EEL 4712
Midterm 2 – Spring 2011
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 (5 points)	
2 (8 points)	
3 (10 points)	
4 (10 points)	
5 (5 points)	
6 (10 points)	
7a (13 points)	
7b (13 points)	
7c (13 points)	
7d (13 points)	

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 __identifier is type_definition;

subtype  __identifier is subtype_indication;
```

1) (5 points) Block RAMs on FPGA generally only support synchronous reads. Explain why the following code will not be inferred as block RAM, and also mention how to change the code so that block RAM is inferred during synthesis.

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

entity ram is
  port (clk        : in  std_logic;
        we         : in  std_logic;
        rd_addr    : in  std_logic_vector(4 downto 0);
        wr_addr    : in  std_logic_vector(4 downto 0);
        data_in    : in  std_logic_vector(3 downto 0);
        data_out   : out std_logic_vector(3 downto 0));
end ram;

architecture syn of ram is
  type ram_type is array (31 downto 0) of std_logic_vector (3 downto 0);
  signal RAM : ram_type;
begin
  process (clk)
  begin
    if (clk'event and clk = '1') then
      if (we = '1') then
        RAM(to_integer(unsigned(wr_addr))) <= data_in;
      end if;
    end if;
  end process;

  data_out <= RAM(to_integer(unsigned(rd_addr)));

end syn;
```

2) a. (2 points) Write a VHDL type declaration called MY_ARRAY that creates a 2D array with 5 rows and 10 columns, where each element is a 16-bit std_logic_vector.

b. (2 points) Write a VHDL type declaration called MY_ARRAY that creates a 2D array with unconstrained ranges for each dimension, where each element is a 16-bit std_logic_vector.

c. (2 points) Using the type from part b, instantiate an object of type MY_ARRAY with 10 rows and 20 columns.

d. (2 points) Which of the following, if any, are legal VHDL array declarations (pre-2008)?

```
type MY_ARRAY is array (natural range<>, natural range<>) of std_logic_vector

type MY_ARRAY is array (natural range<>, 0 to 50) of std_logic_vector

type MY_ARRAY is array (0 to 100, 0 to 50) of std_logic_vector
```

3) a. (5 points) For the VGA lab, you were required to display the image in 5 different locations. Given an image that is 128x128 and a screen resolution of 640x480, define the constants that specify the pixel boundaries when displaying the image centered vertically, but horizontally aligned with the left side of the screen. Show your work.

constant X_START : integer :=

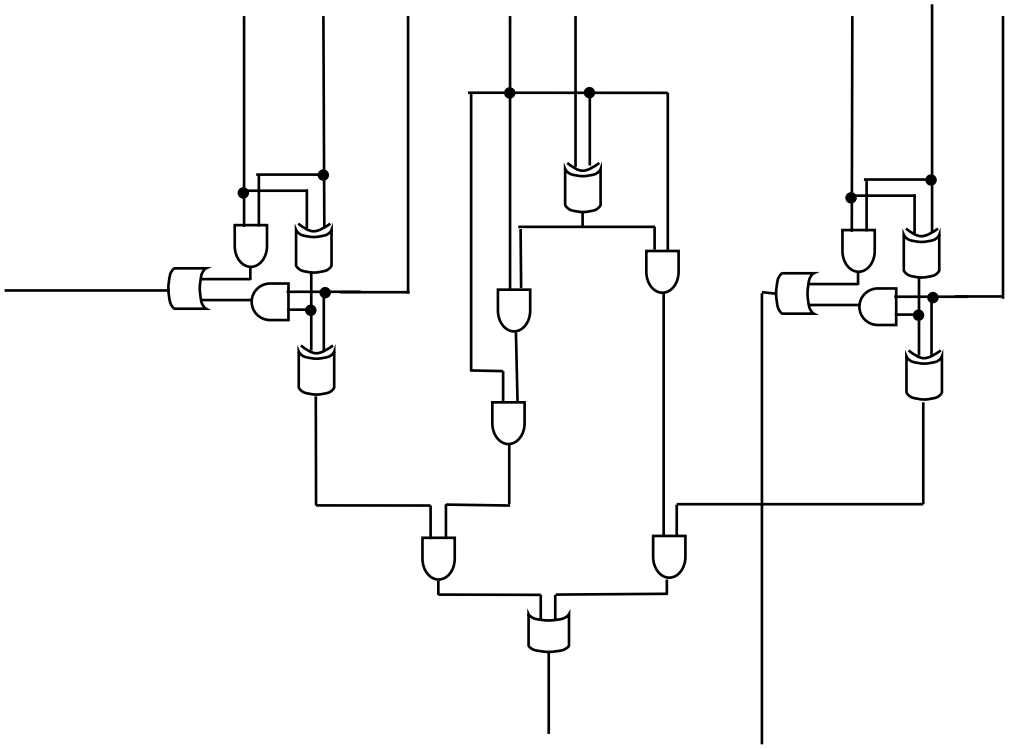
constant X_END : integer :=

constant Y_START : integer :=

constant Y_END : integer :=

b. (5 points) In class, I explained that the VGA control signals needed to be delayed to align with the output of the ROM. Explain why these signals would not be aligned without the delay registers.

4) (10 points) Map the following circuit onto 4-input, 2-output LUTs by drawing shapes around each portion of the circuit that is mapped to an individual LUT.



5) (5 points) What is the maximum number of gates that can be implemented in a 4-input, 2-output LUT?

a. 2^4 gates
b. 4^2 gates
c. 2^4*2 gates
d. 4^2*2 gates
e. other

6) (10 points) ***Briefly*** describe the purpose of each of the following components:

Switch box:

Connection box:

Routing tracks:

7) a. (13 points) Create an FSMD that implements the following pseudo-code. Do not write VHDL and instead leave the FSMD in graphical form (i.e., state machine with corresponding operations in each state). Make sure to specify all operations and state transitions. Note that "input" and "output" are I/O. *Assume there is also a go signal that starts the FSMD (i.e. the circuit waits at the beginning until go is asserted) and a done signal that you should assert when the output is valid.*

```
n = input;
i = 3;
x = 1;
y = 1;
while (i <= n) {
  temp = x+y;
  x = y;
  y = temp;
  i ++;
}
output = y;
```

b. (13 points) Convert the previous FSMD into VHDL using the 1-process FSMD model:

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

entity FSMD is
  port (
    clk    : in  std_logic;
    rst    : in  std_logic;
    go     : in  std_logic;
    done   : out std_logic;
    input  : in  std_logic_vector(7 downto 0);
    output : out std_logic_vector(7 downto 0));
end FSMD;

architecture ONE_PROCESS of FSMD is









begin

  process(clk, rst)



  begin
```

```
  end process;
end ONE_PROCESS;
```

c. (13 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). Hint: to make things easier, do not try to share resources (e.g., x+y and i++ as a single adder).

d. (13 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 FSM to do the corresponding operations. Hint: to save yourself time, try to use the same states as the FSMD, and just change the operations to the corresponding control signals.