EEL 4712
Midterm 2 – Spring 2017
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 (12 points)	
3 (5 points)	
4 (8 points)	
5 (20 points)	
6 (20 points)	
7 (20 points)	
Free	3

Total:

Regrade Info:

1. (4 points) a. Create a VHDL type called *my_type* for a constrained two-dimensional array with 100 rows and 100 columns, where each element is 32 bits.

type my_type is array (0 to 99, 0 to 99) of std_logic_vector(31 downto 0);

(4 points) b. Create a VHDL type called *my_type2* that is an unconstrained array type where each element is 16 bits.

type my_type2 is array (natural range <>) of std_logic_vector(15 downto 0);

(4 points) c. Instantiate an array of type *my_type2* with 50 elements.

signal x : my_type2(0 to 49); -- 49 downto 0 is fine too

2. (4 points) a. What is the name of the FPGA resource that connects routing tracks from different channels?
Switch box/matrix

(4 points) b. What is the name of the FPGA resource that connects CLB I/O to routing tracks?
Connection box

(4 points) c. Briefly explain the difference between a long track and a short track

A long track skips over some amount of switch boxes, whereas a short track connects adjacent switch boxes

3. (5 points) Briefly explain why using gates as a metric for FPGA size is not accurate.

FPGAs have no gates. They have LUTs, which only depend on the number of inputs and outputs of combinational logic.

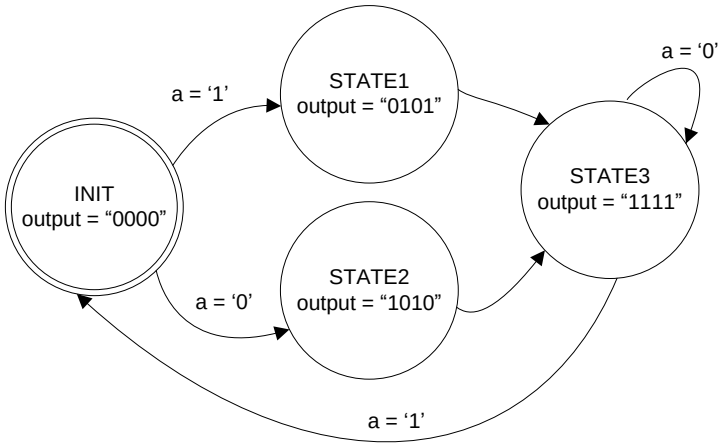
4. (8 points) You are designing a circuit with a 33 MHz clock that generates an output rising edge after it can *guarantee* that an asynchronous control input (e.g., a button) has been asserted for 500 ms. Show the potential range of timings for the output under the assumption that you have no control over when the button is pressed.

Clock period = 1 / 33M

Minimum time = ceil(500 ms / clock period) * clock period = **500 ms**

Maximum time = minimum time + clock period = **500.00003 ms**

5. (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.

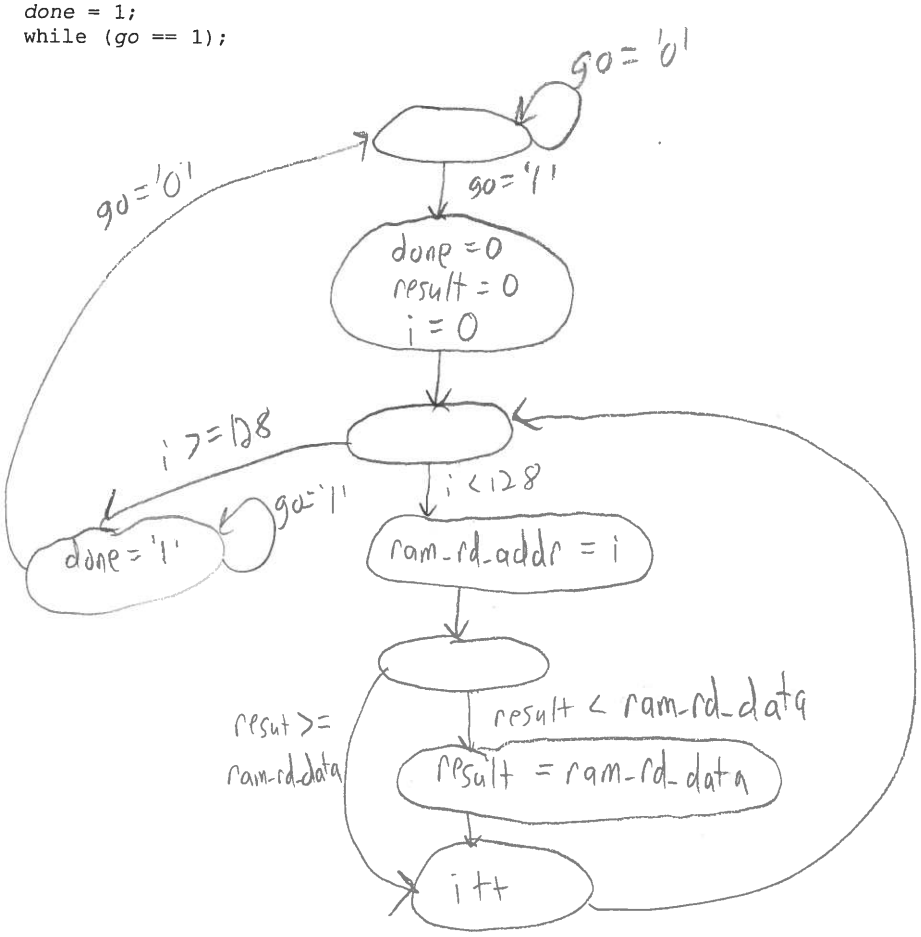



```
library ieee;
use ieee.std_logic_1164.all;

entity fsm is
    port (
        clk, rst          : in  std_logic;
        a                 : in  std_logic;
        output            : out std_logic_vector(3 downto 0)
);
end fsm;

architecture PROC2 of fsm is

    type STATE_TYPE is (INIT, STATE1, STATE2, STATE3);
    signal state, next_state : STATE_TYPE;

begin

    process(clk, rst)
    begin
        if (rst = '1') then
            state <= INIT;
        elsif (rising_edge(clk)) then
            state <= next_state;
        end if;
    end process;

    process(state, a)
    begin

        next_state <= state;

        case state is
            when INIT =>
                output <= "0000";

                if (a = '1') then
                    next_state <= STATE1;
                else
                    next_state <= STATE2;
```

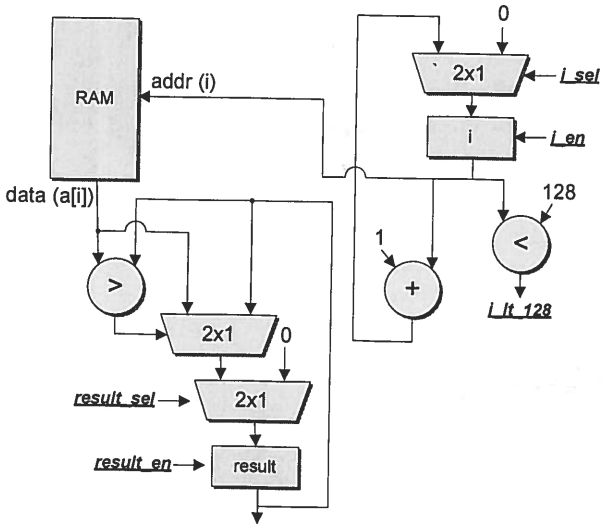
```
                end if;

            when STATE1 =>
                output <= "0101";
                next_state <= STATE3;

                    when STATE2 =>
                output <= "1010";
                next_state <= STATE3;

                    when STATE3 =>
                           output <= "1111";
                           if (a = '1') then
                                  next_state <= INIT;
                           end if;

        end case;
    end process;

end PROC2;
```

6. (20 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. For the array a[i], assume that your circuit has a RAM outside the FSMD that stores the entire array, and that all values are already stored in the RAM. To access a[i], send i to the specified output *ram_rd_addr*. Data from ram will arrive on the *ram_rd_data* input **one cycle** later (i.e., there is a one-cycle read latency).

```
const int N = 128; // In VHDL: generic( N : positive )

Inputs: go (std_logic), ram_rd_data (std_logic_vector)
Outputs: ram_rd_addr (std_logic_vector), result (std_logic_vector), done (std_logic)

int i, result;
int a[128]; // Accessed via ram_rd_addr, data provided via ram_rd_data 1 cycle after address

// reset values for outputs
done = 0; result = 0;

while (1) {

        while (go == 0);
        done = 0;

        result = 0;
        for (i=0; i < 128; i++) {
                if (result < a[i]) {
                        result = a[i];
                }
        }

        done = 1;
        while (go == 1);
}
```

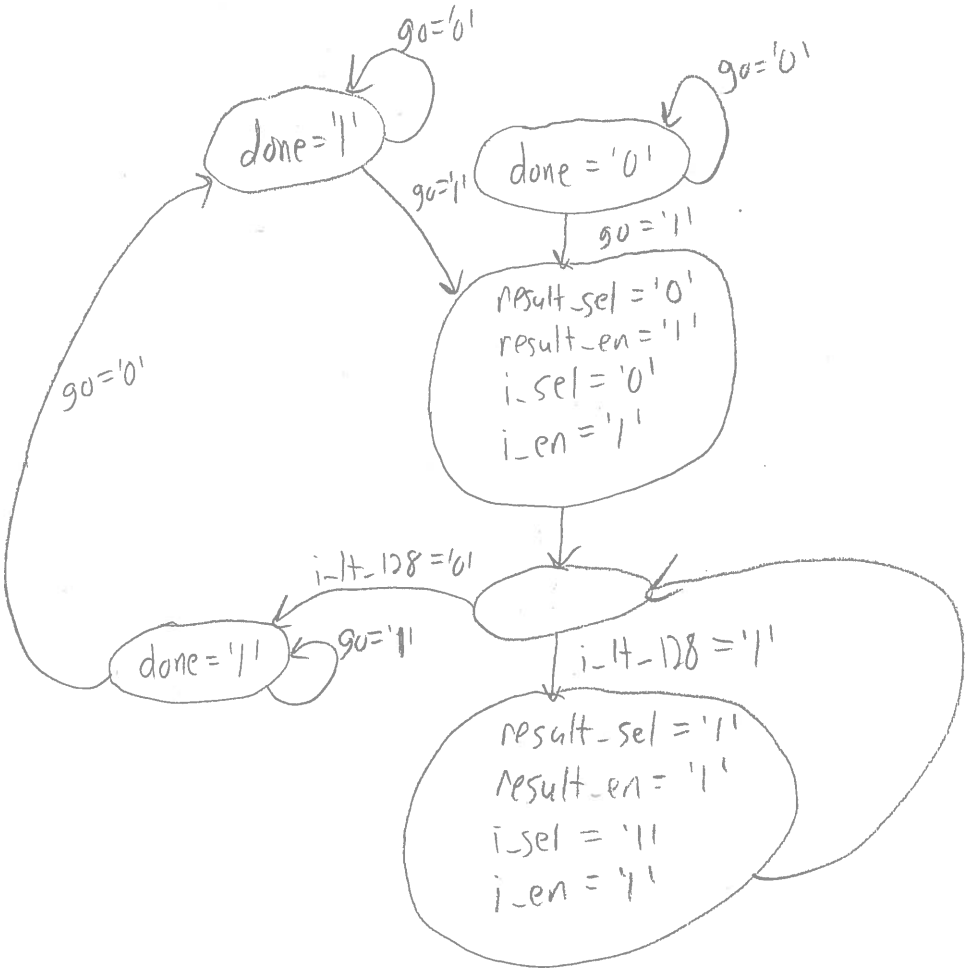


7. (20 points) Draw an FSM capable of controlling the illustrated datapath to perform the pseudo-code in question 6, by assigning or reading from the underlined control signals. Assume that *go* is an input to the controller, done is an output from the controller, and that left mux inputs have a select value of 1. Also assume that RAM contents store the a[] array and that these values have already been stored. Note that this datapath assumes that *N=128*. **Do not write any VHDL code**, just show the FSM and control signals. Be sure to mention default signal values to save space.



defaults

all enables are '0'
all selects are '-'
done = '0'



Problem 6+7 Reference

```
const int N = 128; // In VHDL: generic( N : positive )

Inputs: go (std_logic), ram_rd_data (std_logic_vector)
Outputs: ram_rd_addr (std_logic_vector), result (std_logic_vector), done (std_logic)

int i, result;
int a[128]; // Accessed via ram_rd_addr, data provided via ram_rd_data 1 cycle after address

// reset values for outputs
done = 0; result = 0;

while (1) {

        while (go == 0);
        done = 0;

        result = 0;
        for (i=0; i < 128; i++) {
                if (result < a[i]) {
                        result = a[i];
                }
        }

        done = 1;
        while (go == 1);
}
```