EEL 4712
Midterm 1 – Spring 2019
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:

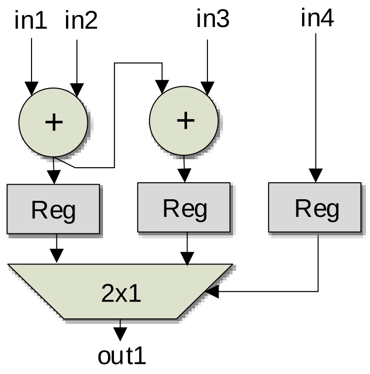
Problem#:	Points
1 (25 points)	
2 (9 points)	
3 (14 points)	
4 (15 points)	
5 (5 points)	
6 (5 points)	
7 (5 points)	
8 (5 points)	
9 (12 points)	
10 (5 points)	5

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) (25 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 except for in4, which is a single bit. Ignore adder overflow. Assume the mux selects the left input when the select = ‘1’. Use the next page if necessary.



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

entity example is
    generic (width : positive := 8);
    port(
        clk, rst      : in  std_logic;
        in1, in2, in3 : in  std_logic_vector(width-1 downto 0);
        in4           : in  std_logic;
        out1, out2    : out std_logic_vector(width-1 downto 0));
end example;

architecture BHV of example is

    signal add1_r, add2_r : std_logic_vector(width-1 downto 0);
    signal add1           : std_logic_vector(width-1 downto 0);
    signal in4_r          : std_logic;

begin
    process(clk, rst)
    begin
        if (rst = '1') then
            add1_r <= (others => '0');
            add2_r <= (others => '0');
            in4_r  <= '0';
        elsif (rising_edge(clk)) then
            add1_r <= add1;
            add2_r <= std_logic_vector(unsigned(add1)+unsigned(in2));
            in4_r  <= in4;
        end if;
    end process;

    add1 <= std_logic_vector(unsigned(in1)+unsigned(in2));

    process(add1_r, add2_r, in4_r)
    begin
        if (in4_r = '1') then
            out1 <= add1_r;
        else
            out1 <= add2_r;
        end if;
    end process;

end BHV;
```

2) (9 points) Given the two following entities *alu* and *alu_top*:

   a. (3 points) Modify the generic map for alu instantiation in *alu_top* to create an alu with a width of 7 bits.

   b. (3 points) Assume that entity *alu* has 2 architectures: FAST and SMALL. Modify the alu instantiation in *alu_top* to explicitly use the FAST architecture.

   c. (3 points) True/False. The initialization of width to 4 in the *alu* entity's generic definition overrides any value specified in the generic map. **FALSE**

```
entity alu is
    generic (
        width : positive := 4);
    port (
        in1    : in  std_logic_vector(width-1 downto 0);
        in2    : in  std_logic_vector(width-1 downto 0);
        sel    : in  std_logic_vector(1 downto 0);
        output : out std_logic_vector(width-1 downto 0));
end alu;
```

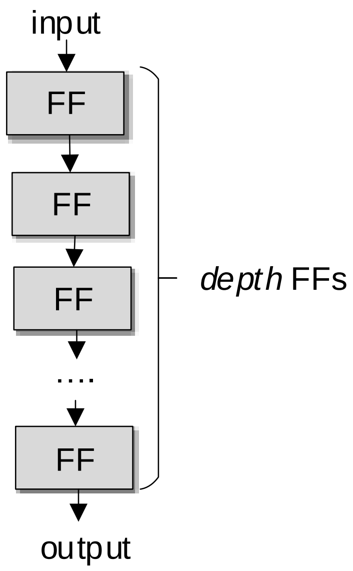
```
library ieee;
use ieee.std_logic_1164.all;

entity alu_top is
    port (
        in1    : in  std_logic_vector(6 downto 0);
        in2    : in  std_logic_vector(6 downto 0);
        sel    : in  std_logic_vector(1 downto 0);
        output : out std_logic_vector(6 downto 0));
end alu_top;

architecture STR of alu_top is
begin
    U_ALU : entity work.alu(FAST)
        generic map (

             width => 7

                    )
        port map (
            in1    => in1,
            in2    => in2,
            sel    => sel,
            output => output);
end STR;
```

3) a. (10 points) Identify the violation of the synthesis coding guidelines for *combinational* logic in the following code, and state the effect on the synthesized circuit. Note: there are no syntax, casting, or width-mismatch errors.

b. (4 points) Fix the violation with a single line of code.

```
process(en, state)
begin

   next_state <= state;


  case state is
    when STATE_0 =>

      output <= "0001";

      if (en = '1') then
        next_state <= STATE_1;      -- next state not defined on all paths, causes
      end if;                       -- latches during synthesus

    when STATE_1 =>

      output <= "0010";

      if (en = '1') then
        next_state <= STATE_2;
      end if;

    when STATE_2 =>

      output <= "0100";

      if (en = '1') then
        next_state <= STATE_3;
      end if;

    when STATE_3 =>

      output <= "1000";

      if (en = '1') then
        next_state <= STATE_0;
      end if;

    when others => null;
  end case;
end process;
```

4) (15 points) Fill in the provided code to create the illustrated circuit as a structural architecture using the specified FF component. Connect each FF to the clock and reset (not shown in figure). Use the next page if necessary.



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

entity structure is
    generic (depth : positive := 16);
    port (clk    : in  std_logic;
          rst    : in  std_logic;
          input  : in  std_logic;
          output : out std_logic);
end structure;

architecture STR of structure is

    component ff
        port(clk, rst, D : in  std_logic;
             Q           : out std_logic);
    end component;

    signal ff_s : std_logic_vector(depth downto 0);

begin
    U_FFS : for i in 0 to depth-1 generate
        U_FF : ff port map (
            clk => clk,
            rst => rst,
            d   => ff_s(i),
            q   => ff_s(i+1));
    end generate;

    ff_s(0) <= input;
    output <= ff_s(depth);

end STR;
```

5) (5 points) Create a for loop that defines the generate (g) and propagate (p) bits for a CLA in terms of inputs x and y. Assume that g, p, x, and y are all std_logic_vector signals of *width* bits. *Note: you do not need the carry logic, just the generate and propagate.*

```
process(x, y)
begin

    for i in 0 to width-1 loop
            g(i) <= x(i) and y(i);
            p(i) <= x(i) or y(i);
    end loop;

 end process;
```

6) (5 points) When is it safe to use a for loop in synthesizable code (choose the best answer):
   a. When feedback from one register to another is required in the circuit
   b. When you need to chain multiple operations into a single cycle
   c. When the unrolled version of the loop corresponds to the intended circuit
   d. When the number of iterations of the loop is known at compile time

7) (5 points) Explain why the following code will have errors during compilation/synthesis (assume the numeric_std package is used):

```
signal x, y, z : std_logic_vector(7 downto 0);
…
x <= unsigned(y) + unsigned(z);
```

Unsigned can't be assigned to std_logic_vector. It must be explicitly cast because VHDL is strongly typed.

8) (5 points) Explain why the following code will have errors during compilation/synthesis (assume the numeric_std package is used):

```
signal in1, in2 : unsigned(7 downto 0);
signal sum : unsigned(8 downto 0);
…

sum <= in1 + in2;
```

Signals of different widths can't be assigned to each other.

9) (12 points)

a. (2 points) What type of relationship exists between area/resources and width for a ripple-carry adder?

   linear

b. (2 points) Why in practice does a carry-lookahead adder not actually achieve a constant propagation delay?

   Fan-in limitations on gates

c. (2 points) True/false. A 2-level carry-lookahead adder uses fewer resources as width increases than a single level, without an increase in delay.

   False

d. (2 points) For the hierarchical carry-lookahead adder, how much does the width have to increase for the propagation delay to increase?

   The depth/delay increases every time the width doubles.

e. (2 points) **True/false**. The delay of a ripple-carry adder using blocks of carry-lookahead adders, instead of full adders, achieves a constant propagation delay when ignoring fan-in limitations.

   False

f. (2 points) Define the logic for the carry out c4 of a carry look-ahead adder (CLA) in terms of the propagate signals (p_i), generate signals (g_i), and carry in (c_0).

c4 = g3 + p3g2 + p3p2g1 + p3p2p1g0 + p3p2p1p0c0

10) 5 free points for having to take a test at 8:30am.