EEL 4712
Midterm 3 – Spring 2018
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 (3 points)	
2 (3 points)	
3 (4 points)	
4 (3 points)	
5 (3 points)	
6 (6 points)	
7 (3 points)	
8 (3 points)	
9 (3 points)	
10 (3 points)	
11 (3 points)	
12 (8 points)	
13 (12 points)	
14 (5 points)	
15 (8 points)	
16 (24 points)	
17 (6 points)	6

Total:

Regrade Info:

1) (3 points) **True/false**. During FPGA synthesis, a bus with 4 sources connected by 4 tristates gets replaced by an 8x1 mux.

false

2) (3 points) What causes a flip-flop output to become metastable?

When the input changes during the setup and hold window

3) (4 points) What are the two situations where metastability is unavoidable?

1) asynchronous inputs, 2) clock-domain crossing

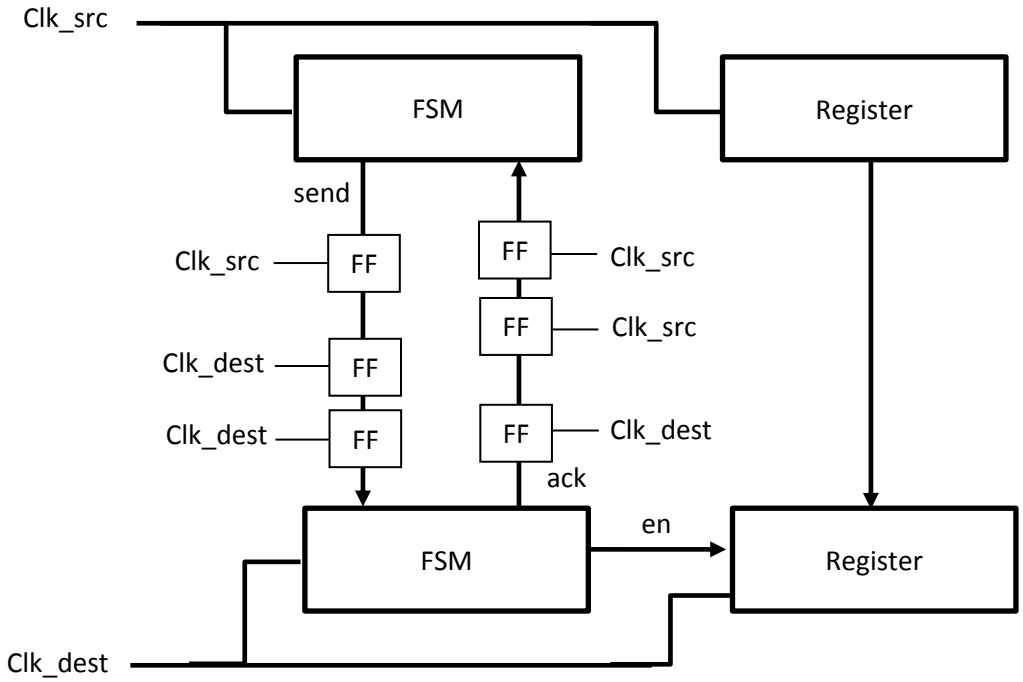
4) (3 points) You are designing a circuit that feeds data to a pipeline from an external memory running on a different clock. What type of synchronizer should you use for this signal?

FIFO

5) (3 points) You are designing a circuit that occasionally transfers a multi-bit control signal across clock domains. What type of synchronizer should you use for this signal?

handshake

6) (6 points) Complete the handshake architecture shown below by adding flip flops to the send and ack signals. Connect each flip flop to the appropriate clock.



7) (3 points) A MIPS Jump and Link instruction stores a return address into what register?

31

8) (3 points) What value is always stored in MIPS register 0?

0

9) (3 points) MIPS jump instructions specify a target using an instruction index instead of a byte address. How does the datapath convert this instruction index into a byte address? (Describe high-level functionality, not specific control signals)

Word address = byte address / 4

10) (3 points) What MIPS instructions set the HI and LO registers?

Multiplication

11) (3 points) What range of byte addresses get mapped to the RAM inside of the memory entity?

0 to 1023

12) (8 points) Write MIPS assembly code that matches the following behavior. Inport0 corresponds to byte address 0xFFF8. Outport corresponds to byte address 0xFFFC. Use $r0-$r31 for registers. Make two columns if necessary.

```
x = inport0;
if (x > 10) {
      y = 10;
else
      y = 15;
}
outport = y;
```

```
lw $r1, 0xFFF8($r0)
addui $r2, $r0, 10
blez $r1, $r2, ELSE
addu $r3, $r0, $r2
j DONE
ELSE:
addiu $r3, $r0, 15
DONE:
sw $r3, 0xFFFC($r0)
```

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 $r1, 0x000F($r0)
        addiu $r2, $r1, 0x7
        addiu $r3, $r0, 0xA
        xor $r4, $r2, $r3
        sw $r4, 0xFFFC($r0)
DONE:
        j DONE
```

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

00 : 100011 00000 00001 0000000000001111
01 : 001001 00001 00010 0000000000000111
02 : 001001 00000 00011 0000000000001010
03 : 000000 00010 00011 00100 00000 100110
04 : 101011 00000 00100 1111111111111100
05 : 000010 00000000000000000000000000101

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. Area: 500 LUTs, Time: 18s
b. Area: 700 LUTs, Time: 20s
c. Area: 2000 LUTs, Time: 12s
d. Area: 2500 LUTs, Time: 17s
e. Area: 3000 LUTs, Time: 8s

15) (4 points) a. For the solution space in problem 14, what tradeoff (a-e) would be best if the optimization goal was to minimize execution time with a LUT constraint of 2500 LUTs?

c

(4 points) b. What tradeoff would be best for an optimization goal of minimizing LUTs given a time constraint of 10s?

e

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 < 10000; i++) {
   a[i] = b[i]*10 + b[i+1]*20 + b[i+2]*30 + b[i+3]*40;
}
```

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 < 10000, load b[i] … b[i+3]
2) c, d, e, f // c = b[i] * 20, d = b[i+1]*20, e = b[i+2]*30, f = b[i+3]*40
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 + 10,000 iterations * 5 cycles/iteration = ~50,000 cycles

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

Time = 1 + 10,000/2 * 5 = ~25,000 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 + 9999 = ~10,000 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 + (10,000 – 2) / 2 = ~5,000 cycles

17) (6 points) Free points just because.