ECE 370: Digital Systems-Logic Design

VHDL Sample Lab Exam 1: Solutions
Spring 2005

Design Problem 1:

The figure below shows the schematic for a 4-bit digital system known as an *array multiplier*. Use VHDL to design the 4-bit *array multiplier*.

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

ENTITY arrayMult_4bit IS
     PORT( x, y  : IN  STD_LOGIC_VECTOR( 3 DOWNTO 0 );
           p     : OUT STD_LOGIC_VECTOR( 7 DOWNTO 0 ) );
END arrayMult_4bit;

ARCHITECTURE structure OF arrayMult_4bit IS
     SIGNAL mult  : STD_LOGIC_VECTOR( 0 TO 14 );
     SIGNAL c     : STD_LOGIC_VECTOR( 0 TO 10 );
     SIGNAL s     : STD_LOGIC_VECTOR( 0 TO 5 );
     COMPONENT halfAdder
           PORT( input1, input2    : IN  STD_LOGIC;
                 sum, carryOut     : OUT STD_LOGIC );
     END COMPONENT;
     COMPONENT fullAdder
           PORT( in1, in2, cin     : IN  STD_LOGIC;
                 s, cout           : OUT STD_LOGIC );
     END COMPONENT;
BEGIN

     p(0) <= x(0) AND y(0);        mult(0) <= x(1) AND y(0);
     mult(1) <= x(2) AND y(0);     mult(2) <= x(3) AND y(0);

     mult(3) <= x(0) AND y(1);     mult(4) <= x(1) AND y(1);
     mult(5) <= x(2) AND y(1);     mult(6) <= x(3) AND y(1);

     halfAdder1: halfAdder PORT MAP ( mult(0), mult(3), p(1), c(0) );
     fullAdder1: fullAdder PORT MAP ( mult(1), mult(4), c(0), s(0), c(1) );
     fullAdder2: fullAdder PORT MAP ( mult(2), mult(5), c(1), s(1), c(2) );
     halfAdder2: halfAdder PORT MAP ( mult(6), c(2), s(2), c(3) );

     mult(7) <= x(0) AND y(2);     mult(8) <= x(1) AND y(2);
     mult(9) <= x(2) AND y(2);     mult(10) <= x(3) AND y(2);

     halfAdder3: halfAdder PORT MAP ( mult(7), s(0), p(2), c(4) );
     fullAdder3: fullAdder PORT MAP ( mult(8), s(1), c(4), s(3), c(5) );
     fullAdder4: fullAdder PORT MAP ( mult(9), s(2), c(5), s(4), c(6) );
     fullAdder5: fullAdder PORT MAP ( mult(10), c(3), c(6), s(5), c(7) );

     mult(11) <= x(0) AND y(3);    mult(12) <= x(1) AND y(3);
     mult(13) <= x(2) AND y(3);    mult(14) <= x(3) AND y(3);
```

```
    halfAdder4: halfAdder PORT MAP ( mult(11), s(3), p(3), c(8) );
    fullAdder6: fullAdder PORT MAP ( mult(12), s(4), c(8), p(4), c(9) );
    fullAdder7: fullAdder PORT MAP ( mult(13), s(5), c(9), p(5), c(10) );
    fullAdder8: fullAdder PORT MAP ( mult(14), c(7), c(10), p(6), p(7) );
END structure;
```

Design Problem 2

Design an n-bit carry-look-ahead adder (CLA) using VHDL and create a package. Design a 12-bit carry-look-ahead adder/subtractor **with overflow detection** by instantiating 3, 4-bit instances of the above component by including the package. This creates a 12-bit CLA adder that uses a ripple-carry scheme in 4-bit chunks.

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

ENTITY fastCarryAdder_4bit IS
    PORT( x, y  : IN  STD_LOGIC_VECTOR( 3 DOWNTO 0 );
          sum   : OUT STD_LOGIC_VECTOR( 3 DOWNTO 0 );
          cin   : IN  STD_LOGIC;
          cout  : OUT STD_LOGIC );
END fastCarryAdder_4bit;

ARCHITECTURE dataFlow OF fastCarryAdder_4bit IS
    SIGNAL c : STD_LOGIC_VECTOR( 0 TO 4 );
BEGIN
    c(0) <= cin;
    cout <= c(4);
    FOR i IN 0 to 3 GENERATE
          c(i+1) <= (x(i) AND y(i)) OR (c(i) AND x(i)) OR (c(i) AND y(i));
    END GENERATE;
    sum <= x XOR y XOR c( 3 DOWNTO 0 );
END dataFlow;
```

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

PACKAGE CLA_4bit_package IS
     COMPONENT fastCarryAdder_4bit
           PORT( x, y  : IN  STD_LOGIC_VECTOR( 3 DOWNTO 0 );
                 sum   : OUT STD_LOGIC_VECTOR( 3 DOWNTO 0 );
                 cin   : IN  STD_LOGIC;
                 cout  : OUT STD_LOGIC );
     END COMPONENT;
END CLA_4bit_package;
```

```
LIBRARY ieee;
LIBRARY work;
USE ieee.std_logic_1164.ALL;
USE work.CLA_4bit_package.ALL;

ENTITY adder_sub_12bit IS
   PORT( a, b        : IN  STD_LOGIC_VECTOR( 11 DOWNTO 0 );
         s           : OUT STD_LOGIC_VECTOR( 11 DOWNTO 0 );
         add_sub     : IN  STD_LOGIC;
         cout        : OUT STD_LOGIC;
         overflow    : OUT STD_LOGIC; );
END fastCarryAdder_4bit;

ARCHITECTURE structure OF adder_sub_12bit IS
   COMPONENT fastCarryAdder_4bit
       PORT( x, y    : IN  STD_LOGIC_VECTOR( 3 DOWNTO 0 );
             sum     : OUT STD_LOGIC_VECTOR( 3 DOWNTO 0 );
             cin     : IN  STD_LOGIC;
             cout  : OUT   STD_LOGIC );
   END COMPONENT;
   SIGNAL c     : STD_LOGIC_VECTOR( 0 TO 4 );
   SIGNAL b_hat : STD_LOGIC_VECTOR( 11 DOWNTO 0 );
BEGIN
   carries(0) <= cin;
   cout <= carries(4);
   overflow <= ( a(3) AND b(3) AND NOT(s(3)) ) OR
               ( NOT(a(3)) AND NOT(b(3)) AND s(3) );

   FOR i IN 0 to 2 GENERATE
      b_hat( 4*i+3 DOWNTO 4*i ) <= b( 4*i+3 DOWNTO 4*i ) XOR add_sub;
      CLA_Stages:
      fastCarryAdder_4bit PORT MAP ( a( 4*i+3 DOWNTO 4*i ),
                                     b_hat( 4*i+3 DOWNTO 4*i ),
                                     s( 4*i+3 DOWNTO 4*i ),
                                     c(i), c(i+1) );
   END GENERATE;
END structure;
```

Design Problem 3

Use behavioral VHDL design to implement a 4-bit simplified ALU shown below. The ALU only performs AND, OR, ADD, and SUBTRACT. Your design should have all of the inputs and outputs described above, with appropriate lengths.

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

ENTITY alu_4bit IS
   PORT( a, b        : IN STD_LOGIC_VECTOR( 3 DOWNTO 0 );
         result      : OUT STD_LOGIC_VECTOR( 3 DOWNTO 0 );
         overflow    : OUT STD_LOGIC;
         opCode      : IN STD_LOGIC_VECTOR( 1 DOWNTO 0 ) );
END alu_4bit;

ARCHITECTURE behavior OF alu_4bit IS

   SIGNAL v : STD_LOGIC;
   SIGNAL output : STD_LOGIC_VECTOR( 3 DOWNTO 0 );
   CONSTANT LOGIC_ZERO : STD_LOGIC := '0';

BEGIN

   output <= a AND b WHEN opCode = "00" ELSE
             a OR  b WHEN opCode = "01" ELSE
             a + b   WHEN opCode = "10" ELSE
             a - b;

   v <=  (a(3) AND b(3) AND NOT(output (3)) ) OR
         (NOT(a(3)) AND NOT(b(3)) and output(3) );
      overflow <= v WHEN ( opCode = "01" OR opCode = "10" ) ELSE
                  LOGIC_ZERO;
      result <= output;
END behavior;
```