EEL 4712
Midterm 1 – Spring 2016
VERSION 1

Name: _____

UFID: _____

Sign here to give permission to return your test in class, where other students might see your score:

_____

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:

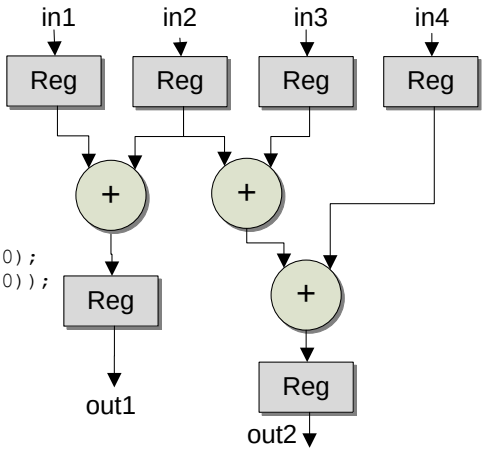
Problem#:	Points
1 (16 points)	
2 (6 points)	
3 (16 points)	
4 (16 points)	
5 (5 points)	
6 (16 points)	
7 (6 points)	
8 (16 points)	
9 (3 points)	3

Total:

Regrade Info:

```
ENTITY _entity_name IS
PORT(__input_name, __input_name : IN STD_LOGIC;
__input_vector_name : IN STD_LOGIC_VECTOR(__high downto __low);
__bidir_name, __bidir_name : INOUT STD_LOGIC;
__output_name, __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) (16 points) Fill in the following VHDL to implement the illustrated circuit. Assume that clk and rst connect to every register. All wires/operations are *width* bits. Ignore adder overflow.




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

entity example is
    generic (
        width : positive := 16);
    port (
        clk,rst            : in  std_logic;
        in1, in2, in3, in4 : in  std_logic_vector(width-1 downto 0);
        out1, out2         : out std_logic_vector(width-1 downto 0));
end example;

architecture BHV of example is









begin
    process(clk, rst)



    begin
        if (rst = '1') then
               -- ASSUME ALL REGISTERS RESET HERE. YOU DON'T NEED TO SPECIFY THE CODE
        elsif (rising_edge(clk)) then

















        end if;
    end process;









end BHV;
```

2) (6 points) Given the following entity, specify the widths of each instantiation in parts a-c.

```
entity test is
  generic (
    width : positive := 32);
  port (
    in1    : in  std_logic_vector(width-1 downto 0);
    in2    : in  std_logic_vector(width-1 downto 0);
    output : out std_logic_vector(width-1 downto 0));
end test;
```

a.

```
U_A : entity work.test
              generic map (width => 16)
              port map (in1 => in1,
                        in2 => in2,
                        output => output);
```

b.

```
U_B : entity work.test
              port map (in1 => in1,
                        in2 => in2,
                        output => output);
```

c. When entity test is used as the top-level entity.

3) (16 points) Fill out the outputs of the waveform for the following two architectures. Assume the left column is the start of the simulation.

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

entity mult is
    generic(
        width : positive := 16);
    port (
        input1, input2 : in  std_logic_vector(width-1 downto 0);
        output         : out std_logic_vector(width-1 downto 0);
        overflow       : out std_logic);
end mult;

architecture BHV1 of mult is

    signal temp : unsigned(2*width-1 downto 0);
begin
    process(input1, input2)
    begin
        temp   <= unsigned(input1) * unsigned(input2);
        output <= std_logic_vector(temp(width-1 downto 0));

        if (temp(2*width-1 downto width) = 0) then
            overflow <= '0';
        else
            overflow <= '1';
        end if;
    end process;
end BHV1;
```

input1	1	2	3	4
input2	10	5	3	5
output				
overflow				

```
architecture BHV2 of mult is
begin
    process(input1, input2)
        variable temp : unsigned(2*width-1 downto 0);
    begin
        temp   := unsigned(input1) * unsigned(input2);
        output <= std_logic_vector(temp(width-1 downto 0));

        if (temp(2*width-1 downto width) = 0) then
            overflow <= '0';
        else
            overflow <= '1';
        end if;
    end process;
end BHV2;
```

input1	1	2	3	4
input2	10	5	3	5
output				
overflow				

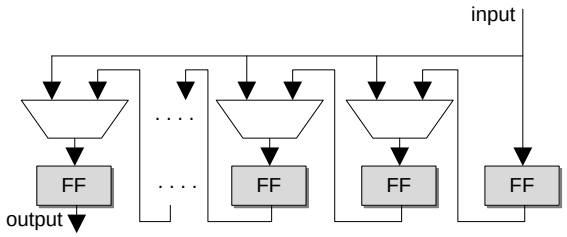
4) (16 points points) **a.** Identify any violations of the *synthesis coding guidelines for combinational logic* and **b.** specify the effect on the synthesized circuit.

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

entity mult is
    generic(
        width : positive := 16);
    port (
        input1, input2 : in  std_logic_vector(width-1 downto 0);
        output         : out std_logic_vector(width-1 downto 0);
        overflow       : out std_logic);
end mult;

architecture BHV4 of mult is
begin
    process(input1, input2)
        variable temp : unsigned(2*width-1 downto 0);
    begin
        temp   := unsigned(input1) * unsigned(input2);
        output <= std_logic_vector(temp(width-1 downto 0));

        if (temp(2*width-1 downto width) /= 0) then
            overflow <= '1';
        end if;
    end process;
end BHV4;
```

5) (5 points) The following code is correct and will synthesize to combinational logic, but what suggestion does it not follow?

```
architecture BHV3 of mult is

    signal temp : unsigned(2*width-1 downto 0) := (others => '0');
begin

    temp   <= unsigned(input1) * unsigned(input2);
    output <= std_logic_vector(temp(width-1 downto 0));

    process(temp)
    begin
        if (temp(2*width-1 downto width) = 0) then
            overflow <= '0';
        else
            overflow <= '1';
        end if;
    end process;
end BHV3;
```

6) (16 points) Fill in the provided code to create the illustrated structural architecture using a series of pre-existing *ff* and *mux2x1* components. Use the component declarations to determine their I/O. Note that there are a total of *width* registers and *width*-1 muxes. The shift input acts as the select for all muxes.




```
entity test is
    generic(width : positive := 8);
    port (
        clk, rst, shift, input : in  std_logic;
        output                 : out std_logic);
end test;

architecture STR of test is
    component ff
        port (
            clk, rst, d : in  std_logic;
            q           : out std_logic);
    end component;

    component mux2x1
        port (
            in1, in2, sel : in  std_logic;
            output : out std_logic);
    end component;



begin









    U_FFS : for i in 1 to width-1 generate
        U_FF : ff port map (
            clk    => clk,
            rst    => rst,
            d  =>

            q  =>

            );

        U_MUX : mux2x1 port map (
            in1    => input,

            in2    =>

            sel    => shift,

            output =>
            );
    end generate U_FFS;









    end STR;
```

7) a. (2 points) What is the propagation delay of operations during a functional simulation?

b. (2 points) What file represents the vhdl for a synthesized circuit used in timing simulations?

c. (2 points) What file specifies propagation delays of signals during timing simulations?

8) a. (9 points) Define the logic for the first 3 carry outs (c_1 to c_3) of a carry lookahead adder (CLA) in terms of the propagate signals (p_i), generate signals (g_i), and carry in (c_0).

b. (2 points) Define the propagate signal (p_i) in terms of adder inputs x_i and y_i.

c. (2 points) Define the generate signal (g_i) in terms of adder inputs x_i and y_i.

d. (3 points) True/false. Adding extra levels of carry lookahead logic trades off propagation delay for reduced area compared to a single-level carry lookahead adder.

9) 3 free points for having to take a test at 8:30am.