EEL 4712
Midterm 3 – 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 (3 points)	
2 (3 points)	
3 (3 points)	
4 (3 points)	
5 (3 points)	
6 (3 points)	
7 (3 points)	
8 (3 points)	
9 (6 points)	
10 (3 points)	
11 (3 points)	
12 (8 points)	
13 (12 points)	
14 (5 points)	
15 (6 points)	
16 (26 points)	
17 (4 points)	
18 (3 points)	3

Total:

Regrade Info:

1) (3 points) In general, FPGA synthesis tools convert tri-state buffers into multiplexors. Briefly explain why the tools perform this conversion.

FPGAs don't internally have tri-state buffers.

2) (3 points) Write enables for a bus often use a one-hot encoding, where at most one source is assumed to be given access at any time. For a 4-source bus, what component can be added to guarantee this assumption is always true?

2:4 decoder

3) (3 points) How long does it take for a metastable signal to stabilize (circle one)?
   a. 1 cycle
   b. 10 cycles
   c. 100 cycles
   d. Unknown

4) (3 points) Explain why a dual-flop synchronizer provides a shorter mean-time between failures as the destination clock frequency increases.

One cycle is a shorter amount of time at higher frequencies

5) (3 points) Describe when to use a FIFO instead of a handshake synchronizer.

   When bandwidth/throughput is important.

6) (3 points) Explain why a dual-flop synchronizer *can* be used to synchronize multiple bits when only a single input bit changes.

   The output is either the previous value or the correct value, but never an incorrect value.

7) (3 points) FPGA virtualization addresses which of the following problems (circle all that apply):
   a. Productivity
   b. Power consumption
   c. Compile times
   d. Application portability

8) (3 points) What is the name of the emerging area of research that looks at sacrificing accuracy to improve performance and/or energy?

Approximate computing

9) a. (3 points) For the MIPS, a function call is usually implemented with what instruction?

JAL

b. (3 points) For the MIPS, a return from a function uses what instruction?

JR

10) (3 points) What operation does the ALU perform for a load word instruction?

add

11) (3 points) Briefly explain what an *immediate* is in a MIPS instruction.

A value used by an instruction that is directly provided by the instruction itself.

12) (8 points) Write MIPS assembly code that matches the following behavior. Inport0 corresponds to byte address 0xFFF8. Inport1 and Outport corresponds to byte address 0xFFFC. Use $0-$31 for registers instead of the normal MIPS naming convention. Make two columns if necessary.

```
x = Inport0;
y = Inport1;
z = 0;
while (x != y) {
      x++;
      z++;
}
Outport = z;
```

```
lw $1, 0xFFF8($0)
lw $2, 0xFFFC($0)
addui $3, $0, 0
LOOP:
bne $1, $2, DONE
addui $1, $0, 1
addui $3, $0, 1
j LOOP
DONE:
sw $3, 0xFFFC($0)
```

13) (12 points) Create a memory initialization file for the following assembly code. Add comments as necessary. Put a small space between different instruction fields to make it easier to read.

```
        lw $1, 0xFFF8($0)
        addiu $2, $0, 10
        blez $1, ELSE
        addu $3, $0, $2
        j STORE
ELSE:
        addiu $3, $0, 15
STORE:
        sw $3, 0xFFFC($0)
DONE:
        j DONE
```

```
Depth = 256;
Width = 32;
Address_radix = hex;
Data_radix = bin;
% Program RAM Data %
Content
Begin

00: 100011 00000 00001 1111111111111000;
01: 000000 00000 00010 0000000000001010;
02: 000110 00001 00000 0000000000000010;
03: 000000 00011 00000 0000000000100001;
04: 000010 00000 00000 0000000000000110;
05: 000000 00000 00011 0000000000001111;
06: 101011 00000 00011 1111111111111100;
07: 000010 00000 00000 0000000000000111;

End;
```

14) (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. LUTs: 500, DSPs: 15, Time: 18s
    b. LUTs: 1000, DSPs: 5, Time: 18s
    c. LUTs: 1500, DSPs: 22, Time: 15s
    d. LUTs: 2000, DSPs: 20, Time: 12s
    e. LUTs: 10, DSPs: 300, Time: 11s

ALL PARETO OPTIMAL

15) (4 points) a. For the solution space in problem 14, give one new implementation that would prove that none of the previous implementations are Pareto optimal.

LUTS: 1, DSPs: 1, Time: 1 s

(4 points) b. What implementation from problem 14 would be best for an optimization goal of minimizing LUTs given a time constraint of 15s and a DSP constraint of 20?

d

16) a. (8 points) For the following code, create a schedule for the provided datapath. Ignore muxes, registers, 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. Do not unroll or pipeline the loop. List any assumptions.

```
for (int i=0; i < 1000000; i++) {
   a[i] = b[i]*11 + b[i+1]*22 + b[i+2]*33 + b[i+3]*44;
}
```

Datapath
4 multipliers
2 adders
1 comparator
1 memory for b[] (can read 4 elements/cycle)
1 memory for a[] (can write 1 element/cycle)

0) i = 0
1) i < 1000000, load b[i] … b[i+3]
2) c, d, e, f // c = b[i] * 11, d = b[i+1]*22, e = b[i+2]*33, f = b[i+3]*44
3) c+d, e+f
4) (c+d) + (e+f)
5) store a[i], i++

b. (4 points) What is the execution time in total cycles based on your schedule from part a? Show your work.

time = 1 + 1,000,000 iterations * 5 cycles/iteration = ~5,000,000 cycles

c. (4 points) What is the execution time in total cycles after unrolling the loop once (i.e. replicating the datapath).

Time = 1 + 1,000,000/2 * 5 = ~2,500,000 cycles

d. (4 points) For a pipelined implementation of the datapath in part a with no unrolling, what is the approximate execution time in total cycles? Show your work.

Time = time for 1st iteration + remaining iterations = ~4 cycles + 999,9999 = ~1,000,000 cycles

e. (4 points) For a pipelined implementation of the datapath in part a after unrolling the loop once, what is the approximate execution time in total cycles? Show your work.

Time = 4 + (1,000,000 – 2) / 2 = ~500,000 cycles

17) a. (2 points) Create a VHDL type *my_array* for a one-dimensional unconstrained array where each element is 16 bits.

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

b. (2 points) Create a signal *array1* that is an instance of *my_array* with 100 total elements.

signal array1 : my_array(0 to 99);

18) (3 points) Free points just because.