McGill University
Department of Electrical and Computer Engineering

Course: ECSE-323 Digital System Design **Fall 2008**

Assignment #8

TOPIC: VHDL for Sequential Circuits

Tutorial Session 1 (Monday)

Problem 1

Write a complete VHDL description of a 4-bit signed counter. The circuit should count up by the amount X, where X is a 3-bit input, whenever X is even. The counter should count down by the amount X whenever X is odd. If the count reaches 15 then it should not increment, and if the count reaches 0 it should not decrement. Provide an asynchronous reset which sets the count to 0. Give the circuit a count enable input. Use a single process block.

```
entity counter4 is
     port (rst, clk, c_enable : in std_logic;
           X : in std_logic_vector(2 downto 0);
           COUNT : out integer range 0 to 15);
end counter4;

architecture Q1 of counter4 is
signal tmp : integer range 0 to 15;
begin
     process(rst,clk)
     begin
          if rst='1' then
               tmp <= 0;
          elsif clk='1' and clk'EVENT then
               if C_enable = ‘1’ then
                    if X(0) = ‘0’ then -- even
                         if tmp < 15 then
                          tmp <= tmp+conv_integer(X);
                         end if;
                    else
                         if tmp > 0 then
                          tmp <= tmp-conv_integer(X);
                         end if;
                    end if; -- if X(0)
               end if; -- if c_enable
          end if; -- if rst
     end process;
```

```
          COUNT <= tmp;
     end Q1;
```

Problem 2

Write a complete VHDL description of a Moore FSM that detects occurrences of the 4-bit sequence 1101 in a sequential stream of input bits. The output Z should go high each time the final bit of this sequence is detected. This should take as input a 1-bit value, X.
For example, the output Z for the input stream X(t) = 110110101101 should be 000100100001.

```
     entity Q2 is
          port ( reset, clk, : in std_logic;
               X : in std_logic;
               Z : out std_logic);
     end Q2;

     architecture FSM of Q2 is
     type state_signal is (S0,S1,S11,S110,S1101);
     signal state : state_signal;
     begin

     state_update : process(clk, reset)
          if reset = '1' then
               state <= S0;
          elsif clk = '1' and clk'EVENT then
               case state is
               when S0 =>
                    if X=’0’ then state <= S0;
                    else state <= S1;
                    end if;
               when S1 =>
                    if X=’0’ then state <= S0;
                    else state <= S11;
                    end if;
               when S11 =>
                    if X=’0’ then state <= S110;
                    else state <= S11;
                    end if;
               when S110 =>
                    if X=’0’ then state <= S0;
                    else state <= S1101;
                    end if;
               when S1101 =>
                    if X=’0’ then state <= S0;
                    else state <= S11;
                    end if;
               end case;
          end if; -- if reset
     end process;

     output_logic : process(state)
     begin
               case state is
                    when S1101 => Z <= ‘1’;
                    when others Z <= ‘0’;
               end case;
     end process;
     end FSM;
```

Tutorial Session 2 (Wednesday)

Problem 1

Write a *complete* VHDL description of a 16-bit counter circuit that counts in a "factorial" sequence (1!,2!,3!,4!,5!,6!,7!,8!, repeat…). Use a single process block. Give the circuit an asynchronous reset, and a count enable input.

```
entity fact_counter is
     port (rst, clk, c_enable : in std_logic;
           COUNT : out std_logic_vector(15 downto 0));
end counter4;

architecture Q1 of fact_counter is
signal tmp : std_logic_vector(15 downto 0);
signal max : std_logic_vector(15 downto 0);
begin
     max <= "1001110110000000"; -- 8! = 40320
     process(rst,clk)
     begin
          if rst='1' then
               tmp <= "0000000000000001";
          elsif clk='1' and clk'EVENT then
               if C_enable = '1' then
                    if tmp < max then
                         tmp <= tmp*(tmp+1);
                    else
                         tmp <= "0000000000000001";
                    end if; -- if tmp
               end if; -- if c_enable
          end if; -- if rst
     end process;
     COUNT <= tmp;
end Q1;
```

Problem 2

Write a complete VHDL description of a Moore FSM that has two 1-bit inputs X,Y, and a single output, Z. The output Z should be set high whenever X=Y=1 for 2 consecutive clock cycles, and go low whenever X=Y=0 for 2 consecutive clock cycles, and should hold its value otherwise. Provide an asynchronous reset, which should set Z=0.

```
entity Q2 is
     port ( reset, clk, : in std_logic;
          X, Y : in std_logic;
          Z : out std_logic);
end Q2;
```

```
architecture FSM of Q2 is
type state_signal is (Ra,Rb,S0a,S0b,S0c,S1a,S1b,S1c);
signal state : state_signal;
begin

state_update : process(clk, reset)
     if reset = '1' then
          state <= Ra;
     elsif clk = '1' and clk'EVENT then
          case state is
          when Ra =>
               if (X=’0’ and Y=’0’) then state <= Rb;
               elsif (X=’1’ and Y=’1’) then state <= S0c;
               else state <= Ra;
               end if;
          when Rb =>
               if (X=’0’ and Y=’0’) then state <= S0a;
               elsif (X=’1’ and Y=’1’) then state <= S0c;
               else state <= Ra;
               end if;
          when S0a =>
               if (X=’0’ and Y=’0’) then state <= S0a;
               elsif (X=’1’ and Y=’1’) then state <= S0c;
               else state <= S0b;
               end if;
          when S0b =>
               if (X=’1’ and Y=’1’) then state <= S0c;
               else state <= S0b;
               end if;
          when S0c =>
               if (X=’1’ and Y=’1’) then state <= S1a;
               else state <= S0b;
               end if;
          when S1a =>
               if (X=’0’ and Y=’0’) then state <= S1c;
               elsif (X=’1’ and Y=’1’) then state <= S1a;
               else state <= S1b;
               end if;
          when S1b =>
               if (X=’0’ and Y=’0’) then state <= S1c;
               else state <= S1b;
               end if;
          when S1c =>
               if (X=’0’ and Y=’0’) then state <= S0a;
               else state <= S1b;
               end if;
          end case;
     end if; -- if reset
end process;

output_logic : process(state)
begin
          case state is
               when S1a => Z <= ‘1’;
               when S1b => Z <= ‘1’;
               when S1c => Z <= ‘1’;
               when others Z <= ‘0’;
          end case;
end process;
end FSM;
```

**********   END OF ASSIGNMENT # 9   ****************