EEL 4712
Midterm 2 – Spring 2014
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:

<table>
<thead>
<tr>
<th>Problem#</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (10 points)</td>
<td></td>
</tr>
<tr>
<td>2 (6 points)</td>
<td></td>
</tr>
<tr>
<td>3a (6 points)</td>
<td></td>
</tr>
<tr>
<td>3b (6 points)</td>
<td></td>
</tr>
<tr>
<td>4 (18 points)</td>
<td></td>
</tr>
<tr>
<td>5a (20 points)</td>
<td></td>
</tr>
<tr>
<td>5b (15 points)</td>
<td></td>
</tr>
<tr>
<td>5c (15 points)</td>
<td></td>
</tr>
<tr>
<td>6 (4 points)</td>
<td>4</td>
</tr>
</tbody>
</table>

Total: ____________________________

Regrade Info:
ENTITY entity_name IS
PORT(input_name, input_name : IN STD_LOGIC;
     --input_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;
END CASE;

<generate_label>: FOR <loop_id> IN <range> GENERATE
-- Concurrent Statement(s)
END GENERATE;

type identifier is type_definition;

subtype identifier is subtype_indication;
1) (10 points) Assume you are given an FPGA that consists of the following CLB structures:

Map the following circuit onto these CLBs by drawing rectangles to represent CLBs. Use the minimum number of CLBs.
2) (6 points) Connection boxes and switch boxes are used for which of the following (only choose one):
   (a) Configuring the internals of a DSP unit to perform different operations
   (b) Connecting LUTs within CLBs (e.g., carry chains)
   (c) Providing reconfigurable interconnect between CLBs, DSPs, block RAMs, and I/O
   (d) Implementing addressing logic for distributed RAM components
   (e) Distributing clock signals without timing problems

3) a. (6 points) For a 25 MHz clock, what is the range of a counter (e.g., the number of cycles to count) that determines when 100 ms have elapsed?

\[
\text{Count} = 25 \times 10^6 \frac{\text{cycles}}{s} \times 1 \text{s} = 25 \times 10^5
\]

b. (6 points) You are designing a circuit with a 50 MHz input clock that generates an output clock pulse when an asynchronous control input (e.g., a button) has been asserted for 1 second. However, it is not possible to guarantee that the input has been asserted for exactly 1 second. Briefly explain why and show the minimum range of absolute error.

\[
\text{Error} = \pm 1 \text{ cycle} = \frac{1}{50 \times 10^6 \text{ s}}
\]
4) (18 points) Fill in the code to implement the following Moore finite state machine (FSM), 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.

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

entity fsm is
  port ( 
    clk, rst, 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
      state <= S0;
    elsif (clk'event and clk = '1') then
      state <= next_state;
    end if;
  end process;

  process(
      y, state
  begin
    next_state <= state;
    case state is
      when S0 => 
        y <= "00";
        if (x = '1') then
          next_state <= S1;
        end if;
      when S1 =>
        y <= "01";
      when S2 => 
        y <= "10";
      when S3 =>
        y <= "11";
      when others => null;
    end case;
  end process;

end PROC2;
```
5) a. (20 points) Create an FSM that implements the following pseudo-code. Do not write VHDL and instead leave the FSM in graphical form (i.e., state machine with corresponding operations in each state). Make sure to specify all operations and state transitions. Note that output, go, num, den, and done are I/O.

```c
int i;
// reset values for outputs
done = 0; output = 0;

while (1) {
    while (go == 0);
    done = 0;
    i = 0;
    // Store inputs in a register
    num_reg = num;
    den_reg = den;

    while (num_reg > 0) {
        num_reg = num_reg - den_reg;
        i ++;
    }

    output = i;
    done = 1;
    while (go == 1);
}```
b. (15 points) For the same pseudo-code, create a datapath capable of executing the code (ignore the controller in this step). Make sure to show all control signals (i.e., mux select signals, register load signals, comparator output signals). Make sure to include a register for `num`, `den`, `output`, and `done` in addition to other registers you might need. To make things easier, I don’t recommend sharing resources. **Do not write any code, just show the datapath.** If you do any non-obvious optimization, make sure to explain.
c. (15 points) For the datapath in the previous step, draw an FSM capable of controlling the datapath to perform the pseudo-code. In each state of the FSM, show the values of your control signals from the previous step that configure the datapath to do the corresponding operations. Assume that \( go \) is an input to the controller. Hint: to save yourself time, try to use the same states as the FSMD, and just change the operations to the corresponding control signals. Use default values for control signals to save time. Do not write any VHDL code, just show the FSM and control signals. Be sure to mention default signal values to save space.

\[
\begin{align*}
default, & \\
all & select = 0 \\
all & id = 0
\end{align*}
\]

6) 4 free points for a long test.