Experiment No. 7
Introduction to Sequential Circuits
ECE 446

Peter Chinetti

October 14, 2014

Date Performed: September 28, 2014
Instructor: Professor Shanechi

1 Introduction

In addition to purely combinational circuits, VHDL can be used to simulate sequential circuits. In order to illustrate this capabilty, a simple turn signal circuit will be created.

The turn signal uses a state machine to generate it's outputs. Specifcally it uses a Moore machine, as the output depends entirely on the internal state of the machine, not it's inputs. A Mealy machine could also depend on the inputs.

Equations for the finite state machine's lights (if each light was connected to a D flip-flop) is below:

- $D_0 = (\bar{L}R)(\bar{Q_2}\bar{Q_1}\bar{Q_0}) + (LR)(\bar{Q_2}\bar{Q_1}\bar{Q_0}) = (R)(\bar{Q_2}\bar{Q_1}\bar{Q_0})$
- $D_1 = (Q_2\bar{Q_1}Q_0)$
- $D_2 = (\bar{R}L)(\bar{Q_2}\bar{Q_1}\bar{Q_0}) + (LR)(\bar{Q_2}\bar{Q_1}\bar{Q_0}) = (L)(\bar{Q_2}\bar{Q_1}\bar{Q_0})$

2 Procedure

a. Write VHDL to implement finite state machine.

b. Assign pins to ports

c. Simulate

d. Program and Test

3 Equipment

- PC
- Spartan-3E development board

4 Code

4.1 Top-level Module

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


entity FSM is
    Port ( l : in  STD_LOGIC;
           r : in  STD_LOGIC;
           ro : out  STD_LOGIC;
           lo : out  STD_LOGIC; ho : out  STD_LOGIC;
         clk_in : in STD_LOGIC);
end FSM;

architecture Behavioral of FSM is

  type state is (idle, sl, sr, slr, sh);
  signal f : state;
  signal p : state := idle;
  signal clk : STD_LOGIC;

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

begin

  clk_0 : selectable_clock
  port map(
    clk => clk_in,
    s0 => '0',
    s1 => '1',
    out_clk => clk
  );

process (clk)
begin

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

process (p, r, l)
```

```
begin
  case p is
    when idle =>
      if r='0' and l='0' then
        f <= idle;
      elsif r='0' and l='1' then
        f <= sl;
      elsif r='1' and l='0' then
        f <= sr;
      else
        f <= slr;
      end if;
    when sl =>
      f <= idle;
    when sr =>
      f <= idle;
    when slr =>
      f <= sh;
    when sh =>
      f <= idle;
  end case;
end process;

process (p)
begin
  case p is
    when idle =>
      ro <= '0';
      lo <= '0';
      ho <= '0';
    when sl =>
      ro <= '0';
      lo <= '1';
      ho <= '0';
    when sr =>
      ro <= '1';
      lo <= '0';
      ho <= '0';
    when slr =>
      ro <= '1';
      lo <= '1';
      ho <= '0';
    when sh =>
      ro <= '0';
      lo <= '0';
      ho <= '1';
  end case;
end process;

end Behavioral;
```

FSM.vhd

4.2 Test Module

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY fsm_test IS
END fsm_test;

ARCHITECTURE behavior OF fsm_test IS

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

    COMPONENT FSM
    PORT(
         li : IN  std_logic;
         ri : IN  std_logic;
         ro : OUT  std_logic;
         lo : OUT  std_logic;
         ho : OUT  std_logic;
         clk_in : IN  std_logic
        );
    END COMPONENT;


   --Inputs
   signal li : std_logic := '0';
   signal ri : std_logic := '0';
   signal clk_in : std_logic := '0';

  --Outputs
   signal ro : std_logic;
   signal lo : std_logic;
   signal ho : std_logic;

   -- Clock period definitions
   constant clk_in_period : time := 10 ns;

BEGIN

  -- Instantiate the Unit Under Test (UUT)
   uut: FSM PORT MAP (
          li => li,
          ri => ri,
          ro => ro,
          lo => lo,
          ho => ho,
          clk_in => clk_in
        );

   -- Clock process definitions
   clk_in_process :process
   begin
    clk_in <= '0';
    wait for clk_in_period/2;
    clk_in <= '1';
    wait for clk_in_period/2;
   end process;

```

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

64       wait for clk_in_period*10;


66


68       -- insert stimulus here

70       wait;
      end process;
72
END;
```

fsm_test.vhd

4.3 Clock Divider Module

```
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
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.
39          end if;
          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.
45            count := 0;
            -- Reset count for next cycle.
47          end if;
            if count >= 0 and count <= 25000000 then
49            out_clk <= '1';
            -- High portion of 1 HZ clock.
51          else
              out_clk <= '0';
53          -- Low portion of 1 HZ clock.
            end if;
55        end if;
          -- Code to create the 10 Hz clock.
57        if s0 = '0' and s1 = '1' then
            if count >= 5000000 then
59            -- Taken off a 50MHz clock.
              count := 0;
61          -- Reset count for next cycle.
            end if;
63          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;
```

clk_div.vhd

5 Conclusions

The purpose of this lab was achieved. A turn signal finite state machine was built and tested. Operation was verified through simulation and physical implementation.