EEL 4712
Midterm 1 – Spring 2018
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:

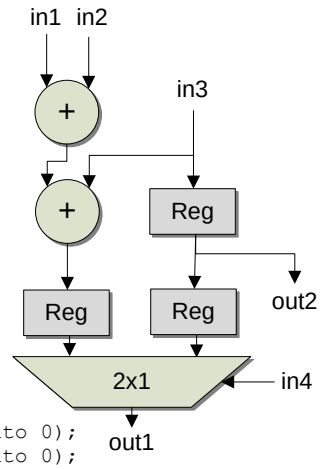
Problem#:	Points
1 (25 points)	
2 (6 points)	
3 (6 points)	
4 (25 points)	
5 (15 points)	
6 (8 points)	
7 (10 points)	
8 (5 points)	5

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 in4 = '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 reg_add2_out, reg_in3, reg_in3_2 : std_logic_vector(width-1 downto 0);
    signal add1_out, add2_out               : std_logic_vector(width-1 downto 0);

begin
    process(clk, rst)
    begin
        if (rst = '1') then
            reg_add2_out <= (others => '0');
            reg_in3      <= (others => '0');
            reg_in3_2    <= (others => '0');
        elsif (rising_edge(clk)) then
            reg_add2_out <= add2_out;
            reg_in3      <= in3;
            reg_in3_2    <= reg_in3;
        end if;
    end process;

    out2     <= reg_in3;
    add1_out <= std_logic_vector(unsigned(in1)+unsigned(in2));
    add2_out <= std_logic_vector(unsigned(add1_out)+unsigned(in3));

    process(reg_add2_out, reg_in3_2, in4)
    begin
        if (in4 = '1') then
            out1 <= reg_add2_out;
        else
            out1 <= reg_in3_2;
        end if;
    end process;

end BHV;
```

2) (6 points) Fill in the VHDL to implement a simple testbench for the specified add component. The testbench should test 5+10 and 15+20, waiting 10 ns in between tests. The testbench does *not* need to verify the correct output.

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

entity add_tb is
end add_tb;

architecture TB of add_tb is

    component add
        port (input1, input2 : in  std_logic_vector(7 downto 0);
              output         : out std_logic_vector(7 downto 0);
              carry          : out std_logic);
    end component;

    signal input1, input2, output : std_logic_vector(7 downto 0) := (others => '0');
    signal carry                  : std_logic;

begin  -- TB

    UUT : entity work.add
        port map (
            input1 => input1,
            input2 => input2,
            output => output,
            carry  => carry);

    process
    begin
        input1 <= std_logic_vector(to_unsigned(5, 8));
        input2 <= std_logic_vector(to_unsigned(10, 8));
        wait for 10 ns;

        input1 <= std_logic_vector(to_unsigned(15, 8));
        input2 <= std_logic_vector(to_unsigned(20, 8));
        wait for 10 ns;

        wait;
    end process;

end TB;
```

3) (6 points) Given the two following entities *alu* and *alu_top*:
   a. (3 points) What is the width of the alu instance when *alu_top* is the top-level entity?

7

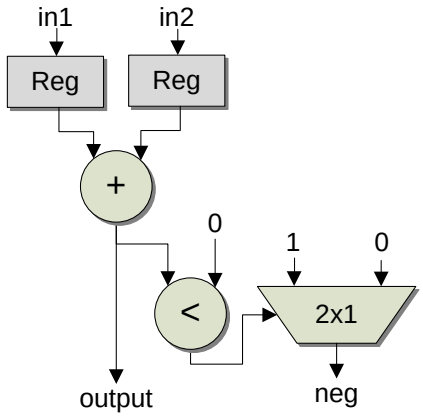
   b. (3 points) What is the width of the alu instance when *alu* is the top-level entity?

4

```
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;
```

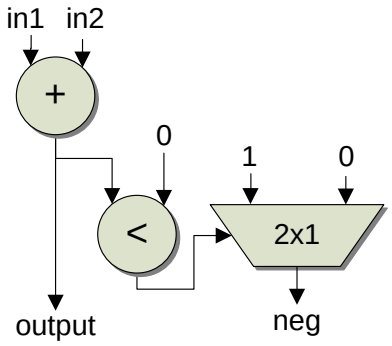
```
library ieee;
use ieee.std_logic_1164.all;

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
        generic map (width => 7)
        port map (
            in1    => in1,
            in2    => in2,
            sel    => sel,
            output => output);
end STR;
```

4) (25 points) For the following code that is intended to implement the illustrated circuit, point out every mistake and every violation of synthesis coding guidelines. For violations that have an effect that was explained in class, specify that effect. All signals are width bits except for the mux and less-than output. Note: there are no syntax, casting, or width-mismatch errors.



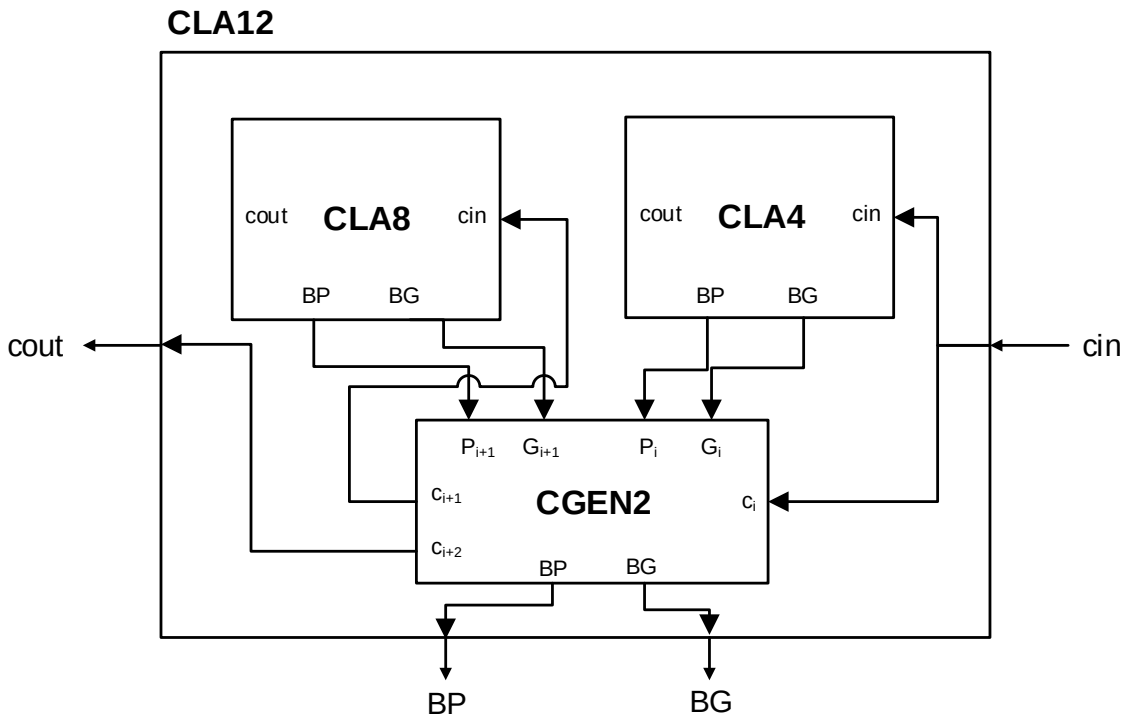

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity bad_vhdl is
    generic (width : positive := 8);
    port (clk    : in  std_logic;
          rst    : in  std_logic;
          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);
          neg    : out std_logic);
end bad_vhdl;

architecture BVH of bad_vhdl is

    -- PROBLEM 1: SHOULD NOT INITIALIZE SIGNALS
    signal reg_in1 : std_logic_vector(width-1 downto 0) := (others => '0');
    signal reg_in2 : std_logic_vector(width-1 downto 0) := (others => '0');
    signal temp    : std_logic_vector(width-1 downto 0) := (others => '0');


begin

    process(clk)  -- PROBLEM 2: CLOCK AND RESET IN SENSITIVITY LIST
    begin
        if (rst = '1') then
            reg_in1 <= (others => '0');
            reg_in2 <= (others => '0');
        elsif (clk = '1') then -- PROBLEM 3: SHOULD CHECK FOR RISING EDGE
            reg_in1 <= in1;
            reg_in2 <= in2;
        end if;
    end process;


    -- PROBLEM 4: SENSIVITY LIST SHOULD HAVE REG_IN1 AND REG_IN2
    process(in1, in2)
    begin
        -- PROBLEM 5: TEMP MUST BE A VARIABLE FOR THIS TO WORK
        temp   <= std_logic_vector(signed(reg_in1)+signed(reg_in2));
        output <= temp;

        if (signed(temp) < 0) then
            neg <= '1';
        end if;
        -- PROBLEM 6: NEG NOT SPECIFIED ON ALL PATHS, SYNTHESIS INFERS A LATCH
    end process;


end BVH;
```

5) (15 points) Fill in the provided code to create the illustrated structural architecture using the specified components. Use the next page if necessary.




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

entity structure is
    generic (width : positive := 16);
    port (in1, in2 : in  std_logic_vector(width-1 downto 0);
          output   : out std_logic_vector(width-1 downto 0);
          neg      : out std_logic);
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 less_than
        generic (width : positive);
        port (in1, in2 : in  std_logic_vector(width-1 downto 0);
              output   : out std_logic);
    end component;

    component mux_2x1
        port (in1, in2 : in  std_logic;
              sel      : in  std_logic;
              output   : out std_logic);
    end component;

    constant C0_WIDTH : std_logic_vector(width-1 downto 0) := (others => '0');
    constant C0 : std_logic := '0';
    constant C1 : std_logic := '1';

    signal add_out : std_logic_vector(width-1 downto 0);
    signal lt_out  : std_logic;
begin
    U_ADD : entity work.add
        generic map (width => width)
        port map (in1    => in1,
                  in2    => in2,
                  output => add_out);

    output <= add_out;

    U_LT : entity work.less_than
        generic map (width => width)
        port map (in1    => add_out,
                  in2    => C0_WIDTH,
                  output => lt_out);

    U_MUX : entity work.mux_2x1
        port map (in1    => C1,
                  in2    => C0,
                  sel    => lt_out,
                  output => neg);

end STR;
```

6) (8 points) Complete the following schematic to implement a 12-bit hierarchical CLA using an 8-bit CLA and a 4-bit CLA. You do not need to show the add inputs and sum output, just connect the carry logic that is shown.



7) (10 points)

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

linear

b. (2 points) What type of relationship exists between delay and width for a carry-lookahead adder, ignoring practical fan-in limitations?

constant

c. (2 points) What type of relationship exists between delay and width for a two-level carry-lookahead adder, ignoring practical fan-in limitations?

constant

d. (2 points) What type of relationship exists between delay and width for a hierarchical carry-lookahead adder, ignoring practical fan-in limitations?

logarithmic

e. (2 points) **True/false**. The area of a carry-lookahead adder increases linearly with width.

false

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