

# Experiment No. 8

## Traffic Light Controller

### ECE 446

Peter CHINETTI

December 8, 2014

Date Performed: September 28, 2014  
Instructor: Professor Shanechi

## 1 Introduction

Typically, traffic light control is a field dominated by micro-controllers. Very complex timing of these lights can be handled with some simple software running on off-the-shelf products. For very simple traffic lights, however, it may be possible to implement their timing functionality with simple sequential logic circuits

For complex, modern traffic lights, often, the use of a micro-controller to handle all the timing functionality of the light is essential. Coordinating the three main signal lamps for each direction, left and right turn arrows, and walk and don't walk signals is not an easy task. It may also be beneficial to alter the timing of the light based upon traffic sensors and time of day. Handling all this functionality is simply too daunting a task for simple logic circuits. In addition, the programming capabilities of a micro-controller allow the timing to be altered relatively easily when necessary. With this said, there are some simple applications in which the cost of developing a micro-controller system and its corresponding software would simply not be justified. For these situations, a simple sequential logic circuit may be constructed to provide all the required functionality.

## 2 Procedure

- a. Write VHDL to implement simple intersection.
- b. Assign pins to ports
- c. Simulate

- d. Program and Test
- e. Write VHDL to implement sensed intersection.
- f. Assign pins to ports
- g. Simulate
- h. Program and Test

### 3 Equipment

- PC
- Spartan-3E development board

## 4 Code

### 4.1 Simple Intersection

#### 4.1.1 Top-level Module

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4
5 entity fsm is
6     Port ( clk_in : in  STD_LOGIC;
7           NS : out  STD_LOGIC_VECTOR (2 downto 0);
8           EW : out  STD_LOGIC_VECTOR (2 downto 0));
9 end fsm;
10
11 architecture Behavioral of fsm is
12
13     type state is (S0, S1, S2, S3, S4, S5);
14     signal f : state;
15     signal p : state := S2;
16     signal clk : STD_LOGIC;
17     signal t1 : STD_LOGIC;
18     signal t5 : STD_LOGIC;
19     signal t10 : STD_LOGIC;
20     signal reset : STD_LOGIC;
21
22     component selectable_clock
23     Port ( clk : in  std_logic;
24           s0 : in  std_logic;
25           s1 : in  std_logic;
26           out_clk : out std_logic);
27 end component;
28
29     component timer
30     Port (clk : in  std_logic;
31           reset : in  std_logic;

```

```

33     t1      : out std_logic;
34     t5      : out std_logic;
35     t10     : out std_logic);
36 end component;
37 begin
38     clk_0 : selectable_clock
39     port map(
40         clk => clk_in ,
41         s0 => '0',
42         s1 => '1',
43         out_clk => clk);
44     timer_0 : timer
45     port map(
46         clk => clk ,
47         reset => reset ,
48         t1 => t1 ,
49         t5 => t5 ,
50         t10 => t10);
51
52
53 process (clk)
54 begin
55     if rising_edge(clk) then
56         p <= f;
57     end if;
58 end process;
59
60
61 process (p, t1, t5, t10)
62 begin
63     case p is
64     when S0 =>
65         if t10 = '1' then
66             reset <= '1';
67             f <= S1;
68         else
69             reset <= '0';
70             f <= S0;
71         end if;
72     when S1 =>
73         if t1 = '1' then
74             reset <= '1';
75             f <= S2;
76         else
77             reset <= '0';
78             f <= S1;
79         end if;
80     when S2 =>
81         if t1 = '1' then
82             reset <= '1';
83             f <= S3;
84         else
85             reset <= '0';
86             f <= S2;
87         end if;
88     when S3 =>

```

```

89     if t5 = '1' then
90         reset <= '1';
91         f <= S4;
92     else
93         reset <= '0';
94         f <= S3;
95     end if;
96 when S4 =>
97     if t1 = '1' then
98         reset <= '1';
99         f <= S5;
100    else
101        reset <= '0';
102        f <= S4;
103    end if;
104 when S5 =>
105     if t1 = '1' then
106         reset <= '1';
107         f <= S0;
108     else
109         reset <= '0';
110         f <= S5;
111     end if;
112 end case;
113 end process;

115 process (p)
116 begin
117     case p is
118     when S0 =>
119         NS <= "001";
120         EW <= "100";
121     when S1 =>
122         NS <= "010";
123         EW <= "100";
124     when S2 =>
125         NS <= "100";
126         EW <= "100";
127     when S3 =>
128         NS <= "100";
129         EW <= "001";
130     when S4 =>
131         NS <= "100";
132         EW <= "010";
133     when S5 =>
134         NS <= "100";
135         EW <= "100";
136     end case;
137 end process;
end Behavioral;

```

1/fsm.vhd

#### 4.1.2 Timer

```
library IEEE;
```

```

2 use IEEE.STD_LOGIC_1164.ALL;
  use IEEE.NUMERIC_STD.ALL;

4
  entity timer is
6     Port ( clk : in  STD_LOGIC;
            t1  : out  STD_LOGIC;
            t5  : out  STD_LOGIC;
            t10 : out  STD_LOGIC;
            reset : in  STD_LOGIC);
10    end timer;

12
  architecture Behavioral of timer is
14    signal counter: unsigned( 3 downto 0) := "0000";

16  begin
    process(clk , reset)
18    begin
        if rising_edge(clk) then
20        if reset = '1' then
            counter <= "0000";
22        elsif counter <= 10 then
            counter <= counter + "0001";
24        end if;
        end if;
26    end process;

28    process(counter)
    begin
30        if counter > 1 then
            t1 <= '1';
32        else
            t1 <= '0';
34        end if;
        if counter > 5 then
36        t5 <= '1';
        else
38        t5 <= '0';
        end if;
40        if counter > 10 then
            t10 <= '1';
42        else
            t10 <= '0';
44        end if;
        end process;
46  end Behavioral;

```

1/timer.vhd

### 4.1.3 Clock Divider

```

1 — Selectable output frequency clock divider code.
  library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;
  use IEEE.STD_LOGIC_ARITH.ALL;
5 use IEEE.STD_LOGIC_UNSIGNED.ALL;
  entity selectable_clock is

```

```

7   Port ( clk : in std_logic;
9         s0 : in std_logic;
11        s1 : in std_logic;
12        out_clk : out std_logic);
13 end selectable_clock;
14 --If s1 and s0 are both low, the output clock rate is 1/10 Hz.
15 --If s1 is low and s0 is high, the output clock rate is 1 Hz.
16 --If s1 is high and s0 is low, the output clock rate is 10 Hz.
17 --If s1 and s0 are both high, the output clock rate is 1 KHz.
18 architecture Behavioral of selectable_clock is
19 begin
20     process (clk, s0, s1)
21     variable count : integer := 0;
22     begin
23         if clk = '1' and clk'event then
24             count := count + 1;
25             -- Start a process.
26             -- Variable declaration.
27             -- Rising edge detection.
28             -- Code to create the 1/10 Hz clock.
29             if s0 = '0' and s1 = '0' then
30                 if count >= 500000000 then
31                     -- Taken off a 50MHz clock.
32                     count := 0;
33                     -- Reset count for next cycle.
34                     end if;
35                     if count >= 0 and count <= 250000000 then
36                         out_clk <= '1';
37                         -- High portion of 1/10 HZ clock.
38                     else
39                         out_clk <= '0';
40                         -- Low portion of 1/10 HZ clock.
41                     end if;
42                 end if;
43             -- Code to create the 1 Hz clock.
44             if s0 = '1' and s1 = '0' then
45                 if count >= 50000000 then
46                     -- Taken off a 50MHz clock.
47                     count := 0;
48                     -- Reset count for next cycle.
49                     end if;
50                     if count >= 0 and count <= 25000000 then
51                         out_clk <= '1';
52                         -- High portion of 1 HZ clock.
53                     else
54                         out_clk <= '0';
55                         -- Low portion of 1 HZ clock.
56                     end if;
57                 end if;
58             -- Code to create the 10 Hz clock.
59             if s0 = '0' and s1 = '1' then
60                 if count >= 5000000 then
61                     -- Taken off a 50MHz clock.
62                     count := 0;
63                     -- Reset count for next cycle.
64                     end if;
65                 if count >= 0 and count <= 2500000 then

```

```

        out_clk <= '1';
65     -- High portion of 10 HZ clock.
        else
67         out_clk <= '0';
        -- Low portion of 10 HZ clock.
69     end if;
    end if;
71     -- Code to create the 1 KHz clock.
    if s0 = '1' and s1 = '1' then
73         if count >= 50000 then
            -- Taken off a 50MHz clock.
75             count := 0;
            -- Reset count for next cycle.
77             end if;
            if count >= 0 and count <= 25000 then
79                 out_clk <= '1';
                -- High portion of 1 KHz clock.
81                 else
                    out_clk <= '0';
83                 -- Low portion of 1 KHz clock.
                    end if;
85             end if;
            end if;
87     end process;
end Behavioral;

```

1/clock.div.vhd

#### 4.1.4 Test

```

LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;

4  -- Uncomment the following library declaration if using
   -- arithmetic functions with Signed or Unsigned values
6  --USE ieee.numeric_std.ALL;

8  ENTITY test IS
   END test;

10 ARCHITECTURE behavior OF test IS

12     -- Component Declaration for the Unit Under Test (UUT)

14     COMPONENT timer
16     PORT(
18         clk : IN  std_logic;
         t1  : OUT  std_logic;
         t5  : OUT  std_logic;
20         t10 : OUT  std_logic;
         reset : IN  std_logic
22     );
   END COMPONENT;

24

26     --Inputs

```

```

28     signal clk : std_logic := '0';
       signal reset : std_logic := '0';

30     --Outputs
       signal t1 : std_logic;
32     signal t5 : std_logic;
       signal t10 : std_logic;

34     -- Clock period definitions
36     constant clk_period : time := 10 ns;

38 BEGIN

40     -- Instantiate the Unit Under Test (UUT)
       uut: timer PORT MAP (
42         clk => clk,
           t1 => t1,
44         t5 => t5,
           t10 => t10,
46         reset => reset
       );

48     -- Clock process definitions
50     clk_process : process
       begin
52         clk <= '0';
           wait for clk_period/2;
54         clk <= '1';
           wait for clk_period/2;
56     end process;

58     -- Stimulus process
60     stim_proc: process
       begin
62         -- hold reset state for 100 ns.
           wait for 100 ns;

64         wait for clk_period*10;

66         -- insert stimulus here
68         wait for clk_period*10;
           wait;
70     end process;

72 END;

```

1/test.vhd

## 4.2 Sensored Intersection

### 4.2.1 Top-level Module

```

library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;

```



```

4
entity fsm is
6   Port ( clk_in : in  STD_LOGIC;
          sens_w , sens_e : in  STD_LOGIC;
          NS : out  STD_LOGIC_VECTOR (2 downto 0);
          EW : out  STD_LOGIC_VECTOR (2 downto 0));
10  end fsm;

12  architecture Behavioral of fsm is

14     type state is (S0, S1, S2, S3, S4, S5, SS);
15     signal f : state;
16     signal p : state := S2;
17     signal clk : STD_LOGIC;
18     signal t1 : STD_LOGIC;
19     signal t5 : STD_LOGIC;
20     signal t10 : STD_LOGIC;
21     signal reset : STD_LOGIC;

22     component selectable_clock
23     Port ( clk : in  std_logic;
           s0 : in  std_logic;
           s1 : in  std_logic;
           out_clk : out std_logic);
28     end component;

30     component timer
31     Port (clk : in  std_logic;
           reset : in  std_logic;
           t1 : out  std_logic;
           t5 : out  std_logic;
           t10 : out  std_logic);
36     end component;
begin

38     clk_0 : selectable_clock
40     port map(
41         clk => clk_in ,
42         s0 => '0',
43         s1 => '1',
44         out_clk => clk);
45     timer_0 : timer
46     port map(
47         clk => clk ,
48         reset => reset ,
49         t1 => t1 ,
50         t5 => t5 ,
51         t10 => t10);
52

54     process (clk)
55     begin
56         if rising_edge(clk) then
57             p <= f;
58         end if;

```

```

60 end process;

62 process (p, t1, t5, t10)
begin
64   case p is
        when S0 =>
66       if sens_w = '0' and sens_e = '0' then
            f <= S0;
68             reset <= '0';
        else
70             reset <= '1';
            f <= SS;
72         end if;
        when SS =>
74             if sens_w = '0' and sens_e = '0' then
                reset <= '0';
76                 f <= S0;
            elsif t10 = '1' then
78                 reset <= '1';
                f <= S1;
80             else
                reset <= '0';
82                 f <= SS;
            end if;
        when S1 =>
84             if t1 = '1' then
                reset <= '1';
86                 f <= S2;
            else
88                 reset <= '0';
                f <= S1;
90             end if;
        when S2 =>
92             if t1 = '1' then
                reset <= '1';
94                 f <= S3;
            else
96                 reset <= '0';
                f <= S2;
98             end if;
        when S3 =>
100             if t5 = '1' then
                reset <= '1';
102                 f <= S4;
            else
104                 reset <= '0';
                f <= S3;
106             end if;
        when S4 =>
108             if t1 = '1' then
                reset <= '1';
110                 f <= S5;
            else
112                 reset <= '0';
                f <= S4;
114             end if;
        when S5 =>
116

```

```

118     if t1 = '1' then
119         reset <= '1';
120         f <= S0;
121     else
122         reset <= '0';
123         f <= S5;
124     end if;
125 end case;
126 end process;

127 process (p)
128 begin
129     case p is
130     when S0 =>
131         NS <= "001";
132         EW <= "100";
133     when S5 =>
134         NS <= "001";
135         EW <= "100";
136     when S1 =>
137         NS <= "010";
138         EW <= "100";
139     when S2 =>
140         NS <= "100";
141         EW <= "100";
142     when S3 =>
143         NS <= "100";
144         EW <= "001";
145     when S4 =>
146         NS <= "100";
147         EW <= "010";
148     when S5 =>
149         NS <= "100";
150         EW <= "100";
151     end case;
152 end process;
end Behavioral;

```

2/fsm.vhd

#### 4.2.2 Timer

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3 use IEEE.NUMERIC_STD.ALL;

5 entity timer is
6     Port ( clk : in  STD_LOGIC;
7           t1  : out  STD_LOGIC;
8           t5  : out  STD_LOGIC;
9           t10 : out  STD_LOGIC;
10          reset : in  STD_LOGIC);
11 end timer;

13 architecture Behavioral of timer is
14     signal counter: unsigned( 3 downto 0) := "0000";

```

```

15 begin
17   process(clk, reset)
18   begin
19     if rising_edge(clk) then
20       if reset = '1' then
21         counter <= "0000";
22       elsif counter <= 10 then
23         counter <= counter + "0001";
24       end if;
25     end if;
26   end process;
27
28   process(counter)
29   begin
30     if counter > 1 then
31       t1 <= '1';
32     else
33       t1 <= '0';
34     end if;
35     if counter > 5 then
36       t5 <= '1';
37     else
38       t5 <= '0';
39     end if;
40     if counter > 10 then
41       t10 <= '1';
42     else
43       t10 <= '0';
44     end if;
45   end process;
46 end Behavioral;

```

2/timer.vhd

### 4.2.3 Clock Divider

```

1  -- Selectable output frequency clock divider code.
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_ARITH.ALL;
5  use IEEE.STD_LOGIC_UNSIGNED.ALL;
6  entity selectable_clock is
7    Port ( clk : in std_logic;
8          s0 : in std_logic;
9          s1 : in std_logic;
10         out_clk : out std_logic);
11 end selectable_clock;
12 --If s1 and s0 are both low, the output clock rate is 1/10 Hz.
13 --If s1 is low and s0 is high, the output clock rate is 1 Hz.
14 --If s1 is high and s0 is low, the output clock rate is 10 Hz.
15 --If s1 and s0 are both high, the output clock rate is 1 KHz.
16 architecture Behavioral of selectable_clock is
17 begin
18   process (clk, s0, s1)
19     variable count : integer := 0;

```

```

begin
21   if clk = '1' and clk'event then
      count := count + 1;
23     -- Start a process.
      -- Variable declaration.
25     -- Rising edge detection.
      -- Code to create the 1/10 Hz clock.
27     if s0 = '0' and s1 = '0' then
          if count >= 500000000 then
29             -- Taken off a 50MHz clock.
              count := 0;
31             -- Reset count for next cycle.
              end if;
33             if count >= 0 and count <= 250000000 then
                  out_clk <= '1';
35             -- High portion of 1/10 HZ clock.
              else
37                 out_clk <= '0';
                  -- Low portion of 1/10 HZ clock.
              end if;
39         end if;
41     -- Code to create the 1 Hz clock.
      if s0 = '1' and s1 = '0' then
43         if count >= 50000000 then
              -- Taken off a 50MHz clock.
              count := 0;
45             -- Reset count for next cycle.
              end if;
47             if count >= 0 and count <= 25000000 then
                  out_clk <= '1';
49             -- High portion of 1 HZ clock.
              else
51                 out_clk <= '0';
                  -- Low portion of 1 HZ clock.
              end if;
53         end if;
55     -- Code to create the 10 Hz clock.
      if s0 = '0' and s1 = '1' then
57         if count >= 5000000 then
              -- Taken off a 50MHz clock.
              count := 0;
59             -- Reset count for next cycle.
              end if;
61             if count >= 0 and count <= 2500000 then
                  out_clk <= '1';
63             -- High portion of 10 HZ clock.
              else
65                 out_clk <= '0';
                  -- Low portion of 10 HZ clock.
              end if;
67         end if;
69     -- Code to create the 1 KHz clock.
      if s0 = '1' and s1 = '1' then
71         if count >= 50000 then
              -- Taken off a 50MHz clock.
              count := 0;
73             -- Reset count for next cycle.
              end if;
75

```

```

77     end if;
79     if count >= 0 and count <= 25000 then
        out_clk <= '1';
        -- High portion of 1 KHz clock.
81     else
        out_clk <= '0';
83     -- Low portion of 1 KHz clock.
        end if;
85     end if;
87     end if;
end process;
end Behavioral;

```

2/clk\_div.vhd

#### 4.2.4 Test

```

LIBRARY ieee;
2 USE ieee.std_logic_1164.ALL;

4 -- Uncomment the following library declaration if using
  -- arithmetic functions with Signed or Unsigned values
6 --USE ieee.numeric_std.ALL;

8 ENTITY test IS
  END test;

10 ARCHITECTURE behavior OF test IS

12     -- Component Declaration for the Unit Under Test (UUT)

14     COMPONENT timer
16     PORT(
18         clk : IN  std_logic;
        t1  : OUT  std_logic;
20         t5  : OUT  std_logic;
        t10 : OUT  std_logic;
        reset : IN  std_logic
22     );
    END COMPONENT;

24

26     --Inputs
    signal clk : std_logic := '0';
28     signal reset : std_logic := '0';

30     --Outputs
    signal t1 : std_logic;
32     signal t5 : std_logic;
    signal t10 : std_logic;

34     -- Clock period definitions
36     constant clk_period : time := 10 ns;

38 BEGIN

```

```

40  -- Instantiate the Unit Under Test (UUT)
    uut: timer PORT MAP (
42      clk => clk,
        t1 => t1,
44      t5 => t5,
        t10 => t10,
46      reset => reset
    );
48
    -- Clock process definitions
50    clk_process : process
    begin
52      clk <= '0';
        wait for clk_period/2;
54      clk <= '1';
        wait for clk_period/2;
56    end process;
58
    -- Stimulus process
60    stim_proc: process
    begin
62      -- hold reset state for 100 ns.
        wait for 100 ns;
64
        wait for clk_period*10;
66
        -- insert stimulus here
68      wait for clk_period*10;
        wait;
70    end process;
72 END;

```

2/test.vhd

## 5 Conclusions

The purpose of this lab was achieved. Two FSM-FPGA based intersection controllers were built and tested. Operation was verified through simulation and physical implementation.