EEL 4712
Midterm 3 – Spring 2015
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.**

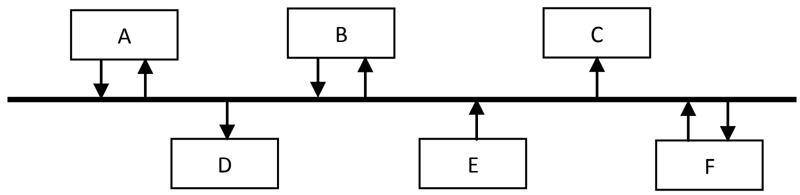
COVER SHEET:

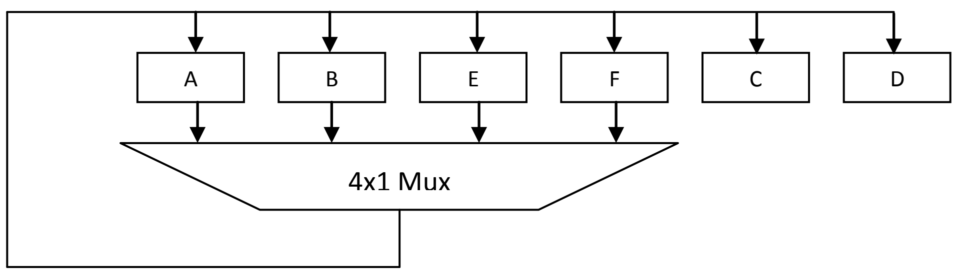
Problem#:	**Points**
1 (10 points)	
2 (5 points)	
3 (5 points)	
4 (5 points)	
5 (5 points)	
6 (5 points)	
7 (5 points)	
8 (15 points)	
9 (10 points)	
10 (5 points)	
11 (10 points)	
12 (20 points)	

Total:

Regrade Info:

1) (10 points) Draw a schematic of the FPGA mux-based circuit that would be synthesized for the following bus structure. Assume there are tri-states at every location that writes to the bus. Show how the A-F components connect to the inputs and outputs of the mux. You can omit control signals.





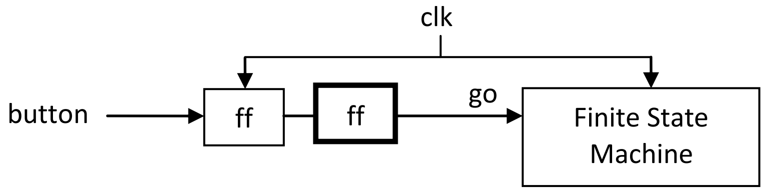
2) (5 points) Why does a dual-flop synchronizer work for synchronizing single bits, but not for multiple bits?

With a dual flop, the output is either the previous value or the correct value. For multiple bits, the output could be incorrect.

3) (5 points) In what situation can a dual-flop synchronizer be used for multiple bits? Describe the *situation*, don't just give an example of a synchronizer that does this.

When the input is guaranteed to change by only 1 bit at a time (e.g. gray coding).

4) (5 points) In many of the labs, you used a circuit similar to the following one. Extend this circuit to prevent metastability from propagating into the finite state machine



5) (5 points) Name two synchronizers that can correctly handle multiple bits.

FIFO, handhake, mux recirculation

6) (5 points) Explain why the SBCR (subtract with borrow) instruction mathematically requires the SETC (set carry) instruction to be executed first.

Two's complement subtraction: output = in1 + not(in2) + 1
SBCR: output = in1 + not(in2) + C

If C != 1, then SBCR does not do a correct subtraction.

7) (5 points) In what situation would a program not use a SETC carry instruction before the SBCR instruction?

When doing subtraction with inputs larger then 8 bits. For example, a 16-bit subtraction would do:

SETC
SBCR
SBCR
where the second subtraction potentially uses a borrow from the first subtract.

8) (15 points) Create a memory initialization file for the following assembly code. Add a comment to show the beginning of each instruction and each variable in memory. B*reak your answer up into two columns and/or use the following page.*

```
OUTPORT0        EQU     $FFFE

BEGIN:
        LDAI    $03
        STAA    COUNT
        LDXI    BUFF
        LDAI    $00
        STAR    D
        CLRC
AGAIN:
        LDAA    0,X
        ADCR    D
        STAR    D
        LDAA    COUNT
        DECA
        STAA    COUNT
        INCX
        BNEA    AGAIN
        LDAD
        STAA    OUTPORT0

INFINITE_LOOP:
        CLRC
        BCCA    INFINITE_LOOP

* Data Area

BUFF:   dc.b    $01
        dc.b    $02
        dc.b    $03

COUNT:  ds.b    1
```

```
Depth = 256;
Width = 8;
Address_radix = hex;
Data_radix = hex;
% Program RAM Data %
Content
  Begin


0000: 84; --LDAI
0001: 03;
0002: F6; --STAA
0003: 26; --countL
0004: 00; --countH
0005: 8A; --LDXI
0006: 23; --buffL
0007: 00; --buffH
0008: 84; --LDAI
0009: 00;
000A: F1; --STAR
000B: F9; --CLRC
000C: BC; --LDAA 0,X AGAIN:
000D: 00;
000E: 01; --ADCR
000F: F1; --STAR
0010: 88; --LDAA
0011: 26; --countL
```

```
0012: 00; --countH
0013: FB; --DECA
0014: F6; --STAA
0015: 26; --countL
0016: 00; --countH
0017: FC; --INCX
0018: B4; --BNEA
0019: 0C;
001A: 00;
001B: 81; --LDAD
001C: F6; --STAA
001D: FE;
001E: FF;
001F: F9; --CLRC, INFINITE_LOOP:
0020: B0; --BCCA
0021: 1F;
0022: 00;
0023: 01; --BUFF[0], DATA_SEGMENT
0024: 02; --BUFF[1]
0025: 03; --BUFF[2]
0026: 01; --COUNT
```

```
[      ..00FF] : 00;
End;
```

9) (10 points) Describe what would happen during a simulation of the following 2-process FSMD when the state reaches S_COUNT. Hint: this is the exact same code used in class.

```
architecture bhv of fsmd is

  type STATE_TYPE is (S_START, S_COUNT, S_DONE);
  signal state, next_state : STATE_TYPE;
  signal count             : unsigned(3 downto 0);
  constant MAX_COUNT_VAL : natural := 10;

begin

  process (clk, rst)
  begin
    if (rst = '1') then
      state <= S_START;
    elsif (clk = '1' and clk'event) then
      state <= next_state;
    end if;
  end process;

  process(go, state, count)
  begin

    case state is
      when S_START =>

        done  <= '0';
        count <= to_unsigned(1, count'length);

        if (go = '0') then
          next_state <= S_START;
        else
          next_state <= S_COUNT;
        end if;

      when S_COUNT =>

        done  <= '0';
        count <= count + 1;

        if (count = MAX_COUNT_VAL) then
          next_state <= S_DONE;
        else
          next_state <= S_COUNT;
        end if;

      when S_DONE =>

        count      <= to_unsigned(MAX_COUNT_VAL, count'length);
        done       <= '1';
        next_state <= S_DONE;

      when others => null;
    end case;

  end process;
end bhv;
```

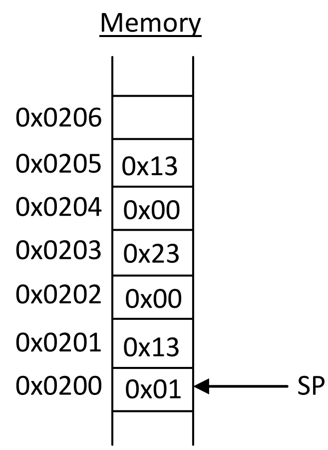
Process is sensitive to a signal that it also defines. This causes an infinite simulation loop.

10) (5 points) Assuming that the stack pointer is initially set to address 0x0206, show the state of the stack and stack pointer immediately after the CALL FUNCTION3 instruction. Assume that none of the instructions executed between function calls are returns.

```
            CALL FUNCTION1        * addr 0x0010
            ….

FUNCTION1:  ….
            ….
            CALL FUNCTION2        * addr 0x0020
            ….

FUNCTION2:  ….
            ….
            CALL FUNCTION3        * addr = 0x0110
            ….
```



11) a. (5 points) Given a solution space with the following implementations, which of the solutions are *not* Pareto optimal? If they are all Pareto optimal, state that.

  a. Area: 5000 LUTs, Time: 3s
  b. Area: 4000 LUTs, Time: 2s
  c. Area: 3000 LUTs, Time 5s
  d. Area: 2000 LUTs, Time 6s
  e. Area: 1000 LUTs, Time 8s

b. (5 points) Given a solution space with the following implementations, which of the solutions are *not* Pareto optimal? If they are all Pareto optimal, state that.

  a. Area: 5000 LUTs, Time: 3s, Energy=10mJ
  b. Area: 4000 LUTs, Time: 2s, Energy=15mJ
  c. Area: 3000 LUTs, Time 5s, Energy=20mJ
  d. Area: 2000 LUTs, Time 6s, Energy=25mJ
  e. Area: 1000 LUTs, Time 8s, Energy=30mJ

They are all Pareto optimal. *a* is worse in both area and time, but best in energy.

12) a. (5 points) For the following code, create a schedule for the provided datapath. Ignore muxes and other glue logic. Like the examples in class, assume that address calculations are done *without* using the specified resources (i.e., address calculations cost nothing). Do not change the code. List any assumptions.

```
for (int i=0; i < 1000000; i++) {
   a[i] = b[i]*22 + b[i+1]*28 + b[i+2]*54 + b[i+3]*97;
}
```

a e b f c g d

<u>Datapath</u>
2 multipliers
1 adder
1 comparator
1 memory for b[] (can read 2 elements/cycle)
1 memory for a[] (can write 1 element/cycle)

0) i=0
1) i < 1M
2) load b[i], b[i+1]
3) load b[i+2],b[i+3], a, b
4) c, d, e
5) g
6) f
7) i++

b. (5 points) What is the execution time in total cycles based on your schedule from part a? Show your work.

Each iteration = 7 cycles, Total iterations = 1M
Execution time = 7 *1M = 7M cycles (actually 7M+1, but the +1 is negligible)

c. (5 points) Create a new schedule for the same code and datapath, except this time using 4 multipliers and 2 adders.

0) i=0
1) i < 1M
2) load b[i], b[i+1]
3) load b[i+2],b[i+3], a, b
4) c, d, e
5) g
6) f
7) i++

This is the same schedule.

d. (5 points) Given a solution space consisting of only the solutions from *a* and *c*, is *c* a Pareto optimal solution? Explain your answer.

C is not Pareto optimal because it has the same performance, but more area.