CSM 61C
Spring 2019

Midterm 2 Review
Mentoring 11: April 1, 2019

1 Lets Adder All Up

1.1 Consider the 4-bit adder shown below. It takes:

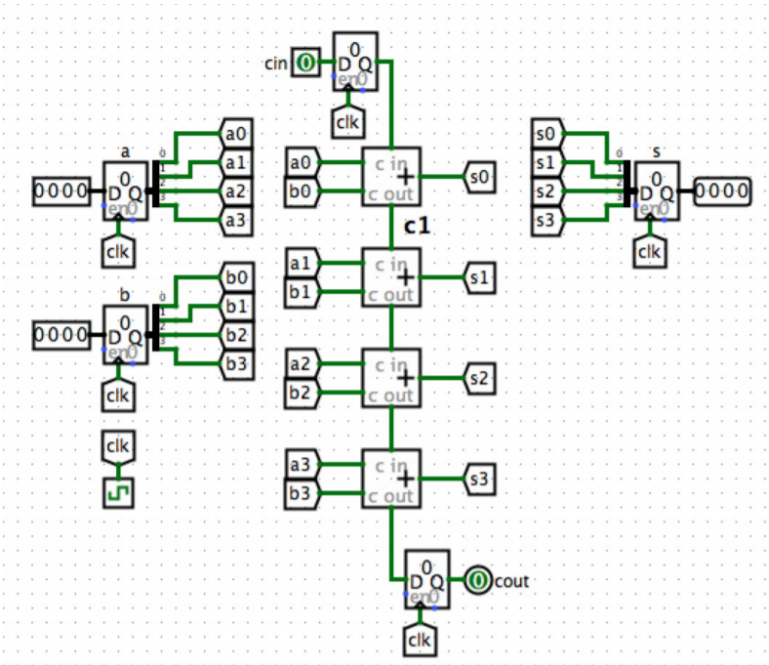
- a carry in (cin)
- two 4-bit inputs:
  - a with bits `a0, a1, a2, a3`
  - b with bits `b0, b1, b2, b3`

And it outputs:

- a carry out (cout)
- one 4-bit input: s with bits `s0, s1, s2, s3`

Assume each adder has a delay of 10ns, and any registers have a clk-to-q, hold time, and setup time of 5ns. Assume the inputs are driven by registers, and outputs are registers as well.

Assume each adder has a delay of 10ns, and any registers have a clk-to-q, hold time, and setup time of 5ns. Assume the inputs are driven by registers, and outputs are registers as well.



1.2 Write Boolean formulas for `s0` and `c1` in terms of the inputs `cin, a0, and b0`. You may use `XOR` as an operator in the Boolean formulas. Each formula should use as few operators as possible.

1.3 What is the critical path delay of the circuit? Please include proper units in your answer.

1.4 What is the maximum clock frequency at which the circuit will function correctly? Please include proper units in your answer.

1.5 What is the maximum hold time the output registers could have at which the circuit would still function correctly?

2 Hazardous Conditions

Assume that we have a standard 5-stage pipelined CPU with no forwarding. Register file writes can happen before reads, in the same clock cycle. We also have comparator logic that begins at the beginning of the decode stage and calculates the next PC by the end of the decode stage. There is no branch delay slot (as in RISC-V). The remainder of the questions pertains to the following piece of MIPS code. Note that MIPS and RISC-V have basically identical instruction syntax but MIPS uses $ for the registers.

	Instructions	Cycle									
		1	2	3	4	5	6	7	8	9	10
0	`start: addu $t0 $t1 $t4`	IF	D	EX	MEM	WB					
1	`addiu $t2 $t0 0`		IF	D	EX	MEM	WB				
2	`ori $t3 $t2 0xDEAD`			IF	D	EX	MEM	WB			
3	`beq $t2 $t3 label`				IF	D	EX	MEM	WB		
4	`addiu $t2 $t3 6`					IF	D	EX	MEM	WB	
5	`label: addiu $v0 $0 10`						IF	D	EX	MEM	WB

2.1 For each instruction dependency below (the line numbers are given), list the type of hazard and the length of the stall needed to resolve the hazard (for only those two instructions). If there is no hazard, say no hazard.

```
0 --> 1: addu $t0 $t1 $t4 --> addiu $t2 $t0
0 --> 3: addu $t0 $t1 $t4 --> beq $t2 $t3 label
1 --> 3: addiu $t2 $t0 0 --> beq $t2 $t3 label
2 --> 3: ori $t3 $t2 0xDEAD --> beq $t2 $t3 label
3 --> 4: beq $t2 $t3 label --> addiu $t2 $t3 6
```

2.2 Now assume that our CPU now has forwarding implemented as presented in class. Which of these instruction dependencies would cause a pipelining hazard?

```
A. 2 --> 3: ori $t3 $t2 0xDEAD --> beq $t2 $t3 label
B. 2 --> 4: ori $t3 $t2 0xDEAD --> addiu $t2 $t3 6
C. 2 --> 5: ori $t3 $t2 0xDEAD --> addiu $v0 $0 10
D. 3 --> 4: beq $t2 $t3 label --> addiu $t2 $t3 6
E. None of the above
```

3 Cache Rules Everything Around Me $

3.1 You are given a RISC-V machine with a single level of 2KiB direct-mapped cache with 512B cache blocks. It has 1MiB of physical address space.

The function foo is ran on the system with a cold cache and as the only process:

```
#define ARRAY_LEN 4096
#define STEP_SIZE 64

// A starts at 0x10000
// B starts at 0x20000
foo(int* A, int* B) {
    int total = 0;
    for (int i = 0; i < ARRAY_LEN; i += STEP_SIZE ) {
        total += A[i];
        total -= B[i];
    }
}
```

Assume `sizeof(int)` returns 4, and that `total` and `i` are located in registers.

Calculate the number of Tag, Index, and Offset bits for this cache.

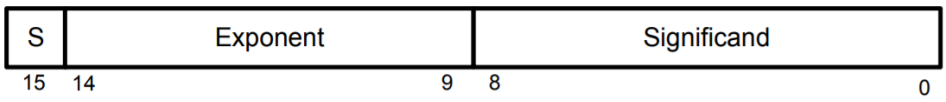
3.2 Calculate the hit percentage for this cache after running `foo`.

3.3 The cache is now cleared and the code is run again. This time, `A` and `B` are pointing to the same array, which starts at `0x10000`. Calculate the new hit percentage.

3.4 Assume A and B starts once again with a cold cache at `0x10000` and `0x20000`. What is the new hit percentage if we ran `foo` on a fully associative cache, with all other parameters staying the same?

4 Dont Let Your Mind Float Away Now

Consider the following 16-bit representation for floating point numbers:



Bits per field: Sign: 1, Exponent: 6, Significand: 9

Everything else follows the IEEE standard 754 for floating point, except in 16 bits. The bias is **31**.

4.1 Convert -15.125 into floating point. Write your answer in hexadecimal.

4.2 What is the value of the largest odd number that can be represented by the above floating point representation?

4.3 How many positive, real numbers can be represented?

5 Finite State Machines

5.1 Construct an FSM that outputs ones until 3 consecutive zeroes have appeared, at which point it outputs zeroes. How many states do we need?

5.2 How many states would we need to construct an FSM that outputs ones until n non-consecutive zeroes have appeared?

5.3 Construct an FSM that returns the XOR of all the inputs.