EEL 4712
Midterm 1 – Spring 2020
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:

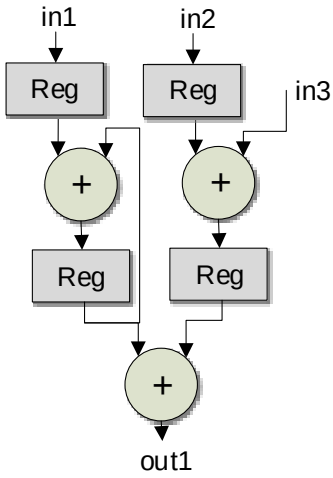
Problem#:	Points
1 (25 points)	
2 (12 points)	
3 (15 points)	
4 (18 points)	
5 (12 points)	
6 (5 points)	
7 (4 points)	
8 (4 points)	
9 (5 points)	5

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. Ignore adder overflow.




```
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);
        out1          : out std_logic_vector(width-1 downto 0));
end example;

architecture BHV of example is









begin



    process(clk, rst)
    begin
        if (rst = '1') then









        elsif (rising_edge(clk)) then









        end if;
    end process;









end BHV;
```

2) (12 points) Fill in the VHDL to implement a simple testbench for the specified mux component. **The testbench should instantiate a mux using an architecture IF_STATEMENT.** The testbench should test 3 separate input combinations, waiting 10 ns in between tests. The testbench does *not* need to verify the correct output. Declare all signals as std_logic.

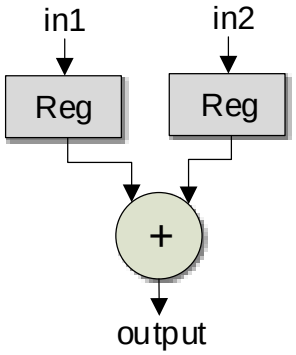
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mux_tb is
end mux_tb;

architecture TB of mux_tb is




begin  -- TB

    UUT : entity work.mux(                    )
        port map (
            in1 => in1,
            in2 => in2,
            sel => sel,
            output => output);

    process
    begin









        wait;
    end process;
end TB;
```

3) a. (12 points) Identify the violation of the synthesis coding guidelines for *combinational* logic in the following priority encoder code, and state the effect on the synthesized circuit. Note: there are no syntax, casting, or width-mismatch errors.

b. (3 points) Fix the violation with a single line of code.

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

entity pe is
	port (
		input : in std_logic_vector(3 downto 0);
		output : out std_logic_vector(1 downto 0);
		valid : out std_logic
		);
end pe;

architecture default of pe is
begin



	process(input)
	begin



		if (input(3) = '1') then
			output <= "11";


		elsif (input(2) = '1') then
			output <= "10";


		elsif (input(1) = '1') then
			output <= "01";


		elsif (input(0) = '1') then
			output <= "00";


		else
			output <= "00";
			valid <= '0';

		end if;



	end process;



end default;
```

4) (18 points) Fill in the provided code to create the illustrated circuit as a structural architecture using the specified reg and add components. Connect each reg to the clock and reset (not shown in figure). All reg and add instances should use the *width* of the *structure* entity. Use the next page if necessary.




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

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


architecture STR of structure is

    component reg
        generic (width : positive := 8);
        port (clk, rst : in  std_logic;
              input    : in  std_logic_vector(width-1 downto 0);
              output   : out std_logic_vector(width-1 downto 0));
    end component;

    component add
        generic (width : positive := 8);
        port (in1, in2 : in  std_logic_vector(width-1 downto 0);
              output   : out std_logic_vector(width-1 downto 0));
    end component;




begin
```

```
end STR;
```

5) (12 points)
   a. (3 points) Resources grow _____ with width for a carry lookahead adder.

   b. (3 points) **True/false**. Due to fan-in limitations, carry lookahead adders tend to have a latency that increases quadratically with width.

   c. (3 points) **True/false**. A ripple-carry adder using blocks of carry-lookahead adders instead of full adders has a delay that increases logarithmically with width.

   d. (3 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).

6) (5 points) For the following process, what will the values of x and y be at the <u>end of the process</u> when x= 20, y = 30, and the process is triggered by in1 becoming 50? Explain your answer for partial credit.

```
signal x, y, in1 : unsigned(7 downto 0);
…
process(in1)
begin
    x <= in1 + 10;
    y <= x + 10;
end process;
```

7) (4 points) **True/false**. A single assignment to all outputs at the beginning of a process for combinational logic will guarantee there are no latches for that logic.

8) (4 points) **True/false**. Multiple concurrent assignments to a signal are not synthesizable, but it is valid to assign the same signal sequentially inside a process and concurrently outside a process.

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