McGill University
Department of Electrical and Computer Engineering

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

Assignment #3 Solutions

TOPIC: VHDL for Combinational Circuits

Tutorial Session 1 (Tuesday)

1.- Write a complete VHDL description for the Boolean function $f = (a_1a_2 \oplus y)\bullet(\bar{a}_3 + a_4) + \overline{(\bar{y} \oplus a_3a_4)}$. ***Use only simple concurrent assignment statements.***

```
entity Q1 is
  port(y  : in std_logic;
       a  : in std_logic_vector(4 downto 1);
        f : out std_logic);
end Q1;

architecture A1 of Q1 is
begin
  f <= (((a(1) and a(2))xor y) and (not a(3) or a(4))) or
  not( (not y) xor (a(3) and a(4)));
end A1;
```

2.- Write a complete VHDL description for the circuit that implements the function of question 1 using only ***a single selected signal assignment statement.***

Write a truth table for f in terms of a, then group together values of a that give a particular output (i.e. ‘0’, ‘1’, y, or ‘not(y)’). Put the output type with the largest number of inputs into the “when others” clause (which in this case would be (y)).

```
entity Q2 is
  port(y  : in std_logic;
       a  : in std_logic_vector(4 downto 1);
        f : out std_logic);
end Q2;

architecture A2 of Q2 is
begin
  WITH a SELECT
       f <= not(y) WHEN “1111”, ‘1’ WHEN “0011”,
            ‘1’ WHEN “1011”, ‘1’ WHEN “1100”,
                ‘1’ WHEN “1101”, ‘1’ WHEN “1110”,
                not(y) WHEN others;
end A2;
```

3.- Write a complete VHDL description for a circuit that compares a 4-bit input, X, with the binary number 1101. It should give a ‘1’ output when the input is greater than 1101 and ‘0’ otherwise. ***Use a single conditional assignment statement.***

Comparisons are easy to do with a conditional assignment statement. Look at one bit at a time, starting with the MSB. Either you will know the answer, in which case you should output the answer, or you will need to look at the next bit(s) to determine the answer, in which case you go to the next line (after the ELSE).

```
entity Q3 is
  port(X  : in std_logic_vector(3 downto 0);
       f : out std_logic);
end Q3;

architecture A3 of Q3 is
begin
  f <= ‘0’ WHEN X(3)=’0’ ELSE
       ‘0’ WHEN X(2)=’0’ ELSE
       ‘1’ WHEN X(1)=’1’ ELSE
       ‘0’;
end A3;
```

4.- Write a complete VHDL design entity for a 2-input NAND gate module. Then, using ***only*** these NAND gate modules as components, write a complete VHDL design entity of a circuit that implements the Boolean function $f = (a_1 a_2 + (a_3 \oplus a_4))$.

Begin by re-expressing the Boolean function in a Sum-of-Products form: $f = a_1 a_2 + a_3 \overline{a_4} + \overline{a_3} a_4$. We know that (c.f. the textbook page 47) that SOP form can be directly implemented in NAND-NAND form. However, we can use only 2-input NANDs, so the problem is a little more complicated, since we need to implement the 3-input NAND using 2-input NAND gates. This can be done with the following equation:

$\overline{xyz} = \overline{x} + \overline{y} + \overline{z} = \overline{(xy)} + \overline{z} = \overline{\overline{\overline{(xy)}}z} = \text{not}(x \text{ nand } y) \text{ nand } z$, where x,y,z are the inputs to the three input NAND gate. Note that we have to invert the output of (x nand y). We can make an inverter out of a 2-input NAND gate by connecting both of its inputs. We will also need to invert the inputs a3 and a4. Thus we will need a total of 8 NAND gates.

```
entity nand2 is
  port(a : in std_logic;
       f : out std_logic);
end nand2;

architecture q4nand of nand2 is
begin
  f <= a and b;
end q4nand2;

entity Q4 is
  port(a : in std_logic_vector(4 downto 1);
       f    : out std_logic;
end Q4;

architecture A4 of Q4 is
component nand2
  port(a,b : in std_logic;
       f : out std_logic);
end component;
signal Ia3, Ia4, x, y, z, xny, Ixny : std_logic;
begin
  G1 : nand2 port map (a=>a(3),b=>a(3),f=>Ia3);
  G2 : nand2 port map (a=>a(4),b=>a(4),f=>Ia4);
  G3 : nand2 port map (a=>a(1),b=>a(2),f=>x);
  G4 : nand2 port map (a=>a(3),b=>Ia4,f=>y);
  G5 : nand2 port map (a=>Ia3,b=>a(4),f=>z);
  G6 : nand2 port map (a=>x,b=>y,f=>xny);
  G7 : nand2 port map (a=>xny,b=>xny,f=>Ixny);
  G8 : nand2 port map (a=>Ixny,b=>z,f=>f);
end A4;
```

<u>Tutorial Session 2 (Wednesday)</u>

1.- Write a complete VHDL description for the Boolean function $f = (ab + bc + cd + ad) \oplus (b\overline{c}d) + a\overline{b}\overline{c}d$. ***Use only simple concurrent assignment statements.*** Do not minimize the expression, but just implement it as written.

```
entity Q1 is
  port(a, b, c, d  : in std_logic;
          f : out std_logic);
end Q1;

architecture A1 of Q1 is
begin
  f <= ((a and b)or (b and c) or (c and d) or (a and d))
  xor (b and (not c) and d) or (a and (not b) and (not c) and
  d));
end A1;
```

2.- Write a complete VHDL description for the circuit that implements the function of question 1 using only ***a single selected signal assignment statement.***

First write a truth table for f in terms of a,b,c,d then group together values that give a particular output (i.e. ‘0’, ‘1’). Put the output type with the largest number of inputs into the “when others” clause (which in this case would be ‘1’). Create a bus (std_logic_vector) to use as the select input for the selected assignment statement.

```
entity Q2 is
  port( a, b, c, d  : in std_logic;
          f : out std_logic);
end Q2;

architecture A2 of Q2 is
signal ins : std_logic_vector(3 downto 0);
begin
  ins(3) <= a; ins(2) <= b; ins(1) <= c; ins(0) <= d;
  WITH ins SELECT
        f <= ‘0’ WHEN “0000”, ‘0’ WHEN “0001”,
             ‘0’ WHEN “0010”, ‘0’ WHEN “0100”,
             ‘0’ WHEN “1000”,
            ‘0’ WHEN “1010”, ‘0’ WHEN “1101”,
           ‘1’ WHEN others;
end A2;
```

3.- Write a complete VHDL description for a circuit that outputs a ‘1’ when the 6-bit input, X, is divisible by 3, and outputs ‘0’ otherwise. ***Use only a single conditional assignment statement.***

There are many ways to do this, but none of them are very compact. It is simplest just to enumerate each multiple of 3 (there are 22 between 0 and 63):

```
entity Q3 is
  port(X  : in std_logic_vector(5 downto 0);
          f : out std_logic);
end Q3;

architecture A3 of Q3 is
begin
  f <= ‘1’ WHEN X = “000000” ELSE
          ‘1’ WHEN X = “000011” ELSE
          ‘1’ WHEN X = “000110” ELSE
          ‘1’ WHEN X = “001001” ELSE
          ‘1’ WHEN X = “001100” ELSE
          ‘1’ WHEN X = “001111” ELSE
          ‘1’ WHEN X = “010010” ELSE
          ‘1’ WHEN X = “010101” ELSE
          ‘1’ WHEN X = “011000” ELSE
          ‘1’ WHEN X = “011011” ELSE
          ‘1’ WHEN X = “011110” ELSE
          ‘1’ WHEN X = “100001” ELSE
          ‘1’ WHEN X = “100100” ELSE
          ‘1’ WHEN X = “100111” ELSE
          ‘1’ WHEN X = “101010” ELSE
          ‘1’ WHEN X = “101101” ELSE
          ‘1’ WHEN X = “110000” ELSE
          ‘1’ WHEN X = “110011” ELSE
          ‘1’ WHEN X = “110110” ELSE
          ‘1’ WHEN X = “111001” ELSE
          ‘1’ WHEN X = “111100” ELSE
          ‘1’ WHEN X = “111111” ELSE
       ‘0’;
end A3;
```

4.- Using the circuit designed in question 3 as a ***component***, write a complete VHDL description of a circuit that determines whether a 16-bit input, Y, is divisible by 3.

As stated, the problem is extremely difficult! This was not the intention. ***So, this question will not be graded.*** The question should have stated: “…a circuit that determines whether a 16-bit input, Y, is divisible by 3 and by a sum of powers of 2^6 (e.g. 1, 64, 4096)”. To do this one needs to connect 3 of the components of question 3, one connected to the 6 LSBs, one connected to the next 6 LSBs, and the third to the remaining bits. Their outputs should be ANDed together.

```
entity Q4 is
  port(X  : in std_logic_vector(15 downto 0);
        Z : out std_logic);
end Q4;

architecture A4 of Q4 is
component Q3
  port(X  : in std_logic_vector(5 downto 0);
        f : out std_logic);
end component;
signal I1, I2, I3 : std_logic;
begin
  G1 : Q3 port map (X=>X(5 downto 0),f=>I1);
  G2 : Q3 port map (X=>X(11 downto 6),f=>I2);
  G3 : Q3 port map (X=>X(15 downto 12),f=>I3);
  Z <= I1 and I2 and I3;
end A4;
```