Engineering 32B

Traffic Light Controller Final Report

Bench: 3

Writer(s): Tyler LaVorgna, Ashley Garcia Cervantes, Fatima Zavala Espinal

Experiment Performed Date: 11/14-28/2022

Report Written Date: 12/12 /2022

Instructor: Dr. Kwong

Abstract

The objective of this experiment was to study and implement the operation of a traffic light that includes a time delay for light changes, a highway road counter to allow pedestrian crossing, and a farm road left turn signal to allow cars to turn left using the ISE Design Suite Schematic and VHDL.

I. Project Specification and Description

The traffic light controller initially implemented had both the highway light change from yellow to red and farm road light change from red to green simultaneously. While this is functional, it is not particularly efficient for moving the flow of traffic. In order to modify the controller for more efficient performance, a time delay was added such that both lights remained red for some time before switching to their respective color.

Using the same state diagram as in the initial design, new states were added to the light to change from highway yellow to highway red, farm road red to farm road green, highway red to farm road green, and farm road yellow to farm road red. The addition of these implements a time delay in the mentioned states. Both the farm road and highway lights stay red for some minimal time when the highway is red and the farm road is getting ready to turn green. This is repeated for when the farm road is red and the highway is getting ready to turn green. Then, the state diagram's logic was used to implement the code onto the Digilent FPGA board.

In order to further enhance the operation of the traffic light for optimal user operation, a down counter was implemented for pedestrian crossing to the highway road and a left turn signal was added to the farm road. According to figure 1, the state diagram was modified to allow these changes and a new bit was used for the counter. When the sensor detects that there is a car in the farm road, the left turn signal turns on and checks if there is a person there (sw1 = 1) in order to start the counter. When the switch is on, the counter starts and as long as the farm road is green, the counter should go on and stop at farm road yellow so that the person can finish crossing.

II. Procedure

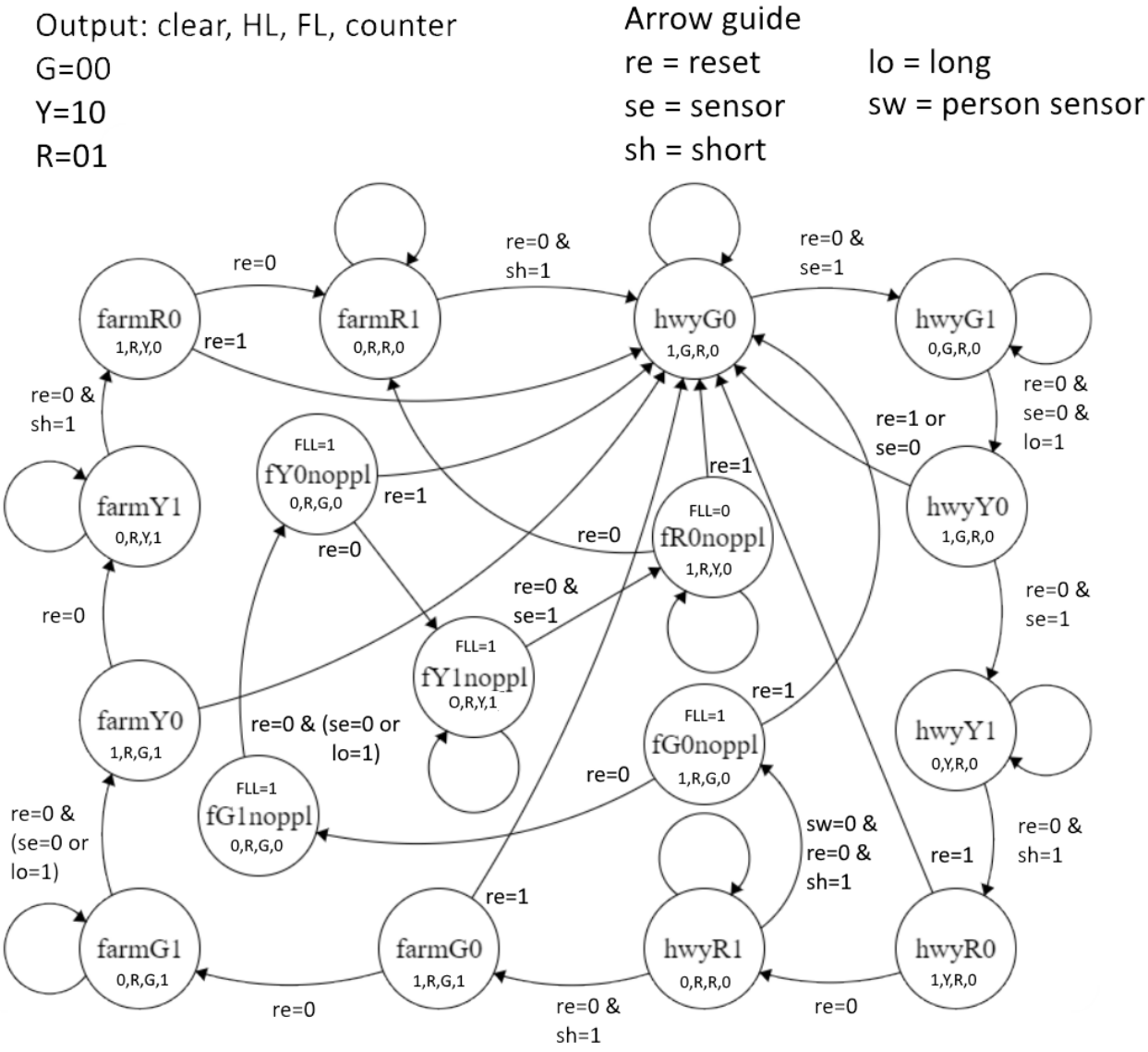


Figure 1 - State Diagram

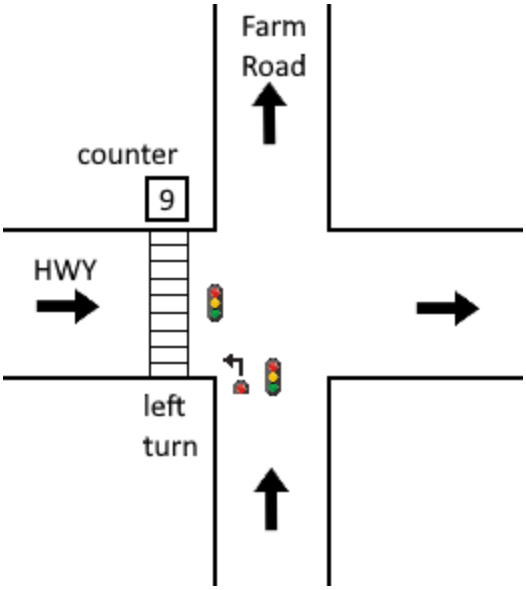


Figure 2: Visual representation of what the traffic light controller is and the added improvements.

Table 1: FPGA Pin Assignment CHANGE with variables from TOP UCF File below

Signal name	**FPGA Pin**	**Peripheral**
HR	N11	LD5
HY	M11	LD4
HG	V15	LD3
FR	U15	LD2
FY	V16	LD1
FG	U16	LD0
Clock	V10	Internal Clock
Sensor	T10	SW0
Person Sensor	T11	SW1
Reset	C9	BTND
s(6)	T17	CA
s(5)	T18	CB
s(4)	U17	CC
s(3)	U18	CD
s(2)	M14	CE
s(1)	N14	CF
s(0)	L14	CG
anode(3)	P17	AN3
anode(2)	P18	AN2
anode(1)	N15	AN1
anode(0)	N16	AN0

```
NET HR LOC=N11 | IOSTANDARD=LVCMOS33; #LD5
NET HY LOC=M11 | IOSTANDARD=LVCMOS33; #LD4
NET HG LOC=V15 | IOSTANDARD=LVCMOS33; #LD3
NET FR LOC=U15 | IOSTANDARD=LVCMOS33; #LD2
NET FY LOC=V16 | IOSTANDARD=LVCMOS33; #LD1
NET FG LOC=U16 | IOSTANDARD=LVCMOS33; #LD0

Net FLL LOC=T11 | IOSTANDARD=LVCMOS33; #LD7

NET clock LOC=V10 | IOSTANDARD=LVCMOS33;
NET sensor LOC=T10 | IOSTANDARD=LVCMOS33; #SW0
NET reset LOC=C9 | IOSTANDARD=LVCMOS33; #BTND

Net s<6> LOC=T17 | IOSTANDARD=LVCMOS33;
Net s<5> LOC=T18 | IOSTANDARD=LVCMOS33;
Net s<4> LOC=U17 | IOSTANDARD=LVCMOS33;
Net s<3> LOC=U18 | IOSTANDARD=LVCMOS33;
Net s<2> LOC=M14 | IOSTANDARD=LVCMOS33;
Net s<1> LOC=N14 | IOSTANDARD=LVCMOS33;
Net s<0> LOC=L14 | IOSTANDARD=LVCMOS33;

Net anode<3> LOC=P17 | IOSTANDARD=LVCMOS33;
Net anode<2> LOC=P18 | IOSTANDARD=LVCMOS33;
Net anode<1> LOC=N15 | IOSTANDARD=LVCMOS33;
Net anode<0> LOC=N16 | IOSTANDARD=LVCMOS33;

Net sw1 LOC=T9 | IOSTANDARD=LVCMOS33; #SW1
```

Figure 3: Top ucf File for complete Schematic of Traffic Light Controller

```
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.STD_LOGIC_UNSIGNED.ALL;
23  entity slow_clock is
24   Port ( clock, clear : in STD_LOGIC;
25   speed : out STD_LOGIC_VECTOR (5 downto 0) );
26  end slow_clock;
27  architecture Behavioral of slow_clock is
28  Signal cnt: STD_LOGIC_VECTOR (30 downto 0);
29  begin
30  process(clock, clear)
31  begin
32  if clear='1' then cnt(30 downto 25) <= "000000";
33  elsif (clock'event and clock='0') then cnt <= cnt + 1;
34  end if;
35  end process;
36  speed <= cnt(30 downto 25);
37  end Behavioral;
38
39
```

Figure 4: VHDL Code for Slow Counter

```
19  ----------------------------------------------------------------------------------
20
21  library IEEE;
22  use IEEE.STD_LOGIC_1164.ALL;
23
24
25  entity bcddecoder is
26      Port ( d : in  STD_LOGIC_VECTOR (3 downto 0);
27             enable : in  STD_LOGIC;
28             s : out  STD_LOGIC_VECTOR (6 downto 0);
29             anode : out  STD_LOGIC_VECTOR (3 downto 0));
30  end bcddecoder;
31
32  architecture Behavioral of bcddecoder is
33  begin
34       anode <= "111" & (not enable);
35
36
37  s <=       "0000100" when d="0000" else
38             "0000000" when d="0001" else
39             "0001111" when d="0010" else
40             "0100000" when d="0011" else
41             "0100100" when d="0100" else
42             "1001100" when d="0101" else
43             "0000110" when d="0110" else
44             "0010010" when d="0111" else
45             "1001111" when d="1000" else
46             "0000001";
47
48
49
50  end Behavioral;
51
52
```

Figure 5: VHDL Code for the BCD Decoder

```
19  ----------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.STD_LOGIC_ARITH.ALL;
23  use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25
26  entity decadecounter is
27      Port( clock : in std_logic;
28            clear : in std_logic;
29
30
31            count : out std_logic_vector(3 downto 0)
32
33            );
34  end decadecounter;
35
36
37  architecture Behavioral of decadecounter is
38     signal cnt : std_logic_vector(30 downto 0);
39
40  begin
41     process(clock, clear)
42        begin
43           if clear='1' then
44              cnt(30 downto 27) <= "0000";
45           elsif (clock'event and clock='0') then
46              cnt <= cnt + 1;
47          end if;
48
49
50     end process;
51
52     count <= cnt(30 downto 27);
53  end Behavioral;
```

Figure 6: VHDL Code for the Decade Counter

```
20  --state_machine.vhd begins here--
21  library IEEE;
22  use IEEE.STD_LOGIC_1164.ALL;
23  use IEEE.STD_LOGIC_ARITH.ALL;
24  use IEEE.STD_LOGIC_UNSIGNED.ALL;
25  entity state_machine is
26     Port ( clock, reset, sensor, long, short, swl : in std_logic;
27            HL1, HL0, FL1, FL0, clear, counter, FLL : out std_logic );
28  end state_machine;
29
30  architecture Behavioral of state_machine is
31     type state_type is ( hwygreen0, hwygreen1, hwyyellow0, hwyyellow1, hwyred0, hwyred1, farmgreen0, farmgreen1, farmyellow0, farmyellow1, farmred0,
32   attribute enum_encoding: string;
33   attribute enum_encoding of state_type: type is "0000000000000001 0000000000000010 0000000000000100 0000000000001000 0000000000010000 00000000
34     signal CS, NS: state_type;
35
36  begin
37     process (clock, reset)
38     begin
39        if (reset='1') then
40           CS <= hwygreen0;
41        elsif rising_edge(clock) then
42           CS <= NS;
43        end if;
44     end process;
45
46     process (CS, reset, sensor, long, short, swl)
47     begin
48        case CS is
49           when hwygreen0 =>
50              HL1 <= '0'; HL0 <= '0';
51              FL1 <= '0'; FL0 <= '1';
52              clear <= '1';
53              counter <= '0'; --new
54           if (reset='0' and (sensor='1' or swl = '1')) then
55              NS <= hwygreen1;
56           else
57              NS <= hwygreen0;
58           end if;
```

```
27
28
29
30
31  mred1, farmgreen0NoPPL, farmgreen1NoPPL, farmyellow0NoPPL, farmyellow1NoPPL, farmred0NoPPL );
32
33  00000 0000000000001000000 0000000001000000 0000000010000000 0000000100000000 0000010000000000 0000100000000000 0001000000000000 0010000000000000 0100000000000000 1000000000000000" ;
34
```

The code above represent the additional states added to the state diagram on the right of lines 31-33.

```
60          when hwygreen1 =>
61             HL1 <= '0'; HL0 <= '0';
62             FL1 <= '0'; FL0 <= '1';
63             clear <= '0';
64             counter <= '0'; --new
65             if (reset='0' and (sensor='1' or sw1 = '1') and long='1' ) then
66                NS <= hwyyellow0;
67             else
68                NS <= hwygreen1;
69          end if;
70
71          when hwyyellow0 =>
72             HL1 <= '0'; HL0 <= '0';
73             FL1 <= '0'; FL0 <= '1';
74             clear <= '1';
75             counter <= '0'; --new
76          if (reset='0' and (sensor='1' or sw1 = '1')) then
77             NS <= hwyyellow1;
78          else
79             NS <= hwygreen0;
80          end if;
81
82             when hwyyellow1 =>
83                HL1 <= '1'; HL0 <= '0';
84                FL1 <= '0'; FL0 <= '1';
85                clear <= '0';
86                counter <= '0'; --new
87             if (reset='0' and short='1') then
88                NS <= hwyred0;
89             else
90                NS <= hwyyellow1;
91             end if;
92
93    when hwyred0 =>
94     HL1 <= '1'; HL0 <= '0';
95     FL1 <= '0'; FL0 <= '1';
96     clear <= '1';
97     counter <= '0'; --new
98     if (reset='0') then
99     NS <= hwyred1;
100    else
101    NS <= hwygreen0;
```

The code below shows changes that were made to implement the 5 new states where the farm left turn signal (FLL) will shine (i.e. when there are no people waiting to cross the highway road).

```
100    else
101    NS <= hwygreen0;
102    end if;
103
104   when hwyred1 =>
105    HL1 <= '0'; HL0 <= '1';
106    FL1 <= '0'; FL0 <= '1';
107    clear <= '0';
108
109
110    --newwww (didn't fix the issue)
111   if (reset='0' and short='1' and sw1='0') then
112   counter <= '0';
113   NS <= farmgreen0NoPPL;
114   elsif (reset='0' and short='1' and sw1='1') then
115   counter <= '1';
116   NS <= farmgreen0;
117   else
118   NS <= hwyred1;
119   end if;
120
121   --(also didn't fix issue)
122   -- if (reset='0' and short='1') then
123      --if (sw1 = '0') then
124         --counter <= '0';
125         --NS <= farmgreen0NoPPL;
126      --else
127         --counter <= '1';
128         --NS <= farmgreen0;
129      --end if;
130   --else
131      -- NS <= hwyred1;
132   --end if;
133
134
135
```

The issue stated in the code refers to the problem caused in the scenario when a person wants to cross the Highway street but no cars are waiting on the farm road: sw1 = 1 but sw0 = 0 This may have an easy fix, but there is a double if statement or code section that might not be read by the program properly. In the above code, two different ways to implement this code segment were tested but they both resulted in the same unsuccessful result. This makes this

program breakable for this specific scenario, and further analysis regarding the state machine code should be done to find the problem and fix it.

Below is the code implementation for the 5 new states when there are no people waiting to cross the highway, so the farm left-turn signal is on (FLL=1) and the counter remains 0, as we don't want to start the counter in this situation. Adding 5 new states with the same logic as their counterparts, with the exception of the parts mentioned above, makes the program clearer, prevents major errors between the counter and left turn signal, and makes it harder to break.

```
138  --LEFT TURN -NoPPL----------------
139  when farmgreen0NoPPL =>
140     HL1 <= '0'; HL0 <= '1';
141     FL1 <= '0'; FL0 <= '0';
142     clear <= '1';
143     counter <= '0';
144     FLL <= '1';
145     if (reset='0') then
146        NS <= farmgreen1NoPPL;
147     else
148        NS <= hwygreen0;
149     end if;
150
151  when farmgreen1NoPPL =>
152     HL1 <= '0'; HL0 <= '1';
153     FL1 <= '0'; FL0 <= '0';
154     clear <= '0';
155     counter <= '0';
156     FLL <= '1';
157     if (reset='0' and ( sensor='0' or long='1' ) ) then
158        NS <= farmyellow0NoPPL;
159     else
160        NS <= farmgreen1NoPPL;
161     end if;
162
163  when farmyellow0NoPPL =>
164      HL1 <= '0'; HL0 <= '1';
165      FL1 <= '0'; FL0 <= '0';
166      clear <= '1';
167      counter <= '0';
168      FLL <= '1';
169      if (reset='0') then
170      NS <= farmyellow1NoPPL;
171      else
172      NS <= hwygreen0;
173      end if;
174
175
```

```
 when farmyellow1NoPPL =>
    HL1 <= '0'; HL0 <= '1';
    FL1 <= '1'; FL0 <= '0';
    clear <= '0';
    counter <= '0';
    FLL <= '1';
    if (reset='0' and short='1') then
    NS <= farmred0NoPPL;
    else
    NS <= farmyellow1NoPPL;
    end if;

 when farmred0NoPPL =>
    HL1 <= '0'; HL0 <= '1';
    FL1 <= '1'; FL0 <= '0';
    clear <= '1';
    counter <= '0';
    FLL <= '0';
    if (reset='0') then
    NS <= farmred1;
    else
    NS <= hwygreen0;
    end if;
-----------------------------------------------
          when farmgreen0 =>
             HL1 <= '0'; HL0 <= '1';
             FL1 <= '0'; FL0 <= '0';
             clear <= '1';
             counter <= '1';
          if (reset='0') then
             NS <= farmgreen1;
          else
             NS <= hwygreen0;
          end if;
```

```
         when farmgreen1 =>
            HL1 <= '0'; HL0 <= '1';
            FL1 <= '0'; FL0 <= '0';
            clear <= '0';
            counter <= '1'; --new
         if (reset='0' and ( sensor='0' or long='1' ) ) then
            NS <= farmyellow0;
         else
            NS <= farmgreen1;
         end if;

when farmyellow0 =>
 HL1 <= '0'; HL0 <= '1';
 FL1 <= '0'; FL0 <= '0';
 clear <= '1';
counter <= '1';
 if (reset='0') then
 NS <= farmyellow1;
 else
 NS <= hwygreen0;
 end if;
```

```
233   when farmyellow1 =>
234    HL1 <= '0'; HL0 <= '1';
235    FL1 <= '1'; FL0 <= '0';
236    clear <= '0';
237    counter <= '1';
238    if (reset='0' and short='1') then
239    NS <= farmred0;
240    else
241    NS <= farmyellow1;
242    end if;
243
244    when farmred0 =>
245    HL1 <= '0'; HL0 <= '1';
246    FL1 <= '1'; FL0 <= '0';
247    clear <= '1';
248    counter <= '0';
249    if (reset='0') then
250    NS <= farmred1;
251    else
252    NS <= hwygreen0;
253    end if;
254
255   when farmred1 =>
256    HL1 <= '0'; HL0 <= '1';
257    FL1 <= '0'; FL0 <= '1';
258    clear <= '0';
259    counter <= '0';
260    if (reset='0' and short='1') then
261    NS <= hwygreen0;
262    else
263    NS <= farmred1;
264    end if;
265
266    end case;
267    end process;
268   end Behavioral;
269
```

Figure 6: VHDL Code for the State Machine (pg. 11-16)

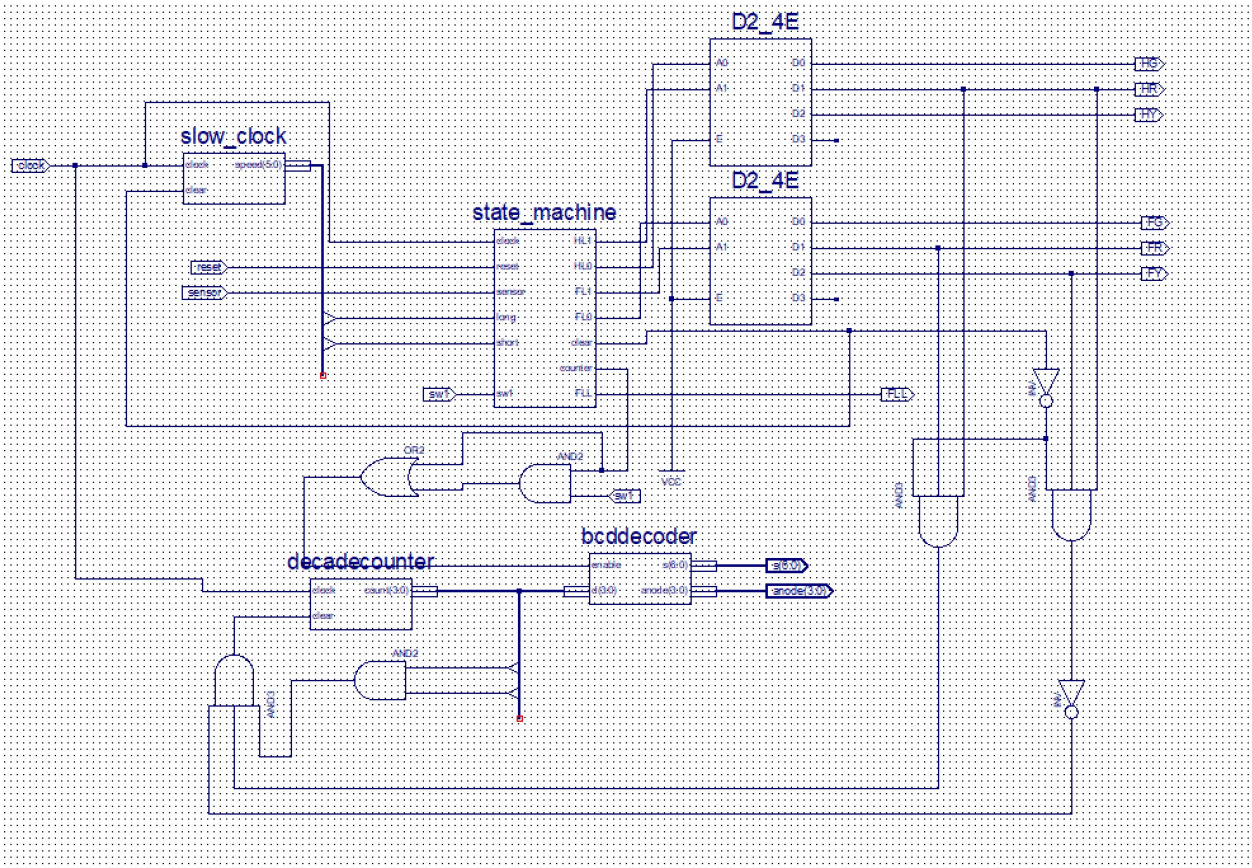
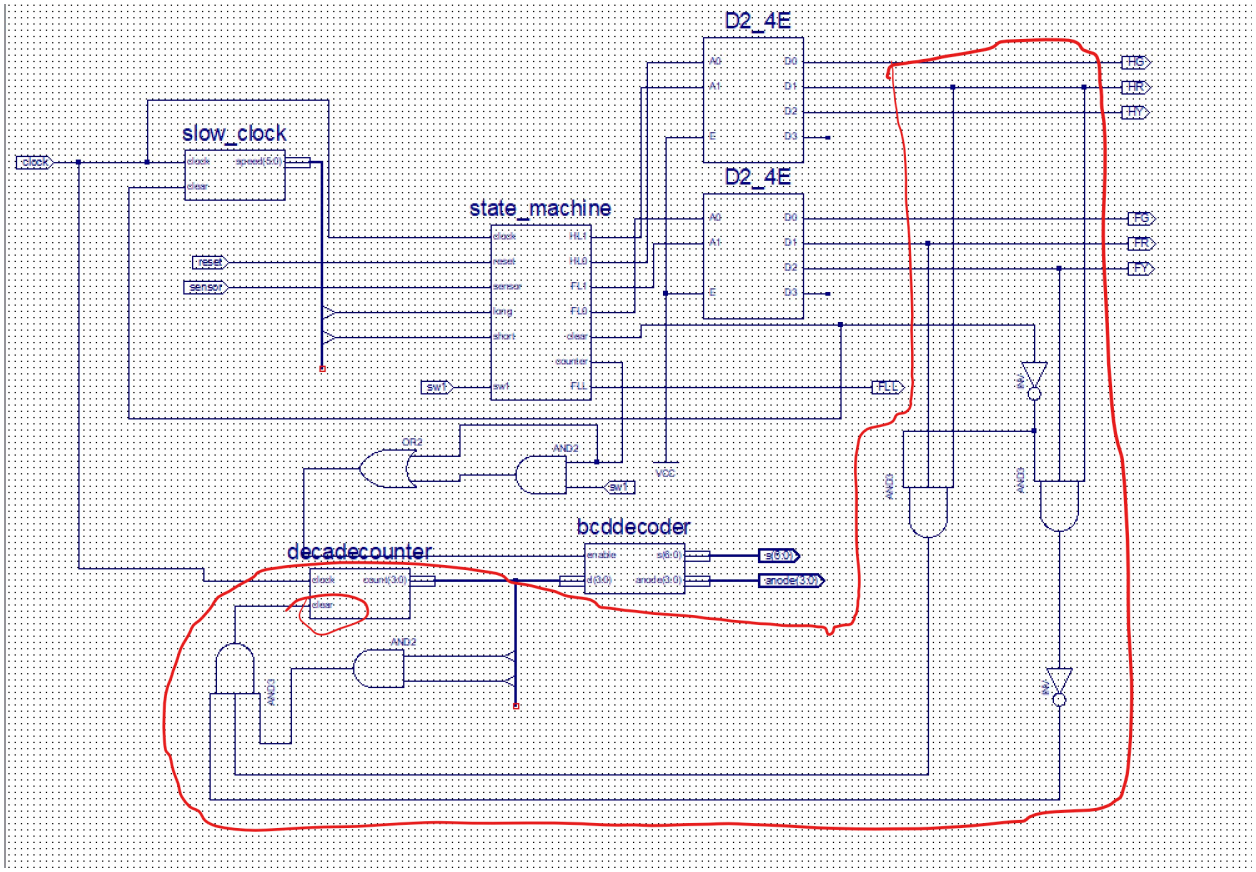
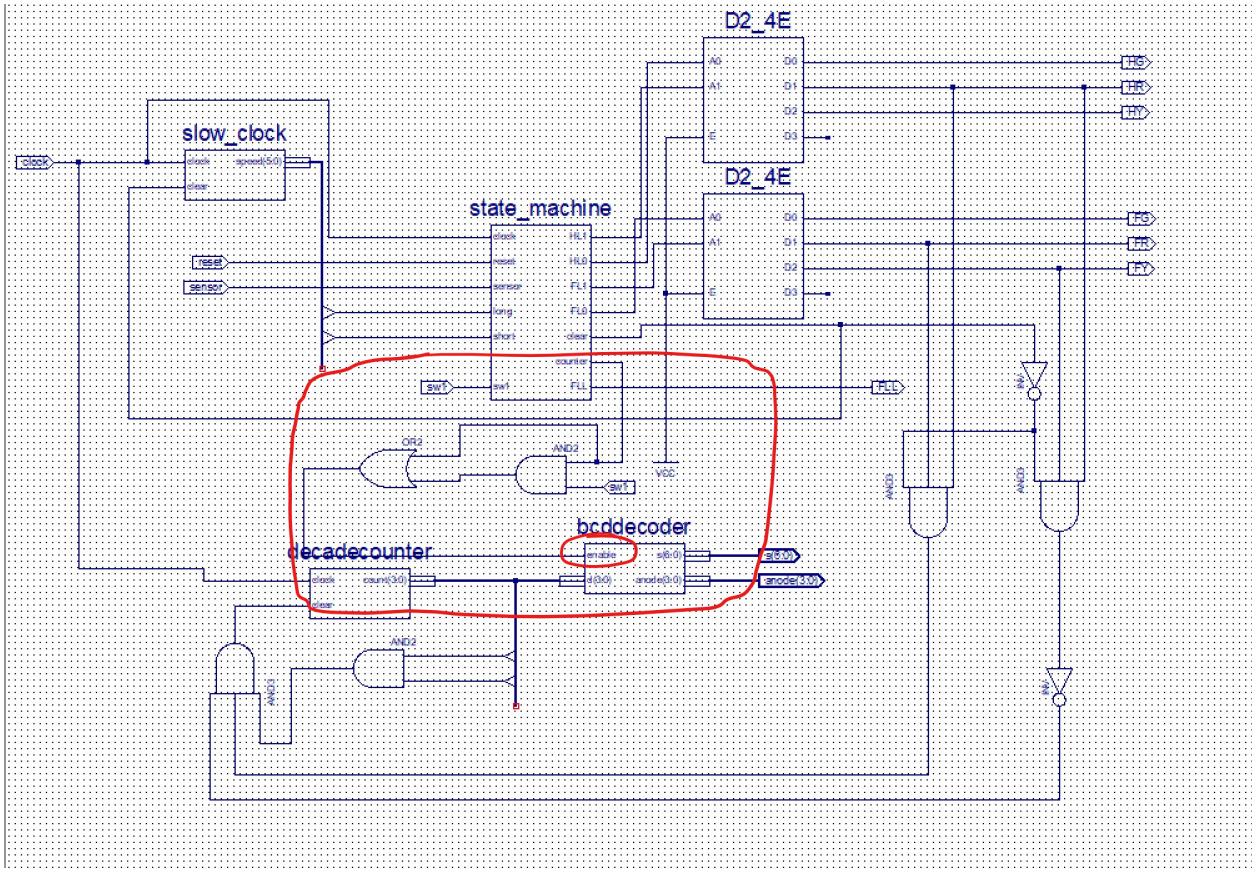


Figure 7: Complete Schematic for the Improved Traffic light Controller



The gates indicated in red were implemented to start and stop the down counter



The gates indicated in red were implemented to enable the BCD decoder when the counter is logic 1 (active) OR if the counter and switch (sw1) are both logic 1, meaning a person is present and wants to cross the highway road. The reason why we do this is because we want the counter to keep counting and showing as long as there is a person sensed before the farm left turn signal is on (FLL=1) (i.e. suppose that in the beginning (at state hwyred1) sw1=1, then that will mean a person wants to cross the highway street. As long as the sw1 was 1 at that time the down counter must start and be displayed as normal, even if the switch shifts to sw1=0 in mid count). This helps prevent further breakable scenarios.

III. Conclusion

The traffic light was successfully implemented with the left turn signal and the down counter for pedestrian crossing making it almost unbreakable. There was, however, a conflict when there was a person waiting to cross the highway, but no cars in the farm road, so the counter did not start in this scenario since it was not thought of, but with more time and further analysis of the code, this problem could have been fixed.