EEL 4712
Midterm 1 – Spring 2014
VERSION 1

Name: \underline{Solution}

UFID: ________________________________

Sign your name here if you would like for your test to be returned in class:

IMPORTANT:
• Please be neat and write (or draw) carefully. If we cannot read it with a reasonable effort, it is assumed wrong.
• As always, the best answer gets the most points.

COVER SHEET:

<table>
<thead>
<tr>
<th>Problem#</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (15 points)</td>
<td></td>
</tr>
<tr>
<td>2 (7 points)</td>
<td></td>
</tr>
<tr>
<td>3 (15 points)</td>
<td></td>
</tr>
<tr>
<td>4 (6 points)</td>
<td></td>
</tr>
<tr>
<td>5 (6 points)</td>
<td></td>
</tr>
<tr>
<td>6 (20 points)</td>
<td></td>
</tr>
<tr>
<td>7 (10 points)</td>
<td></td>
</tr>
<tr>
<td>8 (16 points)</td>
<td></td>
</tr>
<tr>
<td>9 (5 points)</td>
<td>5</td>
</tr>
</tbody>
</table>

Total: 

Regrade Info:
ENTITY entity_name IS
PORT (input_name, output_name : IN STD_LOGIC; 
     input_vector_name : IN STD_LOGIC_VECTOR(high downto low); 
     bidir_name : INOUT STD_LOGIC; 
     output_name : OUT STD_LOGIC); 
END entity_name;

ARCHITECTURE a OF entity_name IS
SIGNAL signal_name : STD_LOGIC;
BEGIN 
   -- Process Statement 
   -- Concurrent Signal Assignment 
   -- Conditional Signal Assignment 
   -- Selected Signal Assignment 
   -- Component Instantiation Statement 
   END a;

   instance_name: component_name PORT MAP (component_port => connect_port, 
                                          component_port => connect_port);

   WITH expression SELECT 
   signal <= expression WHEN constant_value, 
                expression WHEN constant_value, 
                expression WHEN constant_value, 
                expression WHEN constant_value;
   signal <= expression WHEN boolean_expression ELSE 
                expression WHEN boolean_expression ELSE 
                expression;

   IF expression THEN
   statement;
   statement;
   ELSIF expression THEN
   statement;
   statement;
   ELSE
   statement;
   statement;
   END IF;

   CASE expression IS 
   WHEN constant_value => 
   statement;
   statement;
   WHEN constant_value => 
   statement;
   statement;
   WHEN others => 
   statement;
   statement;
   END CASE;

   <generate_label>: FOR loop_id IN range GENERATE
   -- Concurrent Statement(s)
   END GENERATE;

   type array_type is array(upperbound downto lowerbound);
1) (15 points) Fill in the following behavioral VHDL to implement the illustrated circuit. Assume that clk and rst connect to every register. All wires and operations are width bits. Ignore overflow from the adders.

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity test1 is
generic(
   width : positive := 3);
port (
   clk, rst : in std_logic;
   in1, in2, in3 : in std_logic_vector(width-1 downto 0);
   out1, out2 : out std_logic_vector(width-1 downto 0);
);
end test1;

architecture BHV of test1 is
begin

signal reg1, reg2 : std_logic_vector(width-1 downto 0);

process(clk, rst)
begin
   if (rst = '1') then
      reg1 <= (others => '0');
      reg2 <= (others => '0');
      out1 <= (others => '0');
   elsif (rising_edge(clk)) then
      reg1 <= in1;
      reg2 <= in2;
      out1 <= std_logic_vector(unsigned(reg1) + unsigned(in3));
   end if;
end process;

out1 <= std_logic_vector(unsigned(reg1) + unsigned(reg2));

end BHV;
2) (7 points) Draw the circuit that will be synthesized from the following sequential logic description.
You can omit the clk and rst signals. Just show registers and arithmetic operations. For partial credit
add signal labels to inputs, outputs, and registers.

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity test2 is
    generic (width : positive := 8);
    port (clk, rst : in std_logic;
in1, in2, in3 : in std_logic_vector(width-1 downto 0);
output : out std_logic_vector(2*width-1 downto 0))
end test2;

architecture BHV of test2 is
    signal a1, a2 : std_logic_vector(width-1 downto 0);
begin
    process(clk, rst)
        variable mult_out : unsigned(2*width-1 downto 0);
    begin
        if (rst = '1') then
            a1 <= (others => '0');
a2 <= (others => '0');
        elsif (rising_edge(clk)) then
            a1 <= in3;
a2 <= a1;
mult_out := unsigned(in1) * unsigned(in2);
output <= std_logic_vector(mult_out + unsigned(a2));
        end if;
    end process;
end BHV;
3) (15 points) Identify the violation of the synthesis coding guidelines for combinational logic.

library ieee;
use ieee.std_logic_1164.all;

entity combtest is
    port(
        in1 : in std_logic_vector(1 downto 0);
        in2 : in std_logic;
        out1 : out std_logic_vector(3 downto 0);
        out2 : out std_logic_vector(1 downto 0));
end combtest;

architecture BHV of combtest is
    begin
        process(in1, in2)
            begin
                case in1 is
                    when "00" =>
                        out1 <= "0001";
                        if (in2 = '1') then
                            out2 <= "01";
                            end if;
                    when "01" =>
                        out1 <= "0010";
                        if (in2 = '1') then
                            out2 <= "10";
                            end if;
                    when "10" =>
                        out1 <= "0100";
                        if (in2 = '1') then
                            out2 <= "11";
                            end if;
                    when "11" =>
                        out1 <= "1000";
                        if (in2 = '1') then
                            out2 <= "00";
                            end if;
                    when others => null;
                end case;
            end process;
    end BHV;
4) (6 points) You are creating a 4-input priority encoder, where for each independent input, the entity produces a unique output with the higher inputs receiving higher priority. E.g. when input(3) and input(1) are both asserted, the output corresponds to input(3).

a. How many if-elsif conditions are required to specify this behavior?

5 = \( 4 \text{ input conditions } + \text{ else} \)

b. How many when statements within a case statement are required?

\(2^4 = 16 \) different input combinations
possibly 17 if using when others =

5) (6 points) Describe the compilation error that will occur when synthesizing the following code (there are no syntax errors):

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity test is
port(
in1, in2 : in std_logic;
output : out std_logic_vector(3 downto 0));
end test;

architecture WITH_SELECT of test is
begin
  with in1 select
    output <= "0000" when '0',
              "0010" when others;

  with in2 select
    output <= "1000" when '0',
              "0110" when others;

end WITH_SELECT;

output has multiple drivers
```
6) (20 points) Fill in the provided code to create the illustrated structural architecture using a series of pre-existing A and B components. Use the component declarations for A and B to determine their I/O. Make sure to use the for-generate loop for the four A instances. Make sure to declare any required internal signals.

```vhpf
library ieee;
use ieee.std_logic_1164.all;

entity test3 is
  port (    
    input : in  std_logic_vector(3 downto 0);
    output : out std_logic);
  end test3;

architecture STR of test3 is

  component A
    port (        
      input : in  std_logic;
      output : out std_logic);
  end component;

  component B
    port (        
      input : in  std_logic_vector(3 downto 0);
      output : out std_logic);
  end component;

  signal temp : std_logic_vector (3 downto 0);

begin
  U_LOOP : for i in 0 to 3 generate
    U_A : A port map (        
      input => input(i),
      output => temp(i)        
    );
  end generate U_LOOP;

  U_B : B port map (        
    input => temp,
    output => output        
  );

end STR;
```
7) (10 points) When doing timing simulations, you use a vho file that defines the same entity as the vhd file.
   a. How is the vho file different from the vhd file?
      
      The vho file represents the synthesized circuit

   b. What is the purpose of the sdo file?
      
      to specify propagation delays for synthesized signals

8) a. (6 points) Define the block generate (BG) output of a 4-bit carry-lookahead adder in terms of the propagate signal ($p_i$) and generate signal ($g_i$) of each bit $i$.

   \[ BG = g_3 + p_3 \cdot g_2 + p_2 \cdot p_1 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 \]

   b. (4 points) What practical limitation prevents carry lookahead adders from actually providing a constant propagation delay for different widths?

      fan-in limitations

c. (6 points) True/False. The area requirements of a carry lookahead adder increase linearly with input width.

      false

9) 5 free points for having to take a test at 8:30am.