EEL 4712 Midterm 2 – Spring 2018 VERSION 1 Name: _____

UFID:_____

Sign here to give permission for your test to be returned in class, where others 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 (12 points)	
2 (8 points)	
3 (16 points)	
4 (20 points)	
5 (20 points)	
6 (20 points)	
Free	4

Total:

Regrade Info:

 a. (4 points) In the VGA lab, the ROM stored blocks that were displayed as 2x2 pixels to the screen. For blocks consisting of 16x16 pixels, show the row and column addressing logic as a function of the h_count and v_count for an image that is displayed at the top-left of the screen.

b. (4 points) Briefly describe the purpose of the horizontal sync signal in the VGA lab.

c. (4 points) What is the purpose of a MIF file?

2. (8 points) You are designing a circuit that controls a display with a signal similar to h_sync in the VGA lab. This signal must be held low for $1.1 \,\mu s$ (1100 ns) and then high for $5.2 \,\mu s$ (5200 ns), which then repeats indefinitely. For a 50 MHz clock, show the corresponding timer values for the rising edge and falling edge of the signal. Assume that a timer value of 0 corresponds to the beginning of the low section.

3. a. (4 points) What is the relationship between number of inputs and the number of SRAM bits in a LUT? Assume a single output.

b. (4 points) Other than the reconfigurable interconnect, *name* three resources provided by existing FPGAs.

c. (4 points) The carry chain between LUTs in a CLB is frequently used for what type of resource?

d. (4 points) Why do CLBs provide a mux to select between a FF and a LUT output?

(20 points) Fill in the code to implement the following Moore finite state machine (FSM) using the 2-process FSM model. Assume that INIT is the initial state. Transitions without conditions are always taken. Use the next page if extra room is needed.



process(begin

end process; end PROC2; (20 points) Create an FSMD that implements the following pseudo-code. <u>Do not write VHDL and</u> <u>instead leave the FSMD in graphical form</u> (i.e., state machine with corresponding operations in each state).

```
Inputs: go (std_logic), N (std_logic_vector)
Outputs: result (std logic vector), done (std logic)
// reset values for outputs
done = 0; result = 0;
while (1) {
       while (go == 0);
       done = 0;
       x = 0;
       i = 1;
       nReg = N; // store input N into a register
       while (i <= nReg) {
          x = x + (2*i) - 1;

i = i + 1;
        }
       result = x;
       done = 1;
       while (go == 1);
}
```

6. (20 points) Draw an FSM capable of controlling the illustrated datapath to perform the pseudo-code in question 5, by assigning or reading from the underlined control signals. The controller should have an additional input for *go* (not shown in the datapath). Note that the controller specifies the done status through *ctrl_done*, but that the datapath connects this signal to the actual done output. Assume that left mux inputs have a select value of 1. *Do not write any VHDL code*, just show the FSM and control signals. Be sure to mention default signal values to save space. NOTE: this FSM might have different states than your FSMD in problem 5.



Problem 5+6 Reference

```
Inputs: go (std_logic), N (std_logic_vector)
Outputs: result (std_logic_vector), done (std_logic)
// reset values for outputs
done = 0; result = 0;
while (1) {
    while (go == 0);
    done = 0;
        x = 0;
        i = 1;
        nReg = N; // store input N into a register
    while (i <= nReg) {
            x = x + (2*i) - 1;
            i = i + 1;
        }
    result = x;
        done = 1;
    while (go == 1);
}</pre>
```

```
0
                                                                        Ν
                         4
<u>x sel</u> -
                     2x1
                                             2x1
                                                             -<u>i sel</u>
                                 <u>i en</u>
                       ¥
                                                ¥
                                                                                                                    ctrl done
                                                                     nReg
                                                                                ∢–<u>n en</u>
   <u>x en</u>
                                                i
                       Х
                                                                                                 <u>done en</u>-
                                                                                                                       done
                                          Shift Left
                                                                                                                          *
                                                                      <=
                                                                                                                      done
                                                                    <u>i le n</u>
                    result
```