McGill University
Department of Electrical and Computer Engineering

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

Assignment #9 Solutions

TOPIC: VHDL for Sequential Circuits

Tutorial Session 1 (Tuesday)

Problem 1

Write a complete VHDL description of a 2-input (X1,X2) 8-bit signed counter which either counts up on every clock cycle or counts down on every clock cycle. The circuit should count up whenever X1 is high and X2 is low and count down whenever X2 is high and X1 is low. The incrementing or decrementing should be disabled (i.e. no counting) whenever X1 and X2 are either both high or both low. If the count reaches 127 then it should not increment, and if the count reaches -127 it should not decrement. Provide an asynchronous reset which sets the count to 0. Use a single process block. Give the circuit a count enable input.

```
entity counter8 is
     port (rst, clk, X1, X2, c_enable : in std_logic;
            COUNT : out integer range -127 to 127);
end counter8;

architecture Q1 of counter8 is
signal tmp : integer range -127 to 127;
begin
     process(rst,clk)
     begin
          if rst='1' then
               tmp <= 0;
          elsif clk='1' and clk'EVENT then
               if C_enable = ‘1’ then
                    if X1= ‘1’ and X2 = ‘0’ then
                         if tmp < 127 then
                          tmp <= tmp+1;
                         end if;
                    elsif X1= ‘0’ and X2 = ‘1’ then
                         if tmp > -127 then
                          tmp <= tmp-1;
                         end if;
                    end if; -- if X1
               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 generates nonsense words (e.g. for computer passwords) out of an alphabet of 4 consonants (B,C,D,F) and 4 vowels (A,E,I,O). This should take as input a 1bit value, X, and output a 3-bit symbol (encoding one of the 8 possible letters). A new letter should be generated on each rising clock edge based on the current letter value according to the following rule:

on Reset set the output to 'A'.
if X = 0
A -> B, or E-> C, or I->D, or O->F, or B->A, or C->E, or D->I, or F->O
if X = 1
A -> D, or E-> F, or I->B, or O->C, or B->E, or C->I, or D->O, or F->A

Write down the output letter sequence for X=11001001.

```
     Let the encoding be A=000, E=001, I=010, O=011, B=100,
     C=101, D=110, F=111

     The output for X=11001001 (after a reset) is [A]DOFOCECI


     entity Q2 is
          port ( reset, clk, : in std_logic;
               X : in std_logic;
               s : out std_logic_vector(2 downto 0));
     end Q2;

     architecture FSM of Q2 is
     type state_signal is (A,E,I,O,B,C,D,F);
     signal state : state_signal;
     begin

     state_update : process(clk, reset)
          if reset = '1' then
               state <= A;
          elsif clk = '1' and clk'EVENT then
               case state is
               when A =>
                    if X='0' then state <= B;
                    else state <= D;
                    end if;
               when E =>
```

```
                    if X=’0’ then state <= C;
                    else state <= F;
                    end if;
               when I =>
                    if X=’0’ then state <= D;
                    else state <= B;
                    end if;
               when O =>
                    if X=’0’ then state <= F;
                    else state <= C;
                    end if;
               when B =>
                    if X=’0’ then state <= A;
                    else state <= E;
                    end if;
               when C =>
                    if X=’0’ then state <= E;
                    else state <= I;
                    end if;
               when D =>
                    if X=’0’ then state <= I;
                    else state <= O;
                    end if;
               when F =>
                    if X=’0’ then state <= O;
                    else state <= A;
                    end if;
               end case;
          end if; -- if reset
     end process;

     output_logic : process(state)
     begin
               s = “000”; -- default output
               case state is
                    when A => s <= “000”;
                    when E => s <= “001”;
                    when I => s <= “010”;
                    when O => s <= “011”;
                    when B => s <= “100”;
                    when C => s <= “101”;
                    when D => s <= “110”;
                    when F => s <= “111”;
               end case;
     end process;
     end FSM;
```

Tutorial Session 2 (Wednesday)

Problem 1

Write a *complete* VHDL description of a 5-bit counter circuit that counts in the "days-in-a-month" sequence (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, repeat…). On every 4th cycle through the sequence the second count value should be 29 instead of 28. Use a single process block. Give the circuit an asynchronous reset, and a count enable input.

```
entity day_counter is
     port (rst, clk, c_enable : in std_logic;
            days : out std_logic_vector(4 downto 0));
end day_counter;

architecture Q1 of day_counter is
signal month : integer range 0 to 11;
signal year : integer range 0 to 3;
begin
     process(rst,clk)
     begin
          if rst='1' then
               month <= 0;
               year <= 0;
          elsif clk='1' and clk'EVENT then
               if C_enable = ‘1’ then
                    month <= month+1;
                    if month = 11 then
                         year <= year+1;
                    end if;
                    case month is
                         when 0 =>
                              if year = 3 then days <= 29;
                              else days <= 28;
                              end if;
                         when 1 => days <= 31;
                         when 2 => days <= 30;
                         when 3 => days <= 31;
                         when 4 => days <= 30;
                         when 5 => days <= 31;
                         when 6 => days <= 31;
                         when 7 => days <= 30;
                         when 8 => days <= 31;
                         when 9 => days <= 30;
                         when 10 => days <= 31;
                         when 11 => days <= 31;
                    end case;
               end if; -- if c_enable
          end if; -- if rst
     end process;
end Q1;
```

Problem 2

Write a complete VHDL description of a Moore FSM that outputs a ‘1’ if the input stream has had four or more ‘01’ or ‘10’ inputs since the last ‘00’ or ‘11’ input (the 2-bit input is examined on each rising clock edge). The output should be ‘0’ otherwise.

```
     entity Q2 is
          port ( reset, clk, : in std_logic;
               X : in std_logic_vector(1 downto 0);
               s : out std_logic);
     end Q2;

     architecture FSM of Q2 is
     type state_signal is (S0, S1, S2, S3, S4);
     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='01' or X=’10’ then state <= S1;
                    else state <= S0;
                    end if;
               when S1 =>
                    if X='01' or X=’10’ then state <= S2;
                    else state <= S0;
                    end if;
               when S2 =>
                    if X='01' or X=’10’ then state <= S3;
                    else state <= S0;
                    end if;
               when S3 =>
                    if X='01' or X=’10’ then state <= S4;
                    else state <= S0;
                    end if;
               when S4 =>
                    if X='01' or X=’10’ then state <= S4;
                    else state <= S0;
                    end if;
               end case;
          end if; -- if reset
     end process;
```

```
output_logic : process(state)
begin
          s = ‘0’; -- default output
          case state is
               when S0 => s <= ‘0’;
               when S1 => s <= ‘0’;
               when S2 => s <= ‘0’;
               when S3 => s <= ‘0’;
               when S4 => s <= ‘1’;
          end case;
end process;
end FSM;
```

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