EEL 4712
Midterm 1 – Spring 2013
VERSION 1

Name: ____________________

UFID: ____________________

Sign your name here if you would like for your test to be returned in class:

____

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:

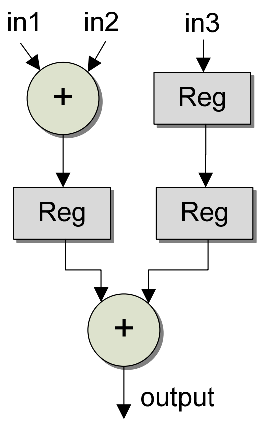
Problem#:	Points
1 (15 points)	
2 (12 points)	
3 (6 points)	
4 (6 points)	
5 (6 points)	
6 (6 points)	
7 (6 points)	
8 (12 points)	
9 (12 points)	
10 (15 points)	
11 (4 points)	4

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) (15 points) Fill in the following behavioral VHDL to implement the illustrated circuit. Assume that clk and rst connect to every register. All wires and operations are *width* bits. Ignore overflow from the adders.



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

entity test1 is
  generic (
    width         :     positive := 16);
  port (
    clk, rst      : in  std_logic;
    in1, in2, in3 : in  std_logic_vector(width-1 downto 0);
    output        : out std_logic_vector(width-1 downto 0));
end test1;

architecture BHV of test1 is






begin

  process(clk, rst)


  begin
    if (rst = '1') then








    elsif (rising_edge(clk)) then










    end if;
  end process;






end BHV;
```

2) (12 points) Identify the violations (if any) of the *synthesis coding guidelines for sequential logic*.

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

entity count is
  generic (
    width :     positive := 16);
  port (
    clk   : in  std_logic;
    rst   : in  std_logic;
    en    : in  std_logic;
    input : in  std_logic_vector(width-1 downto 0);
    zero  : out std_logic);
end count;

architecture BHV of count is

  signal count : signed(width-1 downto 0);

begin

  process(clk, rst)
  begin

    if (count > 0) then
      count <= count - 1;
    end if;

    if (rst = '1') then
      count <= (others => '0');

    elsif (clk = '1' and clk'event) then

      if (en = '1') then
        count <= signed(input);
      end if;

      if (count = 0) then
        zero <= '1';
      else
        zero <= '0';
      end if;

    end if;
  end process;
end BHV;
```

3) (6 points)When using sequential statements, when does a signal’s value get updated?

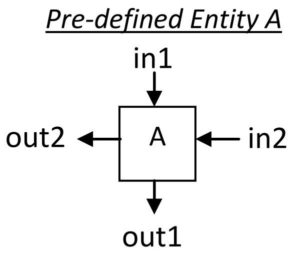
4) (6 points)When using sequential statements, when does a variable’s value get updated?

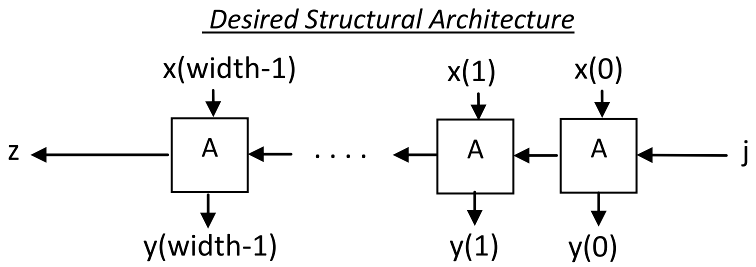
5) (6 points) For a behavioral architecture of a priority encoder, briefly explain if a *case* statement or an *if* statement is a more appropriate construct.

6) (6 points) While implementing a behavioral architecture for an ALU, you get a warning during synthesis about a latch being inferred for output “overflow.” What does this suggest is wrong with your VHDL code?

7) (6 points) When defining a state machine with 100 states, you notice that a particular output *x* has a value of ‘0’ in 95 of the states. What does this suggest that you should do to simplify your code?

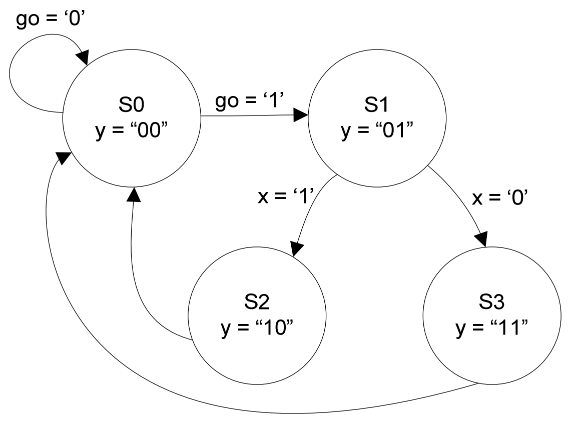
8) (12 points) Fill in the provided code to create the illustrated structural architecture using a series of *A* components. Assume you are given the shown *A* entity, which has already been defined elsewhere. The length of the series is specified by the generic *width*.





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

entity test2 is
  generic (
    width          :     positive := 8);
  port (
    x : in  std_logic_vector(width-1 downto 0);
    j : in std_logic;
    y : out std_logic_vector(width-1 downto 0);
    z : out std_logic);
end test2;

architecture STR of test2 is

  component A
    port (
      in1, in2    : in  std_logic;
      out1, out2  : out std_logic);
  end component;




begin
  U_LOOP : for i in 0 to width-1 generate

    U_A : entity work.A
      port map (

          in1 =>

          in2 =>

          out1 =>

          out2 =>

       );

  end generate U_LOOP;




end STR;
```

9) (12 points) Fill in the code to implement the following Moore finite state machine, *using the 2-process FSM model*. **Assume that if an edge does not have a corresponding condition, that edge is always taken on a rising clock edge.** Assume that S0 is the start state. Use the next page if extra room is needed.

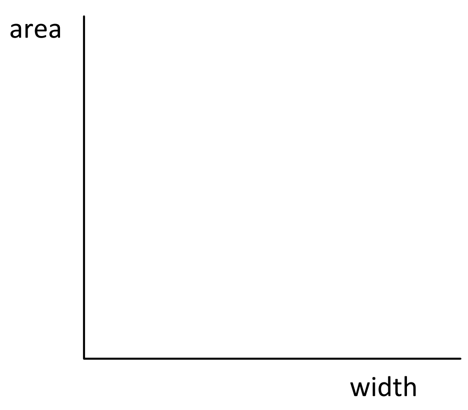


```
library ieee;
use ieee.std_logic_1164.all;

entity fsm is
  port (
    clk, rst, go, x : in  std_logic;
    y               : out std_logic_vector(1 downto 0));
end fsm;

architecture PROC2 of fsm is

  type STATE_TYPE is (S0, S1, S2, S3);
  signal state, next_state : STATE_TYPE;

begin

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




    elsif (clk'event and clk = '1') then




    end if;
  end process;

  process(                                    )
  begin
  end process;
end PROC2;
```

10) a. (3 points) Briefly describe the tradeoffs of ripple-carry and carry-lookahead adders.

b. (3 points) Define the block propagate (BP) output of a 4-bit carry-lookahead adder in terms of the propagate signal (p_i) of each bit *i.*

c. (8 points) On the graphs below, draw the area and delay as a function of width for both ripple-carry and carry-lookahead adders. Assume the carry-lookahead adder has the ideal delay. Make sure to label each curve.



11) 4 free points for having to take a test at 8:30am.