USAF ACADEMY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ECE 383 GRADED REVIEW #1
SPRING 2016

Name: ______________________________ **Section:** __________

Academic Security	This examination is not released from academic security until **1630** on **Feb 15, 2016**. Until this time, you may not discuss the examination contents or the course material with anyone other than your instructor.
Integrity	Your honor is extremely important. This academic security policy is designed to help you succeed in meeting academic requirements while practicing the honorable behavior our country rightfully demands of its military. Do not compromise your integrity by violating academic security or by taking unfair advantage of your classmates.
Authorized Resources	1. Any calculator
Instructions	▪ **Show all work for full credit.** ▪ Box or circle your final answer. ▪ For all numerical answers, use engineering notation and include units. ▪ Completely label all your diagrams, drawings, graphs, etc. for full credit. ▪ You have the remainder of the period to complete this exam.

Problem	Value	Earned	Course Objective
1a – Analysis	10		Objective 1
1b – Instantiate	10		Objective 1
1c – Design	10		Objective 1
1d – Design	10		Objective 1
2 – Analyze Timing	20		Objective 4
3 – Tech Tradeoffs	10		Objective 7
4 – Analyze Tech	10		Objective 6
5 – FSM Design	10		Objective 1
6 – Testing	10		Objective 2
Total	100		

This page intentionally left blank

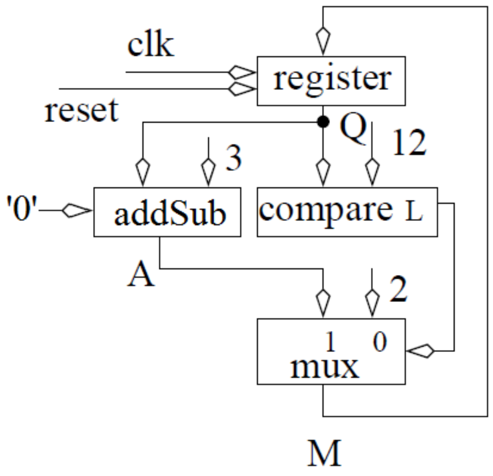
Problem 1 (40 points) *High-Level System Design* [Objective 1]

a. (10 points) Describe the count sequence, until it starts repeating, produced by the counter below after coming out of reset.

```
entity ExamCounter is
    port(   clk, reset: in std_logic;
            q: out unsigned(3 downto 0));
end ExamCounter;

architecture behavior of ExamCounter is
    signal temp: unsigned(3 downto 0);
begin
      process(clk, reset)
      begin
            if (rising_edge(clk)) then
                  if (reset='0') then
                        temp <= "0011";
                  elsif (temp = "0101") then
                        temp <= "1010";
                  elsif (temp = "1101") then
                        temp <= "0010";
                  else
                        temp <= temp + 1;
                  end if;
            end if;
      end process;
      q <= temp;
end architecture;
```

Count sequence =

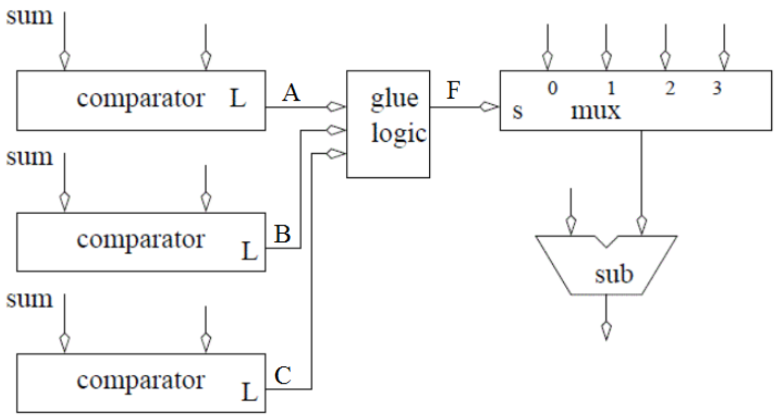
b. (10 points) Using the components below, define the signals and instantiate the block to reproduce the diagram below. You may assume a word size of 5-bits will work for all the signals and that the clk and reset are given in the entity description.



```
architecture structure of exam is

component comp
    generic(N: integer := 4);
    port(x,y : in unsigned(N-1 downto 0);
         g,l,e: out std_logic);
end component;

component reg
    generic(N: integer := 4);
    port(clk, reset: in std_logic;
         d : in unsigned(N-1 downto 0);
         q : out unsigned(N-1 downto 0));
end component;

component addSub
    generic(N: integer:=4);
    port( a,b: in unsigned(N-1 downto 0);
          fun: in std_logic;
          sum: out unsigned(N-1 downto 0));
end component;

component mux2x1
    generic(N: integer := 4);
    port(y0,y1: unsigned(N-1 downto 0);
         s: in std_logic;
         f: out unsigned(N-1 downto 0));
end component;

    signal                                          // Put stuff here
    signal
    signal

begin
                                                    // instantiate here












end structure;
```

c. (10 points) You have a digital design which calls for a circuit that performs the following task (written as a C if/then statement). Complete the architecture below by writing the names/values of the signals on the arrows, and complete the truth table for the glue-logic box. Note that the output column is 2-bits and will have will have some don't cares in it.

```
if       (sum < 5)   z = sum
else if  (sum < 10)  z = sum-5
else if  (sum < 15)  z = sum-10
else                 z = sum-15
```



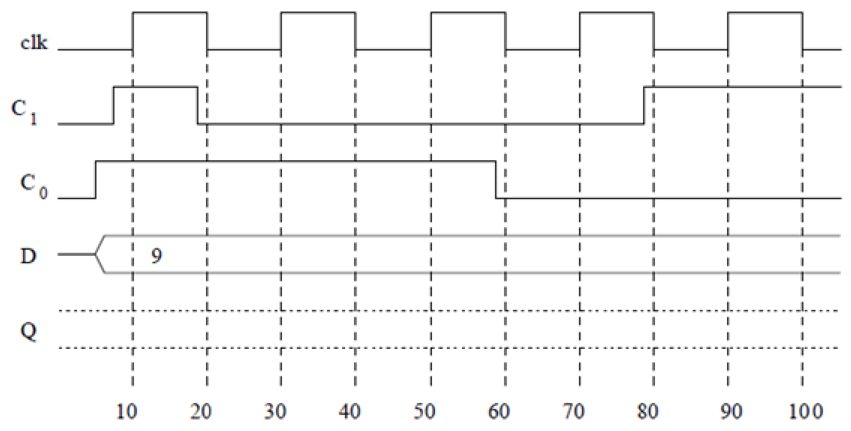
A	B	C	F
0	0	0	.
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

d. (10 points) Draw a block diagram, using the components from problem 1b, for a circuit that counts up from 0 to 127, then back down to 0, repeating this forever.

a. The addSub component adds when func='0' and subtracts when func='1'.
b. You may using discrete logic gates (and, or, not) or describe glue logic using a truth table (like that shown in problem 1b).
c. Your may use a single D flip flop.
d. Your design has a clock and an active low reset.

Problem 2 (20 points) *Timing Analysis* [Objective 4]

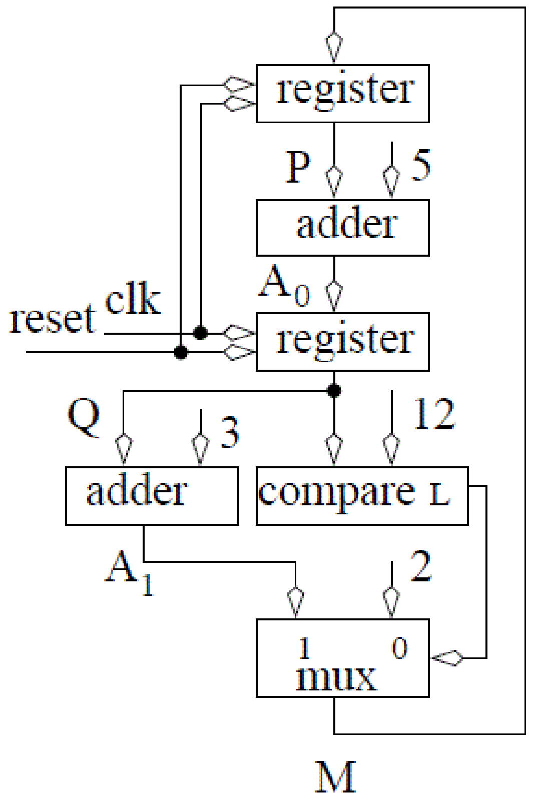
a. (10 points) Assume that a 4-bit up/down counter has the truth table shown at right. Complete the timing diagram below.



clk	C_1C_0	D	Q+
X	XX	X	0
0,1,falling	XX	X	Q
rising	00	X	Q
rising	01	X	Q-1
rising	10	X	Q+1
rising	11	D	D

b. (10 points) Determine the maximum clocking frequency of the circuit shown below under the timing conditions given in the table below. Clearly show your work.

Component	Propagation Delay
Comparator	25nS
Mux	10nS
Adder	35nS
Register	Propagation = 10nS
Register	Setup time = 5nS



Problem 3 (10 points) *Technology Tradeoffs* [Objective 7]

Volume of sale (i.e., the number of parts sold) is a factor when determining which device technology is to be used. Assume that a system can be implemented by an FPGA or standard-cell technology. The per-part cost is $55 and $2 for FPGA and standard cell respectively. Standard-cell technology also involves a one-time mask generation cost of $125,000 respectively.

a. (6 points) Assume that number of parts sold is *N*. Derive the equation of per-unit cost for standard-cell and FPGA.

$\text{Cost}_{\text{FPGA}}$: ______________________________

$\text{Cost}_{\text{Standard Cell}}$: ______________________________

b. (4 points) In what range of *N* is it best to use an FPGA?

Problem 4 (10 points) *Analyze Technology* [Objective 6]

a. (6 points) Write in "best" and "worst" for each technology category below.

	NRE Cost	Performance	Rapid Prototype
ASIC			
FPGA			
Microcontroller			

b. (2 points) Name two benefits of an ASIC over an FPGA.

c. (2 points) Name two benefits of a Microcontroller over FPGA.

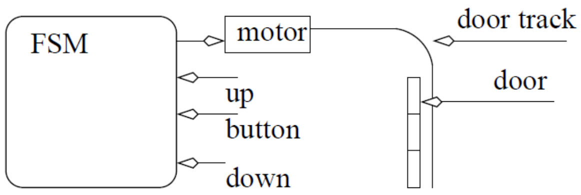
Problem 5 (10 points) *Finite State Machine Design* [Objective 1]

Build a **Moore** FSM which controls an automatic garage door opener. If the door is closed and you press the button, the motor moves the door all the way up and then it stops. If the door is open and you press the button, the motor moves the door all the way closed and then it stops.

The garage door FSM has three bits of input. The first input comes from the main control button used to open or close the garage door. When pressed, the button outputs 1, otherwise it outputs 0. The garage door rides in a track with two limit switches, one at the top and the other at the bottom. The top limit switch outputs 1 when the garage door is all the way up, otherwise its outputs 0. The bottom limit switch outputs 1 when the garage door is all the way down, otherwise its outputs 0.

Your FSM will need 2-bits of output to control the motor. When the motor output is 01, the door slowly moves upward, opening the door. When motor = 10, the door moves slowly downward, closeing the garage door. When motor = 00, the motor is turned off and the door does not move.

Draw a FSM to control the garage door (make sure your diagram is complete and unequivocal).
Draw a control word table, defining the output for each state.



Problem 6 (10 points) *Testing and Verification* [Objective 2]

Modify the testbench below to exercise the counter in problem 2 as shown in the timing diagram of the same problem. Reasonable estimates for signal transition times will be acceptable. You should assume that the units on the horizontal axis are nanoseconds.

```
ARCHITECTURE behavior OF exam_tb IS

    COMPONENT counter
                Port(   clk: in  STD_LOGIC;
                        reset : in  STD_LOGIC;
                        C1, C0: in std_logic;
                        D: in unsigned (3 downto 0);
                        Q: out unsigned (3 downto 0));
    END COMPONENT;

        signal clk : std_logic;
        signal reset : std_logic;
        signal c1, c0 : std_logic;
        signal D : unsigned(3 downto 0);
        signal Q : unsigned(3 downto 0);
        constant clk_period : time := __________ ns;         // **
BEGIN

   uut: counter
   PORT MAP (clk=>clk, reset=>reset, C1=>c1, C0=>c0, D=>D, Q=>Q);

   clk_process :process
   begin
          clk <= '0';
          wait for clk_period/2;
          clk <= '1';
          wait for clk_period/2;
   end process;

   c1 <=                                                     // **


   c0 <=                                                     // **

   D <=                                                      // **

END;
```