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 sensored 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;
   use IEEE.STD_LOGIC_1164.ALL;
3
5  entity fsm is
       Port ( clk_in : in  STD_LOGIC;
7             NS : out  STD_LOGIC_VECTOR (2 downto 0);
              EW : out  STD_LOGIC_VECTOR (2 downto 0));
9  end fsm;

11 architecture Behavioral of fsm is

13   type state is (S0, S1, S2, S3, S4, S5);
     signal f : state;
15   signal p : state := S2;
     signal clk : STD_LOGIC;
17   signal t1 :  STD_LOGIC;
      signal t5 : STD_LOGIC;
19    signal t10 : STD_LOGIC;
      signal reset :  STD_LOGIC;
21
     component selectable_clock
23   Port ( clk : in std_logic;
            s0 : in std_logic;
25          s1 : in std_logic;
            out_clk : out std_logic);
27   end component;

29   component timer
     Port (clk   : in std_logic;
31       reset : in std_logic;
```

```
      t1     : out std_logic;
      t5     : out std_logic;
      t10    : out std_logic);
  end component;
begin

  clk_0 : selectable_clock
  port map(
    clk => clk_in,
    s0 => '0',
    s1 => '1',
    out_clk => clk);
  timer_0 : timer
  port map(
    clk   => clk,
    reset => reset,
    t1    => t1,
    t5    => t5,
    t10   => t10);


process (clk)
begin

  if rising_edge(clk) then
    p <= f;
  end if;
end process;

process (p, t1, t5, t10)
begin
  case p is
    when S0 =>
      if t10 = '1' then
        reset <= '1';
        f <= S1;
      else
        reset <= '0';
        f <= S0;
      end if;
    when S1 =>
      if t1 = '1' then
        reset <= '1';
        f <= S2;
      else
        reset <= '0';
        f <= S1;
      end if;
    when S2 =>
      if t1 = '1' then
        reset <= '1';
        f <= S3;
      else
        reset <= '0';
        f <= S2;
      end if;
    when S3 =>
```

```
89        if t5 = '1' then
            reset <= '1';
91          f <= S4;
          else
93          reset <= '0';
            f <= S3;
95        end if;
        when S4 =>
97        if t1 = '1' then
            reset <= '1';
99          f <= S5;
          else
101         reset <= '0';
            f <= S4;
103       end if;
        when S5 =>
105       if t1 = '1' then
            reset <= '1';
107         f <= S0;
          else
109         reset <= '0';
            f <= S5;
111       end if;
      end case;
113   end process;

115   process (p)
      begin
117     case p is
          when S0 =>
119         NS <= "001";
            EW <= "100";
121       when S1 =>
            NS <= "010";
123         EW <= "100";
          when S2 =>
125         NS <= "100";
            EW <= "100";
127       when S3 =>
            NS <= "100";
129         EW <= "001";
          when S4 =>
131         NS <= "100";
            EW <= "010";
133       when S5 =>
            NS <= "100";
135         EW <= "100";
      end case;
137   end process;
  end Behavioral;
```

1/fsm.vhd

4.1.2 Timer

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

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

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

begin
  process(clk, reset)
  begin
    if rising_edge(clk) then
      if reset = '1' then
        counter <= "0000";
      elsif counter <= 10 then
        counter <= counter + "0001";
      end if;
    end if;
  end process;

  process(counter)
  begin
    if counter > 1 then
      t1 <= '1';
    else
      t1 <= '0';
    end if;
    if counter > 5 then
      t5 <= '1';
    else
      t5 <= '0';
    end if;
    if counter > 10 then
      t10 <= '1';
    else
      t10 <= '0';
    end if;
  end process;
end Behavioral;
```

1/timer.vhd

4.1.3 Clock Divider

```
-- Selectable output frequency clock divider code.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity selectable_clock is
```

```
  Port ( clk : in std_logic;
         s0 : in std_logic;
         s1 : in std_logic;
         out_clk : out std_logic);
end selectable_clock;
--If s1 and s0 are both low, the output clock rate is 1/10 Hz.
--If s1 is low and s0 is high, the output clock rate is 1 Hz.
--If s1 is high and s0 is low, the output clock rate is 10 Hz.
--If s1 and s0 are both high, the output clock rate is 1 KHz.
architecture Behavioral of selectable_clock is
begin
  process (clk, s0, s1)
    variable count : integer := 0;
  begin
    if clk = '1' and clk'event then
      count := count + 1;
      -- Start a process.
      -- Variable declaration.
      -- Rising edge detection.
      -- Code to create the 1/10 Hz clock.
      if s0 = '0' and s1 = '0' then
        if count >= 500000000 then
          -- Taken off a 50MHz clock.
          count := 0;
        -- Reset count for next cycle.
        end if;
        if count >= 0 and count <= 250000000 then
          out_clk <= '1';
        -- High portion of 1/10 HZ clock.
        else
          out_clk <= '0';
        -- Low portion of 1/10 HZ clock.
        end if;
      end if;
      -- Code to create the 1 Hz clock.
      if s0 = '1' and s1 = '0' then
        if count >= 50000000 then
          -- Taken off a 50MHz clock.
          count := 0;
        -- Reset count for next cycle.
        end if;
        if count >= 0 and count <= 25000000 then
          out_clk <= '1';
        -- High portion of 1 HZ clock.
        else
          out_clk <= '0';
        -- Low portion of 1 HZ clock.
        end if;
      end if;
      -- Code to create the 10 Hz clock.
      if s0 = '0' and s1 = '1' then
        if count >= 5000000 then
          -- Taken off a 50MHz clock.
          count := 0;
        -- Reset count for next cycle.
        end if;
        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/clk_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(
        clk : IN  std_logic;
18      t1 : OUT  std_logic;
        t5 : OUT  std_logic;
20      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;
```

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;
8             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);
   signal f : state;
16   signal p : state := S2;
   signal clk : STD_LOGIC;
18   signal t1 :  STD_LOGIC;
    signal t5 : STD_LOGIC;
20    signal t10 : STD_LOGIC;
    signal reset :  STD_LOGIC;
22
   component selectable_clock
24   Port ( clk : in std_logic;
          s0 : in std_logic;
26          s1 : in std_logic;
          out_clk : out std_logic);
28   end component;

30   component timer
   Port (clk   : in std_logic;
32       reset : in std_logic;
       t1    : out std_logic;
34       t5    : out std_logic;
       t10   : out std_logic);
36   end component;
begin
38
   clk_0 : selectable_clock
40   port map(
     clk => clk_in,
42     s0 => '0',
     s1 => '1',
44     out_clk => clk);
   timer_0 : timer
46   port map(
     clk   => clk,
48     reset => reset,
     t1    => t1,
50     t5    => t5,
     t10   => t10);
52

54 process (clk)
begin
56
   if rising_edge(clk) then
58     p <= f;
   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;
84     when S1 =>
         if t1 = '1' then
86         reset <= '1';
           f <= S2;
88       else
           reset <= '0';
90         f <= S1;
         end if;
92     when S2 =>
         if t1 = '1' then
94         reset <= '1';
           f <= S3;
96       else
           reset <= '0';
98         f <= S2;
         end if;
100    when S3 =>
         if t5 = '1' then
102        reset <= '1';
           f <= S4;
104      else
           reset <= '0';
106        f <= S3;
         end if;
108    when S4 =>
         if t1 = '1' then
110        reset <= '1';
           f <= S5;
112      else
           reset <= '0';
114        f <= S4;
         end if;
116    when S5 =>
```

```
      if t1 = '1' then
        reset <= '1';
        f <= S0;
      else
        reset <= '0';
        f <= S5;
      end if;
    end case;
  end process;

  process (p)
  begin
    case p is
      when S0 =>
        NS <= "001";
        EW <= "100";
      when SS =>
        NS <= "001";
        EW <= "100";
      when S1 =>
        NS <= "010";
        EW <= "100";
      when S2 =>
        NS <= "100";
        EW <= "100";
      when S3 =>
        NS <= "100";
        EW <= "001";
      when S4 =>
        NS <= "100";
        EW <= "010";
      when S5 =>
        NS <= "100";
        EW <= "100";
    end case;
  end process;
end Behavioral;
```

2/fsm.vhd

4.2.2 Timer

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

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

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

```
15
begin
17   process(clk, reset)
     begin
19     if rising_edge(clk) then
         if reset = '1' then
21         counter <= "0000";
         elsif counter <= 10 then
23         counter <= counter + "0001";
         end if;
25     end if;
   end process;
27
   process(counter)
29   begin
       if counter > 1 then
31       t1 <= '1';
       else
33       t1 <= '0';
       end if;
35     if counter > 5 then
         t5 <= '1';
37     else
         t5 <= '0';
39     end if;
       if counter > 10 then
41       t10 <= '1';
       else
43       t10 <= '0';
       end if;
45   end process;
end Behavioral;
```

2/timer.vhd

4.2.3 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;
            s0 : in std_logic;
9           s1 : in std_logic;
            out_clk : out std_logic);
11 end selectable_clock;
   --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.
   --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.
   architecture Behavioral of selectable_clock is
17 begin
     process (clk, s0, s1)
19     variable count : integer := 0;
```

```
begin
  if clk = '1' and clk'event then
    count := count + 1;
    -- Start a process.
    -- Variable declaration.
    -- Rising edge detection.
    -- Code to create the 1/10 Hz clock.
    if s0 = '0' and s1 = '0' then
      if count >= 500000000 then
        -- Taken off a 50MHz clock.
        count := 0;
      -- Reset count for next cycle.
      end if;
      if count >= 0 and count <= 250000000 then
        out_clk <= '1';
      -- High portion of 1/10 HZ clock.
      else
        out_clk <= '0';
      -- Low portion of 1/10 HZ clock.
      end if;
    end if;
    -- Code to create the 1 Hz clock.
    if s0 = '1' and s1 = '0' then
      if count >= 50000000 then
        -- Taken off a 50MHz clock.
        count := 0;
      -- Reset count for next cycle.
      end if;
      if count >= 0 and count <= 25000000 then
        out_clk <= '1';
      -- High portion of 1 HZ clock.
      else
        out_clk <= '0';
      -- Low portion of 1 HZ clock.
      end if;
    end if;
    -- Code to create the 10 Hz clock.
    if s0 = '0' and s1 = '1' then
      if count >= 5000000 then
        -- Taken off a 50MHz clock.
        count := 0;
      -- Reset count for next cycle.
      end if;
      if count >= 0 and count <= 2500000 then
        out_clk <= '1';
      -- High portion of 10 HZ clock.
      else
        out_clk <= '0';
      -- Low portion of 10 HZ clock.
      end if;
    end if;
    -- Code to create the 1 KHz clock.
    if s0 = '1' and s1 = '1' then
      if count >= 50000 then
        -- Taken off a 50MHz clock.
        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;
```

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(
          clk : IN  std_logic;
18        t1 : OUT  std_logic;
          t5 : OUT  std_logic;
20        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.