EEL 4712
Midterm 2 – Spring 2019
VERSION 1

Name: SOLUTION

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.**

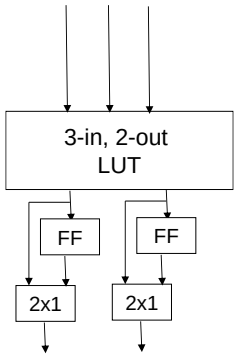
COVER SHEET:

Problem#:	Points
1 (12 points)	
2 (6 points)	
3 (6 points)	
4 (25 points)	
5 (25 points)	
6 (26 points)	

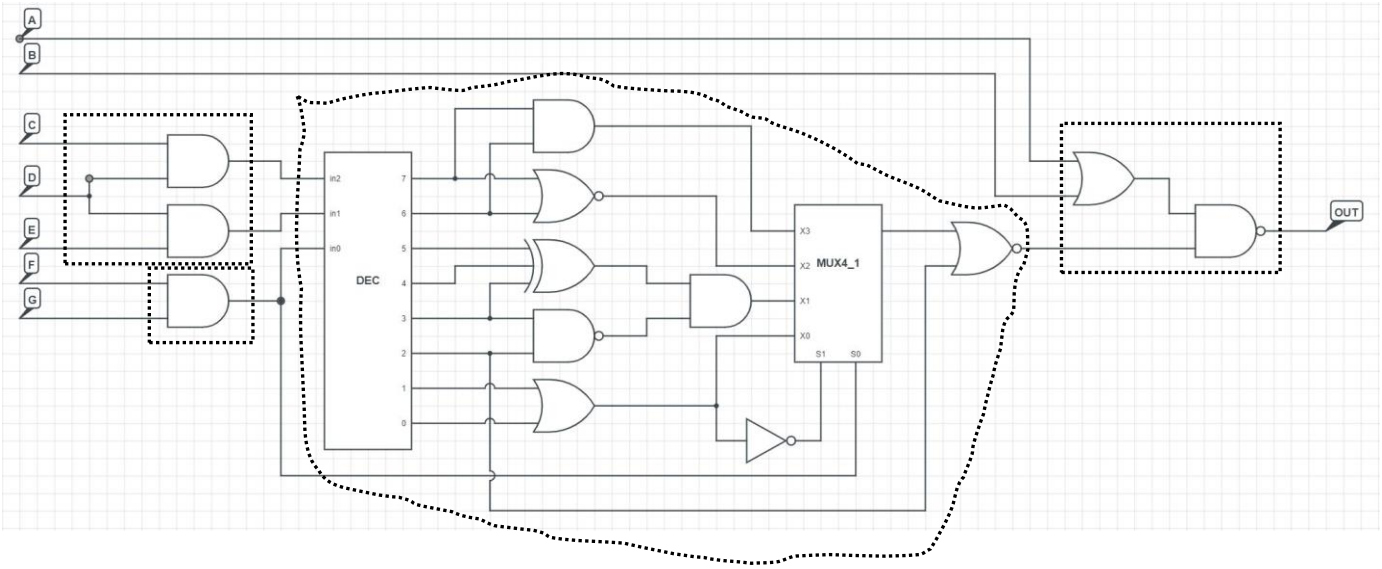
Total:

Regrade Info:

1. (12 points) Assume you are given an FPGA that consists of the following CLB structures with one 3-input, 2-output LUT and optional registers on each output.



Map the following circuit onto these CLBs by drawing boxes to represent CLBs. **Do not try to optimize or modify the circuit in any way.**



2. (6 points) Name three primary FPGA resources, excluding interconnect.

DSPs, LUTs (or CLBs), Block RAM, I/O, PCIe Express, transceivers

3. (6 points) You are designing a circuit that repeatedly outputs a 1-cycle pulse at 6 µs, 11 µs, and 15 µs. Assuming a 50 MHz clock, at what count values should each pulse occur?

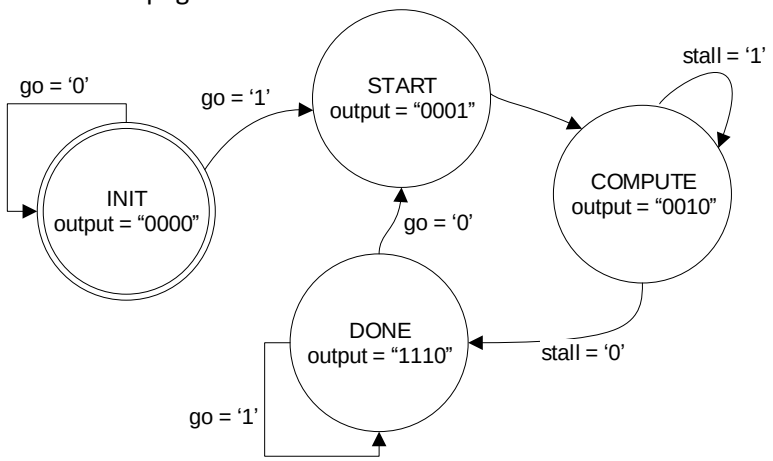
One 50 MHz clock period = 1s / 50M = 20 ns.

Clock periods for 6 µs = 6 µs / 20 ns = **300**

Clock periods for 11 µs = 11 µs / 20 ns = **550**

Clock periods for 15 µs = 15 µs / 20 ns = **750**

4. (25 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.

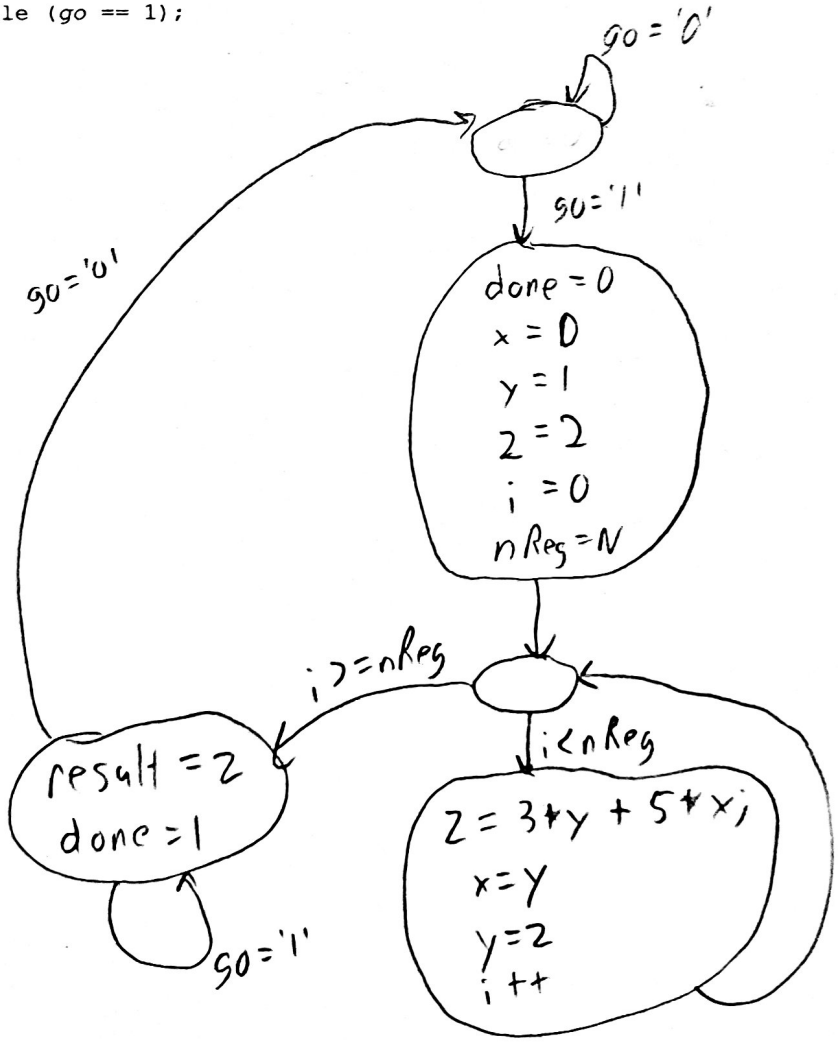



```
library ieee;
use ieee.std_logic_1164.all;

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

architecture PROC2 of fsm is

    type STATE_TYPE is (INIT, START, COMPUTE, DONE);
    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, go, stall)
    begin

        next_state <= state;

        case state is
            when INIT =>
                output <= "0000";
                if (go = '1') then
                    next_state <= START;
                end if;

            when START =>
                output <= "0001";
                next_state <= COMPUTE;

            when COMPUTE =>
                output     <= "0010";
                if (stall = '0') then
```

```
                    next_state <= DONE;
                end if;

            when DONE =>
                output <= "1110";
                if (go = '0') then
                    next_state <= START;
                end if;

        end case;
    end process;

end PROC2;
```

5. (25 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).

```
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;
        y = 1;
        z = 2;
        i = 0;
        nReg = N; // store input N into a register

        while (i < nReg) }

            z = 3*y + 5*x;
            x = y;
            y = z;
            i++;
        }

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



6. (26 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* and an output for *done* (not shown in the datapath). 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.