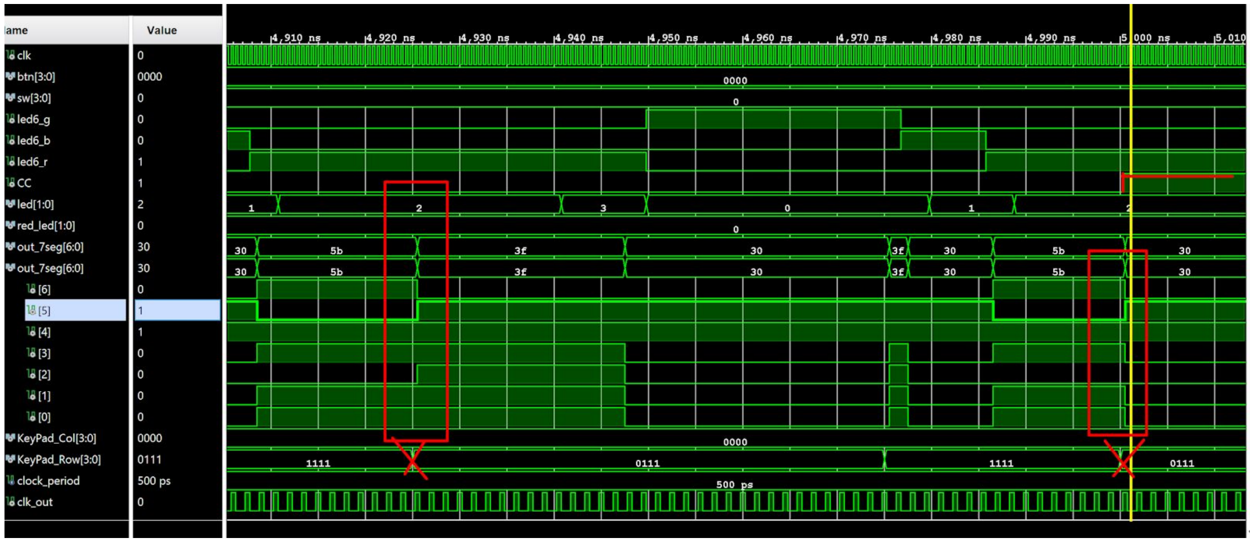
We also observed that when the Keypad_Col vector is "1111" it means that no key if pressed. Rather than have a bunch of if -else statements to check if each key on the keypad is pressed, we had one statement checking if no key is pressed if this statement is not true then it means a button on the keypad has been pressed and we display the pedestrian crosswalk signal.

Design

```
             ----------------------------------------------------------------
            --Requirement #6
             KeyPad_Col: out STD_LOGIC_VECTOR (3 downto 0) :="0000";
             KeyPad_Row: in STD_LOGIC_VECTOR(3 DOWNTO 0)
              ----------------------------------------------------------------

           );
end traffic_intersection;
begin

    Select_7Segment: process (clk,clk_1Hz,select_segment)
    begin
      ----------------------------------------------------------------
            --Requirement #5
         if (KeyPad_Row = "1111") then       --When no key is pressed, KeyPad_Row="111
1";
            --display state on RHS of 7 seg
            Count_OneSecDelay_LSD<= Count_OneSecDelay;
          --display state on LHS on 7 seg
            case state is
                When S0 => Count_OneSecDelay_MSD <= 0;
                When S1 => Count_OneSecDelay_MSD <= 1;
                When S2 => Count_OneSecDelay_MSD <= 2;
                When S3 => Count_OneSecDelay_MSD <= 3;
                When others =>
            end case;
      ----------------------------------------------------------------
      --Requirement #6
        else --When key is pressed
         ---pedestrian crosswalk signal as '0' for stop and '1' for walking on the sev
en segments. You can use
        ---one segment for the East-West direction and other for the North-South direc
tion pedestrian crosswalk signals.
            if (state = S0) then
                Count_OneSecDelay <= 1;
            else
                Count_OneSecDelay <= 0;
            end if;

            if (state = S2) then
               Count_OneSecDelay <= 1;
            else
               Count_OneSecDelay <= 0;
            end if;
        end if;
      ----------------------------------------------------------------

        if select_segment='1' then
            digit_7seg_display<= Count_OneSecDelay_LSD;
        else
            digit_7seg_display<= Count_OneSecDelay_MSD;
        end if;

        CC<=select_segment;

    end process;
```

Simulation Results

Figure 7: Simulation for requirement 6



The test case for this was toggling the key press from KeyPad_Row = "1111" to KeyPad_Row = "0111"

As we can see we are in state 2 so the Most significant digit is zero and the Least significant digit is 1 so we can see the following:

1) We get out7seg = 1011011 to 0111111 meaning the display changes from 2 to 0   C=0 meaning segment 1
2) We get out7seg = 1011011 to 0110000 meaning the display changes from 2 to 1   C=1 meaning segment 2

Conclusion

This was the toughest and the most interesting lab out of the three labs. We spent a lot of time on Requirement 6 trying to figure out how the JE pins works with the keypad. However when we figured it out it turned out to be the most interesting part of the lab. Overall we learned a lot from this lab . It was nice to be able to implement a design that we knew could actually be used in a real life scenario. This lab also allowed us to revisit a lot of concepts from the first two labs. We programs a good number of the components on the FPGA. After completing the lab we can confidently say that we can program an FPGA using VHDL.

Appendix

Constraint File:

```
## This file is a general .xdc for the Zybo Z7 Rev. B
## It is compatible with the Zybo Z7-20 and Zybo Z7-10
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level
signal names in the project


##Clock signal
set_property -dict { PACKAGE_PIN K17   IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_
L12P_T1_MRCC_35 Sch=sysclk
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];



##Switches
set_property -dict { PACKAGE_PIN G15   IOSTANDARD LVCMOS33 } [get_ports { sw[0] }]; #I
O_L19N_T3_VREF_35 Sch=sw[0]
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { sw[1] }]; #I
O_L24P_T3_34 Sch=sw[1]
set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports { sw[2]}]; #IO
_L4N_T0_34 Sch=sw[2]
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { sw[3]}]; #IO
_L9P_T1_DQS_34 Sch=sw[3]


##Buttons
set_property -dict { PACKAGE_PIN K18   IOSTANDARD LVCMOS33 } [get_ports { btn[0]}]; #I
O_L12N_T1_MRCC_35 Sch=btn[0]
set_property -dict { PACKAGE_PIN P16   IOSTANDARD LVCMOS33 } [get_ports { btn[1] }]; #
IO_L24N_T3_34 Sch=btn[1]
set_property -dict { PACKAGE_PIN K19   IOSTANDARD LVCMOS33 } [get_ports { btn[2] }]; #
IO_L10P_T1_AD11P_35 Sch=btn[2]
set_property -dict { PACKAGE_PIN Y16   IOSTANDARD LVCMOS33 } [get_ports { btn[3] }]; #
IO_L7P_T1_34 Sch=btn[3]


##LEDs
set_property -dict { PACKAGE_PIN M14   IOSTANDARD LVCMOS33 } [get_ports { led[0] }]; #
IO_L23P_T3_35 Sch=led[0]
set_property -dict { PACKAGE_PIN M15   IOSTANDARD LVCMOS33 } [get_ports { led[1]}]; #I
O_L23N_T3_35 Sch=led[1]

set_property -dict { PACKAGE_PIN G14   IOSTANDARD LVCMOS33 } [get_ports { red_led[0]}]
; #IO_0_35 Sch=led[2]
set_property -dict { PACKAGE_PIN D18   IOSTANDARD LVCMOS33 } [get_ports { red_led[1] }
]; #IO_L3N_T0_DQS_AD1N_35 Sch=led[3]

##RGB LED 6
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { led6_r }]; #
IO_L18P_T2_34 Sch=led6_r
#To resolve implementation related error.
#[Place 30-876] Port 'btn[0]'  is assigned to PACKAGE_PIN 'K18'  which can only be use
d as the N side of a differential clock input.
```

```
#Please use the following constraint(s) to pass this DRC check:
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets {led6_r_OBUF}]

set_property -dict { PACKAGE_PIN F17   IOSTANDARD LVCMOS33 } [get_ports { led6_g  }];
#IO_L6N_T0_VREF_35 Sch=led6_g

set_property -dict { PACKAGE_PIN M17   IOSTANDARD LVCMOS33 } [get_ports { led6_b }]; #
IO_L8P_T1_AD10P_35 Sch=led6_b


##Pmod Header JC

set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33     } [get_ports { out_7seg
[0] }]; #IO_L10P_T1_34 Sch=jc_p[1]
set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33     } [get_ports { out_7seg
[1] }]; #IO_L10N_T1_34 Sch=jc_n[1]
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33     } [get_ports { out_7seg
[2] }]; #IO_L1P_T0_34 Sch=jc_p[2]
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33     } [get_ports { out_7seg
[3] }]; #IO_L1N_T0_34 Sch=jc_n[2]


##Pmod Header JD

set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33     } [get_ports { out_7seg
[4] }]; #IO_L5P_T0_34 Sch=jd_p[1]
set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33     } [get_ports { out_7seg
[5] }]; #IO_L5N_T0_34 Sch=jd_n[1]
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33     } [get_ports { out_7seg
[6] }]; #IO_L6P_T0_34 Sch=jd_p[2]
set_property -dict { PACKAGE_PIN R14   IOSTANDARD LVCMOS33     } [get_ports { CC }]; #
IO_L6N_T0_VREF_34 Sch=jd_n[2]




##Pmod Header JE

set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Col[0
] }]; #IO_L4P_T0_34 Sch=je[1]
set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Col[1
] }]; #IO_L18N_T2_34 Sch=je[2]
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Col[2
] }]; #IO_25_35 Sch=je[3]
set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Col[3
] }]; #IO_L19P_T3_35 Sch=je[4]
set_property -dict { PACKAGE_PIN V13   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Row[0
] }]; #IO_L3N_T0_DQS_34 Sch=je[7]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Row[1
] }]; #IO_L9N_T1_DQS_34 Sch=je[8]
set_property -dict { PACKAGE_PIN T17   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Row[2
] }]; #IO_L20P_T3_34 Sch=je[9]
set_property -dict { PACKAGE_PIN Y17   IOSTANDARD LVCMOS33 } [get_ports { KeyPad_Row[3
] }]; #IO_L7N_T1_34 Sch=je[10]
```

Traffic intersection code

```
----------------------------------------------------------------------------------
-- Company: University of Alberta
-- Engineer: Raza Bhatti
--
-- Create Date: 05/11/2018 11:22:20 AM
-- Design Name:
-- Module Name: traffic_intersection - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-- East/West and North/South intersection working. btn(0) used to see status of lights
 on respective direction of travel.
-- Red light camera on each direction of travel.
-- Night time quick green if red on direction of travel (e.g. North/South or East/West
) and no vehicles on other direction of travel (e.g. North/South or East/West)

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity traffic_intersection is
    Port (
            clk:    in STD_LOGIC;
            btn :   in STD_LOGIC_VECTOR(3 DOWNTO 0);    -- btn(0) press to see traffic
 light status for North/South or East/West lights.
                                                        -- btn(3) press to emulate veh
icle passing from North/South direction, btn(2) for East/West.
            --Write design line here to get inputs from switches, refer to constraints
 file.
                                                        -- SW(0)='1'=>Vehicle present
on East/West direction of travel, SW(1)=>'1' for North/South
                                                        -- SW(3)='1'=> Lgiht Sensor Em
ulation '0'=>Day '1'=>Night
            sw:     in STD_LOGIC_VECTOR(3 DOWNTO 0);

            led6_r : out STD_LOGIC;     --Traffic light status as Red
            led6_g : out STD_LOGIC;     --Traffic light status as Green
            led6_b : out STD_LOGIC;     --Traffic light status as Yello=>Blue on board


            led: out STD_LOGIC_VECTOR(1 downto 0);        --Monitor states [ led(0), l
ed(1) ]
```

```
            red_led: out STD_LOGIC_VECTOR (1 downto 0) :="00";    -- Red Light Camera
[red_led(0), red_led(1) ];
            CC :        out STD_LOGIC;                    --Common cathode input to s
elect respective 7-segment digit.
            out_7seg :  out STD_LOGIC_VECTOR (6 downto 0);  -- Output  signal for sele
cted 7 Segment display.
             ----------------------------------------------------------------
            --Requirement #6
             KeyPad_Col: out STD_LOGIC_VECTOR (3 downto 0) :="0000";
             KeyPad_Row: in STD_LOGIC_VECTOR(3 DOWNTO 0)
              ----------------------------------------------------------------

           );
end traffic_intersection;

architecture Behavioral of traffic_intersection is
component Clock_OneHz is
    port (  clk: in STD_LOGIC;
            clk_1Hz: out STD_LOGIC
          );
end component;

signal clk_1Hz: std_logic;
signal count, Count_OneSecDelay_MSD, Count_OneSecDelay_LSD, digit_7seg_display, count_
7seg : natural;
signal Count_OneSecDelay: natural:=20;
signal states_mon: std_logic_vector(1 downto 0):="00";


TYPE STATES IS (S0,S1,S2,S3,S4,S5,S6);
signal state: STATES:=S0;

-- You can use following signals to implement design requirements or make your own.
signal NTSwitch: std_logic:='0';
signal VehiclesPresence: std_logic_vector(1 downto 0);
signal red_light_camera: std_logic_vector(1 downto 0):="00";
signal Count_RedLight: natural:=0;
signal blinking:STD_LOGIC:='0';
signal clk_out: std_logic:='0';
signal select_segment, clk_7seg_cc:std_logic:='0';

begin

    Decoder_4to7Segment: process (clk)
    begin

    -- Update following case statement to display complete range of digit_7seg_display
 values on 7-segments.
        case digit_7seg_display is
                       when 0 =>
                          out_7seg<="0111111";
            when 1 =>
                          out_7seg<="0110000";
            when 2 =>
                          out_7seg<="1011011";
            when 3 =>
                          out_7seg<="1111001";
            when 4 =>
                          out_7seg<="1110100";
```

```
            when 5 =>
                          out_7seg<="1101101";
            when 6 =>
                          out_7seg<="1101111";
            when 7 =>
                          out_7seg<="0111000";
            when 8 =>
                          out_7seg<="1111111";
            when 9 =>
                          out_7seg<="1111101";

            when others =>

    end case;



    -- End of your design lines.
    end process;


    --Instantiate components
    clock_1Hz: process(clk)
    begin
        if rising_edge(clk) then
            if(count<2) then   --2 for testbench, 125000000 for board
                count<=count+1;
            else
                count<=0;
                clk_out<=not clk_out;
                clk_1Hz<=clk_out;
            end if;

            if (count_7seg<10000) then
                count_7seg<=count_7seg+1;
            else
                select_segment<=not select_segment;
                count_7seg<=0;
            end if;
        end if;
    end process;

    Select_7Segment: process (clk,clk_1Hz,select_segment)
    begin
      ----------------------------------------------------------------
            --Requirement #5
         if (KeyPad_Row = "1111") then       --When no key is pressed, KeyPad_Row="111
1";
            --display state on RHS of 7 seg
            Count_OneSecDelay_LSD <= Count_OneSecDelay;
          --display state on LHS on 7 seg
            case state is
                When S0 => Count_OneSecDelay_MSD <= 0;
                When S1 => Count_OneSecDelay_MSD <= 1;
                When S2 => Count_OneSecDelay_MSD <= 2;
                When S3 => Count_OneSecDelay_MSD <= 3;
                When others =>
            end case;
      ----------------------------------------------------------------
```

```
      --Requirement #6
        else --When key is pressed
         ---pedestrian crosswalk signal as '0' for stop and '1' for walking on the sev
en segments. You can use
        ---one segment for the East-West direction and other for the North-South direc
tion pedestrian crosswalk signals.
            if (state = S0) then
                Count_OneSecDelay_MSD <= 1;
            else
                Count_OneSecDelay_MSD <= 0;
            end if;

            if (state = S2) then
                Count_OneSecDelay_LSD <= 1;
            else
                Count_OneSecDelay_LSD <= 0;
            end if;
        end if;
      ----------------------------------------------------------------

        if select_segment='1' then
            digit_7seg_display<= Count_OneSecDelay_LSD;
        else
            digit_7seg_display<= Count_OneSecDelay_MSD;
        end if;

        CC<=select_segment;

    end process;


 TrafficIntersection: process (clk, clk_1Hz)
   begin
     ----------------------------------------------------------------
    --Requirement #4 Night Time Operation You can write design lines here to capture v
ehicles presence and Night Time input (LDR).
    -- Write your design line here to update VehiclesPresence(0)
    VehiclesPresence(0) <= sw(0);
    -- Write your design line here to update VehiclesPresence(1)
    VehiclesPresence(1) <= sw(1);
    -- Write your design line here to update NTSwitch
    NTSwitch <= sw(3);
    --End of design lines.
  ----------------------------------------------------------------

        if btn(1)='1' then
            state<=S0;
        end if;

        if rising_edge(clk_1Hz) then
            Count_OneSecDelay<=Count_OneSecDelay-1;     --Increment one second count.
~1.84 sec delay here

            case state is
                when S0 =>                              --East/West direction light gr
een
                        if Count_OneSecDelay>0 then
                            if btn(0)='0' then               --Since only have one RGB
light, else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
```

```
                                led6_r<='0';
                                led6_b<='0';
                                led6_g<='1';
                            else
                                led6_r<='1';
                                led6_b<='0';
                                led6_g<='0';
                            end if;
                ----------------------------------------------------------------
                --    Requirement #4 Night Time Operation
                            if (NTSwitch = '1' and VehiclesPresence(1) = '1' and btn(2
) = '0') then

                                state<= S1;
                                Count_OneSecDelay <= 2;
                             end if;
                ----------------------------------------------------------------

                        else
                            state<=S1;
                            Count_OneSecDelay<=2;
                            if btn(0)='0' then         --Since only have one RGB light
, else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                                led6_r<='0';
                                led6_b<='1';
                                led6_g<='0';
                            else
                                led6_r<='1';
                                led6_b<='0';
                                led6_g<='0';
                            end if;
                        end if;

                    states_mon<="00";
                    -- ~1.7 sec delay here

                when S1 =>                             --East/West direction light yel
low=>blue on board
                        if Count_OneSecDelay>0 then
                            if btn(0)='0' then         --Since only have one RGB light
, else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                                led6_r<='0';
                                led6_b<='1';
                                led6_g<='0';
                            else
                                led6_r<='1';
                                led6_b<='0';
                                led6_g<='0';
                            end if;
                        else
                            state<=S2;
                            Count_OneSecDelay<=9;
                            if btn(0)='0' then          --Since only have one RGB ligh
t, else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                                led6_r<='1';
                                led6_b<='0';
                                led6_g<='0';
                            else
                                led6_r<='0';
                                led6_b<='0';
```

```
                                led6_g<='1';
                            end if;
                        end if;
                    states_mon<="01";

                when S2 =>                              -- East/West direction light r
ed and North/South direction green.
                        if Count_OneSecDelay>0 then
                            if btn(0)='0' then          --Since only have one RGB ligh
t, else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                                led6_r<='1';
                                led6_b<='0';
                                led6_g<='0';
                            else
                                led6_r<='0';
                                led6_b<='0';
                                led6_g<='1';
                            end if;
                ----------------------------------------------------------------
                --    Requirement #4 Night Time Operation
                            if (NTSwitch = '1' and VehiclesPresence(0) = '1' and btn(3
) = '0') then
                                state<= S3;
                                Count_OneSecDelay <= 2;
                             end if;
                ----------------------------------------------------------------

                        else
                            state<=S3;
                            Count_OneSecDelay<=2;
                            if btn(0)='0' then            --Since only have one RGB li
ght, else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                                led6_r<='1';
                                led6_b<='0';
                                led6_g<='0';
                            else
                                led6_r<='0';
                                led6_b<='1';
                                led6_g<='0';
                            end if;
                        end if;

                    states_mon<="10";

                    -- ~1.7 sec delay here

                when S3 =>
                        if Count_OneSecDelay>0 then
                            if btn(0)='0' then            --Since only have one RGB li
ght, else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                                led6_r<='1';
                                led6_b<='0';
                                led6_g<='0';
                            else
                                led6_r<='0';
                                led6_b<='1';
                                led6_g<='0';
                            end if;
                        else
```

```
                            state<=S0;
                            Count_OneSecDelay<=9;
                            if Count_OneSecDelay>0 then
                                if btn(0)='0' then              --Since only have one
RGB light, else no need. btn(0)='0' => East/West  btn(0)='1'=> North/South
                                    led6_r<='0';
                                    led6_b<='0';
                                    led6_g<='1';
                                else
                                    led6_r<='1';
                                    led6_b<='0';
                                    led6_g<='0';
                                end if;
                            end if;
                          end if;
                        states_mon<="11";

                        -- ~1.7 sec delay here

                when others =>                      --Error condition
                        state<=S0;
                        Count_OneSecDelay<=9;
                        led6_r<='1';
                        led6_b<='1';
                        led6_g<='1';
            end case;
         end if;

    end process;

---------------------------------------------------------------
   --Requirement #3 When a vehicle approaching intersection encounters a red light
    -- Write a process for Red Light Camera feature at the intersection.
    -- Hint: Since a flash light demo is desired, you can write a process to turn LED
ON and another for OFF, in respective direction of travel.
    -- Start of your design

    -- End of your design
    Red_light_Flash_WE: process(btn(2)) --EAST WEST
    begin
    case btn(2) is
      when '1'  =>
        if (state = S2 and btn(0)='0') or ( state = S3 and btn(0)='0') then
            red_led(0)<= '1';
        end if;
      when '0'  =>
        red_led(0)<= '0';
      when others =>
    end case;
    end process;

    Red_light_Flash_NS: process(btn(3))-- NORTH SOUTH
    begin
        case btn(3) is
          when '1'  =>
            if (state = S0 and  btn(0)='1') or  (state = S1 and btn(0)='1') then
                red_led(1)<= '1';
            end if;
          when '0'  =>
```

```
            red_led(1)<= '0';
          when others =>
        end case;
    end process;

    led<=states_mon;


end Behavioral;
```

Clock divider component to testbench

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 09/20/2019 02:35:38 PM
-- Design Name:
-- Module Name: clock_divider - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
--
Dependencies--------------------------------------------------------------------------
--------
-- Company:
-- Engineer:
--
-- Create Date: 09/20/2019 02:35:38 PM
-- Design Name:
-- Module Name: clock_divider - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity clock_divider is
    Port ( clk : in STD_LOGIC;
           clk_out : out STD_LOGIC);
```

```
end clock_divider;

architecture Behavioral of clock_divider is
signal clock_out : std_logic;
signal count : std_logic_vector (31 downto 0);

begin
     process(clk)
        begin
        if clk = '1' and clk'event then
            if count < 2 then  --2 for testbench, 125000000 = 1 second  for board
                count <= count + 1;
                clock_out <= '0';
            else
                count <= (others=>'0');
                clock_out<= '1';
            end if;


            clk_out<=clock_out;

         end if;
       end process ;

end Behavioral;
:
--
----------------------------------------------------------------------------------



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity clock_divider is
    Port ( clk : in STD_LOGIC;
           clk_out : out STD_LOGIC);
end clock_divider;

architecture Behavioral of clock_divider is
signal clock_out : std_logic;
signal count : std_logic_vector (31 downto 0);

begin
     process(clk)
        begin
        if clk = '1' and clk'event then
            if count < 2 then  --2 for testbench, 125000000 = 1 second  for board
                count <= count + 1;
                clock_out <= '0';
            else
                count <= (others=>'0');
                clock_out<= '1';
            end if;


            clk_out<=clock_out;

         end if;
```

```
        end process ;

end Behavioral;
```

Traffic intersection code changed to work for testbench

```
--Instantiate components
clock_1Hz: process(clk)
begin
    if rising_edge(clk) then
        if(count<2) then   --2 for testbench, 125000000 for board
            count<=count+1;
        else
            count<=0;
            clk_out<=not clk_out;
            clk_1Hz<=clk_out;
        end if;

        if (count_7seg<10000) then
            count_7seg<=count_7seg+1;
        else
            select_segment<=not select_segment;
            count_7seg<=0;
        end if;
    end if;
end process;
```

Testbench

```
-- -- Company:
-- Engineer:
--
-- Create Date: 11/08/2019 02:28:39 PM
-- Design Name:
-- Module Name: traffic_intersection_tb - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
--use IEEE.STD_LOGIC_1164.ALL;


-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity traffic_intersection_tb is
--  Port ( );
end traffic_intersection_tb;

architecture Behavioral of traffic_intersection_tb is
--Add traffic_intersection component

component traffic_intersection is
    Port (
            clk:    in STD_LOGIC;
            btn :   in STD_LOGIC_VECTOR(3 DOWNTO 0);    -- btn(0) press to see traffic
 light status for North/South or East/West lights.
                                                        -- btn(3) press to emulate veh
icle passing from North/South direction, btn(2) for East/West.
            --Write design line here to get inputs from switches, refer to constraints
 file.
                                                        -- SW(0)='1'=>Vehicle present
on East/West direction of travel, SW(1)=>'1' for North/South
                                                        -- SW(3)='1'=> Lgiht Sensor Em
ulation '0'=>Day '1'=>Night
            sw:     in STD_LOGIC_VECTOR(3 DOWNTO 0);

            led6_r : out STD_LOGIC;     --Traffic light status as Red
            led6_g : out STD_LOGIC;     --Traffic light status as Green
            led6_b : out STD_LOGIC;     --Traffic light status as Yello=>Blue on board


            led: out STD_LOGIC_VECTOR(1 downto 0);        --Monitor states [ led(0), l
ed(1) ]
            red_led: out STD_LOGIC_VECTOR (1 downto 0):="00";    -- Red Light Camera [
red_led(0), red_led(1) ];
            CC :        out STD_LOGIC;                      --Common cathode input to s
elect respective 7-segment digit.
            out_7seg :  out STD_LOGIC_VECTOR (6 downto 0);  -- Output  signal for sele
cted 7 Segment display.
            KeyPad_Col: out STD_LOGIC_VECTOR (3 downto 0) :="0000";
            KeyPad_Row: in STD_LOGIC_VECTOR(3 DOWNTO 0)
           );
end component;

--Clock component
component clock_divider is
    port (  clk: in STD_LOGIC;
            clk_out: out STD_LOGIC
          );
End component;
```

```
-- input signals
signal clk: std_logic :='0'; --also for clk
signal btn,sw: std_logic_vector(3 downto 0):="0000";

--output signals
signal led6_r,  led6_g, led6_b, CC: std_logic;
signal led, red_led: std_logic_vector(1 downto 0);
signal out_7seg: std_logic_vector(6 downto 0);
signal clk_out: std_logic; -- for clk
signal KeyPad_Col: STD_LOGIC_VECTOR(3 downto 0) :="0000";
signal  KeyPad_Row: STD_LOGIC_VECTOR(3 DOWNTO 0);

Constant clock_period: time:=500ps;

begin
    --initialize
   TI: traffic_intersection Port Map
                (
                clk=>clk,
                btn=>btn,
                sw=>sw,
               led6_r=>led6_r,
               led6_g=>led6_g,
               led6_b=>led6_b,
               led=>led,
               red_led=>red_led,
               CC=>CC,
               out_7seg=>out_7seg,
               KeyPad_Row=> KeyPad_Row,
               KeyPad_Col=> KeyPad_Col
         );
    -- initialize

    divider: clock_divider port map
    (
            clk=>clk,
            clk_out=> clk_out
          );

    clock: process
    begin
        clk <='0';
        wait for clock_period/2;
        clk <='1';
        wait for clock_period/2;
    end process;
 ---------------------------------------------------------------
 ---NOTE:
   --Input
      -- SW(0)='1' =>Vehicle present on East/West direction
      -- SW(1)='1' =>Vehicle present on North/South direction
      --btn(0)='1' =>Shows the Led,r,g,b in N/S
      --btn(1)='1' =>Reset to state zero on clk change
  ---------------------------------------------------------------
   --Output
       --Led shows the state
       --Led,r,g,b shows the light in E/W by default
    ---------------------------------------------------------------
```

```
    simulation: process
    begin
    ------------------------Requirement #2 Normal Simulation--------------------------
-----------

        wait for 100ns; --then continue

        btn<= "0001";

        wait for 50ns; --holding the input

        btn<= "0000";

        wait for 100ns;
    ----------------------- Requirement #3 Red Light Flash----------------------------
----------

--        Vehicle approaches intersection so flash_red if state is 2 or 3(red light)

        btn<= "0000";

        wait for 50ns;

        btn(2) <= '1'; -- Vehicle crossing on East/West (Output: red_led(0)=>1, at sta
tes: 2, 3)

        wait for 50ns;

        btn(2) <= '0';

        wait for 50ns;

 --        Vehicle approaches intersection so flash_red if state is 0 or 1(red light)

        btn<= "0001";

        wait for 50ns;

        btn(3) <= '1'; -- Vehicle crossing on North West (Output: red_led(1)=>1, at st
ates: , 0, 1)

        wait for 50ns;

        btn(3) <= '0';

        wait for 50ns;

        ---------------------Requirement #4 Night Time Operation----------------------
---------------------
        --1)
        btn<= "0000";
        sw<= "0000";

        wait for 50ns;

        --SW(3) to simulate the light sensor
        sw(3) <= '1';
```

```
        wait for 50ns;

        sw(0) <= '1'; -- vehicle's presence in the East or West direction

        --should encounter a red light, we should then change it to green sw(1) is not
 1(no vehicle in N/S)

        --checking that the other direction is red
        wait for 20ns;
        btn<= "0001";

        wait for 50ns;

        --2)
        btn<= "0001";
        sw<= "0000";

       wait for 50ns;

        --SW(3) to simulate the light sensor
        sw(3) <= '1';

        wait for 50ns;

        sw(1) <= '1'; -- vehicle's presence in the North or South direction

        --should encounter a red light, we should then change it to green sw(0) is not
 1(no vehicle in E/W)

        --checking that the other direction is red
        wait for 20ns;

        btn<= "0000";

        wait for 50ns;

    ---------------------------------------------------------------
    --Requirement #6
--Display the pedestrian crosswalk signal as '0' for stop and '1' for walking on the s
even segments. You can use
--one segment for the East-West direction and other for the North-South direction pede
strian crosswalk signals.
        btn<= "0000";
        sw<= "0000";

        wait for 25ns;

        KeyPad_Row <= "1111"; -- key is not pressed

        wait for 25ns;

        KeyPad_Row <= "0111"; -- key is not pressed

        wait for 25ns;
    ---------------------------------------------------------------
    end Process;

end Behavioral;
```