EEL 4712
Midterm 1 – Spring 2017
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 (15 points)	
2 (4 points)	
3 (5 points)	
4 (5 points)	
5 (5 points)	
6 (4 points)	
7 (4 points)	
8 (4 points)	
9 (5 points)	
10 (12 points)	
11 (15 points)	
12 (4 points)	
13 (13 points)	
14 (5 points)	**5**

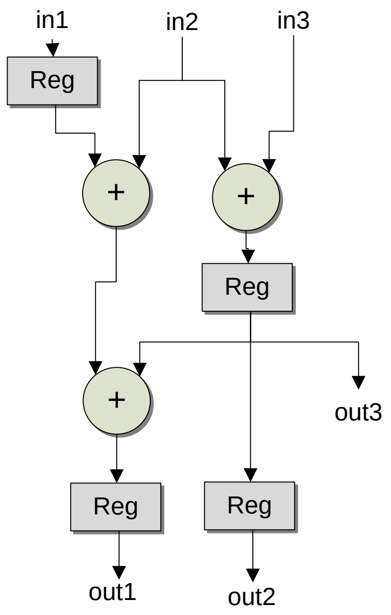
Total:

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
GENERIC MAP(__component_generic => __connect_generic)
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 VHDL to implement the illustrated circuit. Assume that clk and rst connect to every register in the schematic. 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      : in  std_logic_vector(width-1 downto 0);
        out1, out2, out3   : 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) (4 points) When an entity with generics is used as the top-level entity for synthesis, what values does the synthesis tool use for the generics?

3) (5 points) Briefly explain why you should not initialize signals in synthesizable code.

4) (5 points) Complete the following waveform. Pay close attention to the sensitivity list of the process.

```
entity alu is
    generic (
        width : positive := 8);
    port (
        in1, in2 : in  std_logic_vector(width-1 downto 0);
        sel      : in  std_logic;
        output   : out std_logic_vector(width-1 downto 0));
end alu;

architecture BHV of alu is
begin
    process(in1, in2)
    begin
        case sel is
            when '0' =>
                output <= std_logic_vector(unsigned(in1)+unsigned(in2));
            when '1' =>
                output <= std_logic_vector(unsigned(in1)-unsigned(in2));
            when others => null;
        end case;
    end process;
end BHV;
```

input1	5	5	15	15	2
input2	5	5	5	5	4
sel	‘0’	‘1’	‘1’	‘0’	‘0’
output					

5) (5 points) For signals assigned using sequential statements inside a process, when does the signal get updated with the value from the assignment?

6) (4 points) **True/false**. Testbenches should follow the same synthesis coding as other entities.

7) (4 points) **True/false**. Sequential statements inside a process can reassign a signal any number of times.

8) (4 points) **True/false.** Concurrent statements can reassign a signal any number of times.

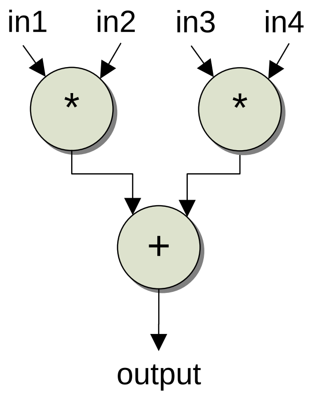
9) (5 points) Assuming you use a variable solely to get an immediately updated value, what will type of hardware resource will be synthesized?

10) (12 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 alu is
    generic (
        width : positive := 8);
    port (
        in1, in2 : in  std_logic_vector(width-1 downto 0);
        sel      : in  std_logic;
        output   : out std_logic_vector(width-1 downto 0);
        neg      : out std_logic);
end alu;

architecture BHV of alu is
begin
    process(sel)
        variable temp : std_logic_vector(width-1 downto 0);
    begin
        case sel is
            when '0' =>
                output <= std_logic_vector(signed(in1)+signed(in2));
            when '1' =>
                temp   := std_logic_vector(signed(in1)-signed(in2));
                neg    <= temp(width-1);
                output <= temp;
            when others => null;
        end case;
    end process;
end BHV;
```

11) (15 points) Fill in the provided code to create the illustrated structural architecture using the specified *add* and *mul* components.

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

entity structure is
    generic (width : positive := 16);
    port (in1, in2, in3, in4 : in  std_logic_vector(width-1 downto 0);
          output             : out std_logic_vector(2*width-1 downto 0));
end structure;

architecture STR of structure is
    component add
        generic (width : positive);
        port (in1, in2 : in  std_logic_vector(width-1 downto 0);
              output   : out std_logic_vector(width-1 downto 0));
    end component;

    component mul
        generic (width : positive);
        port (in1, in2 : in  std_logic_vector(width-1 downto 0);
              output   : out std_logic_vector(2*width-1 downto 0));
    end component;
```



```
begin
```

```
end STR;
```

12) a. (2 points) What information is provided by an sdo file?

b. (2 points) How is a vho file different than a normal vhd file?

13) a. (8 points) Define the logic for the carry out c_4 of a carry look-ahead adder (CLA) in terms of the propagate signals (p_i), generate signals (g_i), and carry in (c_0).

b. (1 point) **True/false**. The delay of a ripple-carry adder increases linearly with width.

c. (1 point) **True/false**. Ignoring fan-in limitations, a CLA has a constant delay for any width.

d. (1 point) **True/false**. Ignoring fan-in limitations, a two-level CLA has a constant delay for any width.

e. (1 point) **True/false**. Ignoring fan-in limitations, a hierachical CLA has a constant delay for any width.

f. (1 point) **True/false**. The delay of an adder than uses a ripple-carry connection between CLA blocks increases linearly with width.

14) 5 free points for having to take a test at 8:30am.