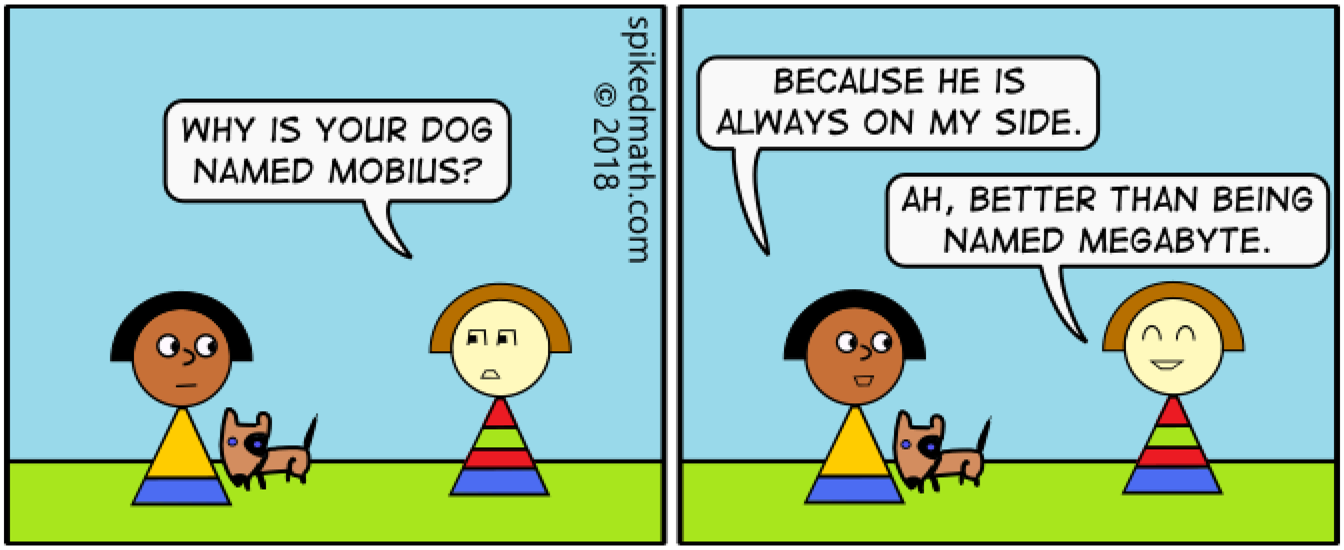
UC Berkeley EECS151 Fall 2019 Midterm 2

Your Name (first last) ____________________ SID ____________________

← Name of person on left (or aisle) ____________________ TA name ____________________ Name of person on right (or aisle) → ____________________

Fill in the correct circles & squares completely…like this: ● (select ONE), and ■(select ALL that apply)

Question	**1**	**2**	**3**	4	**Total**
Minutes	20	26	12	22	**80**
Max Points	16	24	12	18	**70**
Points					

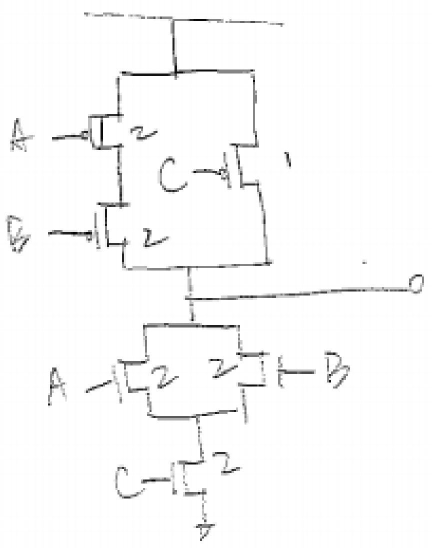

WHY IS YOUR DOG NAMED MOBIUS?
spikedmath.com © 2018
BECAUSE HE IS ALWAYS ON MY SIDE.
AH, BETTER THAN BEING NAMED MEGABYTE.

1) ***It's all logical…*** **(16 points, 20 minutes)**

a) The following function F that implements an OR-AND-Invert OAI21 gate.

$$F = \overline{(A+B)C}$$

Implement the function F as standard, complementary CMOS logic gate. Size your transistors to match the pull-up/pull-down resistances to those of a unit inverter. You can assume NMOS and PMOS devices have equal strength in this technology.

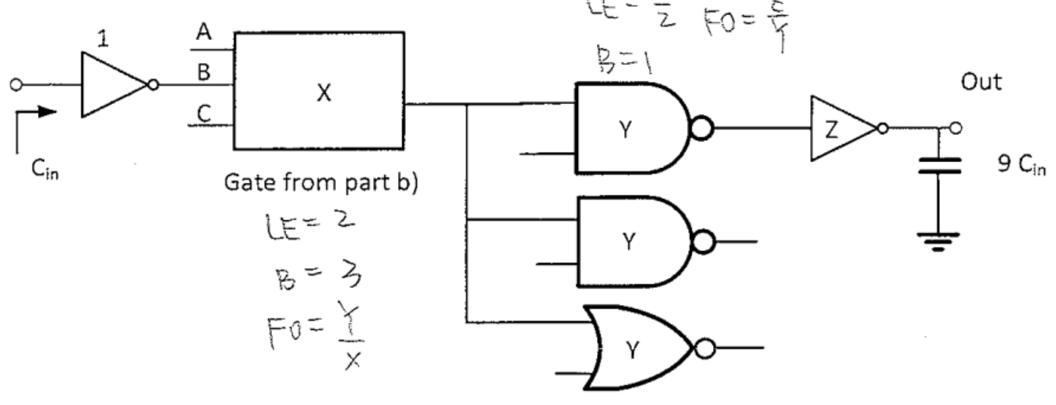


b) Find the logical effort for the input B for the gate from part a).

LE = (2+2)/2 = 2

g_B = ____________________.

c) The gate from part a) is used in the following circuit. Determine the gate sizes, X, Y, Z, that minimize the delay in the path below. If you're not confident about your answer to part c, assume the logical efforts of the new gate is 3. Gates sized as '1' have the equivalent driving resistance and input capacitance equal to a unit-sized inverter.



$LE = \frac{3}{2}$ $FO = \frac{Z}{Y}$

$B = 1$

$LE = 2$

$B = 3$

$FO = \frac{Y}{X}$

$PE = \pi LE \times \pi B \times FO$

$= 9 \times \frac{3}{2} \times 2 \times 3 = 81$

$\therefore SE = \sqrt[4]{81} = 3$

X: $1 \cdot x \cdot 1 = 3,\ x = 3$

Y: $2 \cdot 3 \cdot \frac{Y}{X} = 3,\ Y = \frac{3}{2}$

Z: $\frac{3}{2} \cdot 1 \cdot \frac{Z}{Y} = 3,\ \frac{3}{2} \cdot 1 \cdot Z \cdot \frac{2}{3} = 3 \quad Z = 3$

Verify: $1 \cdot 1 \cdot \frac{9}{Z} = 3$ ✓

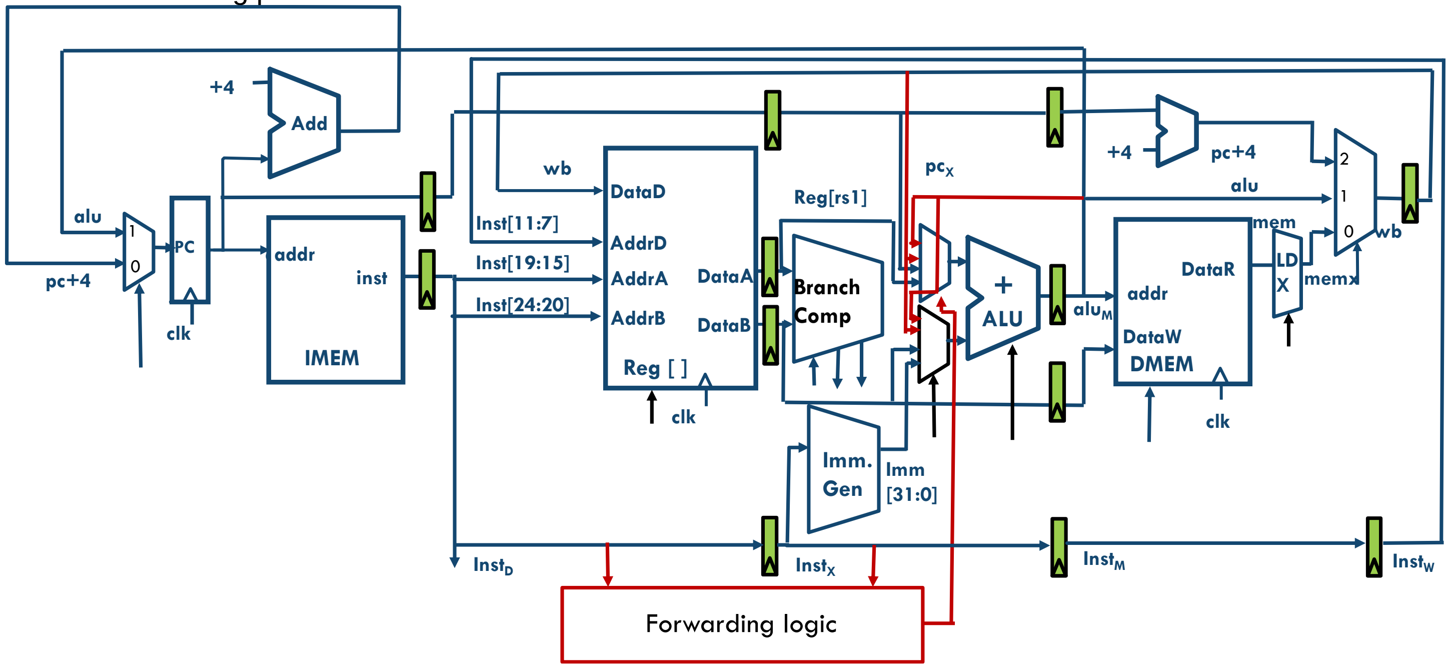
X = 3.

Y = $\frac{3}{2}$.

Z = 3.

2) *RISC-V Pipelining and Hazards…* (24 points, 26 minutes)

Consider the 5-stage pipeline presented in lecture with combinational-read IMEM and DMEM and the forwarding paths as drawn.



Fill the pipeline diagrams for the following assembly program. Indicate when NOPs are injected into the pipeline by writing 'NOP'. Assume you can write to and read from the same address in the register file (Reg [ ]) in the same cycle. Consider two cases:

Case 1: The forwarding paths in the diagram are unused and all hazard resolution is accomplished with stalling.
Case 2: The forwarding mux in the diagram is driven correctly by the forwarding logic.

```
1           addi x1, x0, 0xFF
2           andi x2, x1, 0xF
3           bge x1, x2, label
4           xori x2, x2, 1
5           ori x3, x2, x1
6    label: lw x5, 0(x6)
7           sw x5, 4(x6)
```

Please write down instruction names in the pipeline diagram until the last instruction has written back to the regfile.

	Case 1					Case 2				
Cycle	F	D	X	M	WB	F	D	X	M	WB
1	addi					addi				
2	andi	addi				andi	addi			
3	bge	andi	addi			bge	andi	addi		
4	bge	andi	nop	addi		xori	bge	andi	addi	
5	bge	andi	nop	nop	addi	xori	bge	nop	andi	addi
6	xori	bge	andi	nop	nop	xori	bge	nop	nop	andi
7	xori	bge	nop	andi	nop	ori	xori	bge	nop	nop
8	xori	bge	nop	nop	andi	lw	ori	xori	bge	nop
9	ori	xori	bge	nop	nop	lw	lw (kill)	ori (kill)	xori (kill)	bge
10	lw	ori	xori	bge	nop	sw	lw	nop	nop	nop
11	lw	lw (kill)	ori (kill)	xori (kill)	bge		sw	lw	nop	nop
12	sw	lw	nop	nop	nop		sw	nop	lw	nop
13		sw	lw	nop	nop		sw	nop	nop	lw
14		sw	nop	lw	nop			sw	nop	nop
15		sw	nop	nop	lw				sw	nop
16			sw	nop	nop					sw
17				sw	nop					
18					sw					

18 points
9 points per case
Case 1:
We will accept 3 NOP injections for the bge instruction, instead of killing already fetched instructions
Instructions must stall in the D stage where their rs1/rs2 values can be read and decoded

NOPs cannot be injected in the F stage itself before the dependent instruction has been fetched

- +1 andi stalls at D stage
- +1 andi moves forward once addi in WB stage
- +1 bge stalls at D stage
- +1 bge stalls until andi is in WB stage
- +1 3 NOPs are injected after bge or PC+4 instructions are fetched (exactly 3)
- +1 fetched xori/ori/lw are killed (or never fetched)
- +1 lw moves through without stalling
- +1 sw stalls at D stage
- +1 sw moves through pipeline once lw is at WB stage
- -3 for any premature (predictive) NOP injection (need to have instruction in pipeline before injecting NOPs if it has any dependencies)
- -1 for NOP injection from F stage (can only inject NOPs once instruction has been decoded and is in D stage)

Case 2:
bge needs to wait at the D stage until andi is in WB since there's no forwarding to the branch comp unit
sw needs to wait at the D stage until lw is in WB since there's no rs2 forwarding path

- +5 addi -> andi forwarding works (no NOPs injected)
    - +2 addi -> andi forwarding resolved with 1 NOP injected (using WB forwarding path)
- +2 andi -> bge forwarding path doesn't exist, must inject NOPs to resolve hazard
- +2 lw -> sw forwarding path doesn't exist, must inject NOPs to resolve hazard
- No penalties for predictive NOP injection, wrong number of NOPs, or other stuff covered in case 1. Only looked at forwarding paths used in the pipeline diagram.

b) Answer true or false for the following statements, about the 5-stage pipeline in this problem.

i) A mispredicted branch instruction causes 2 instructions to be killed

○ True [✓ False]

ii) Jumps (**`jal`**, **`jalr`**) cause 3 NOPs to be injected into the pipeline

[✓ True] ○ False

iii) Using forwarding paths could increase CPI (clocks per instruction) over the stalling-only baseline

○ True [✓ False]

iv) Immediate generation, in general, can proceed in parallel with register file reads

[✓ True] ○ False

v) Making the IMEM and the DMEM synchronous read would yield a 8-stage pipeline

○ True [✓ False]

vi) If the immediate generation was moved to the F stage, and you had an additional adder, you can make **`jal`** inject no NOPs

[✓ True] ○ False

3) *Energy and Performance* **(12 points, 12 minutes)**
We would like to examine some properties of a single-cycle RISC-V datapath. The datapath presents a total load capacitance of 5pF to the supply, operates at 500 MHz, and has an activity factor of $\alpha_{\{0\rightarrow 1\}}=0.1$. V_{DD} = 1V. The CPI is 1. You may find the following two equations useful for this problem.

$$\text{CPI}=\frac{\text{Clocks}}{\text{Instruction}}$$

$$\frac{\text{Seconds}}{\text{Program}}=\frac{\dfrac{\dfrac{\text{Instructions}}{\text{Program}}*\text{Cycles}}{\text{Instruction}}*\text{Seconds}}{\text{Cycle}}$$

a) What is the dynamic power consumption?

$P = \alpha * c * V^2 * f = (0.1) * (5\ \text{pF}) * (1\ \text{V})^2 * (500\ \text{MHz})$
$P = 250\ \mu W$

b) What is the average energy per instruction?

$E_{inst} = P * t_{inst} = P * CPI * (1/f)$
$E_{inst} = (250\ \mu W)*(1)*(1/500MHz)$
$E_{inst} = 1 * 10^{-13}\ J$

c) How much energy is drawn from the supply to run a program with 1000 executed instructions?

$E_{1000} = E_{inst} * 1000$ inst
$E_{1000} = 5 * 10^{-10}\ J{\cdot}s$

d) What is the energy-delay product for the case in c) ?

$EDP = E_{1000} * (1000\ \text{inst}) * (1\ \text{CPI}) * (1/f)$
$EDP = 1 * 10^{-15}\ J{\cdot}s$

e) You are able to pipeline this design so a 5-stage pipeline operates at 1.5GHz, at the same supply V_{DD} = 1V, has the same activity factor, and has a CPI of 3. The load capacitance has increased by 50% compared to the non-pipelined baseline. Calculate the energy-delay product when running the same program as in c)

We can walk through the same process as above

$P_{dyn\_5\text{-}stage} = \alpha * c * V^2 * f = 1.125\ mW$
$E_{inst\_5\text{-}stage} = P * t_{inst} = P * CPI * (1/f) = 2.25 * 10^{-12}\ J/instr$
$E_{1000\_5\text{-}stage} = E_{inst} * 1000$ inst
$EDP_{5\text{-}stage} = E_{1000\_5\text{-}stage} * 1000\ \text{inst} * CPI * 1/f$
$EDP_{5\text{-}stage} = 4.5 * 10 * 10^{-15}\ J{\cdot}s$

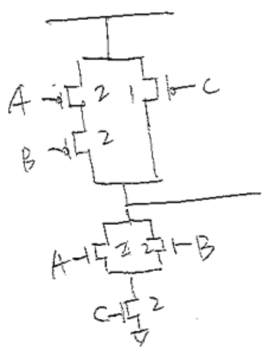
4) *Delays and Adders* (18 points, 22 minutes)

You are designing a datapath in a brand-new CMOS FinFET technology, where NMOS and PMOS devices have equal strength. In particular, it has only four CMOS gates: 2- input NAND, 2-input NOR, 2-input XOR, and an OAI21 gate ($Y = \overline{(A+B)C}$). Gate capacitance equals drain capacitance per unit area ($\gamma = 1$), and the four gates come in with only one size each, i.e., you do not need to size gates in this problem.

a) If both the 2-input NAND gate and the 2-input NOR gate have a delay dependence on a fanout, f, given as $t_{NAND2} = t_{NOR2} = 2ns(4 + 3f)$, what is the delay dependence on the fanout of the input ~~C~~ B of the OAI21 gate (~~C~~ B is in the critical path)?

$p = \frac{7}{2}$

$g = 2$

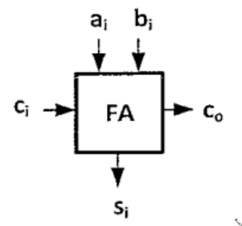


$p = \frac{3+2+2}{2} = \frac{7}{2}$

$g = \frac{2+2}{2} = 2$

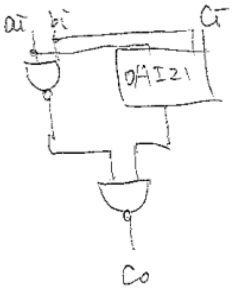
t_{AOI21} = 2ns(7+4f).

b) Draw a fast full-adder using the four types of gates and calculate the C_i->Sum and C_i->C_out delay. Note that using an OAI21 gate could potentially simplify the logic for C_out. Assume the capacitance driven by the sum and the carry bits is equal to the capacitance of inputs a_i, b_i and t_{XOR2} = 2ns (8 + 8f).

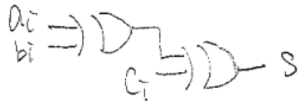


$S = a_i \oplus b_i \oplus C_{out}$

$C_o = \overline{\overline{a_i b_i} + \overline{(a_i+b_i)C_i}} = \overline{(\overline{a_i b_i}) \cdot \overline{(a_i+b_i)c_i}}$



$\therefore t_{FA, Ci \to Co} = 2ns(7+4f+4+3f)$
$= 2ns(11+7f)$

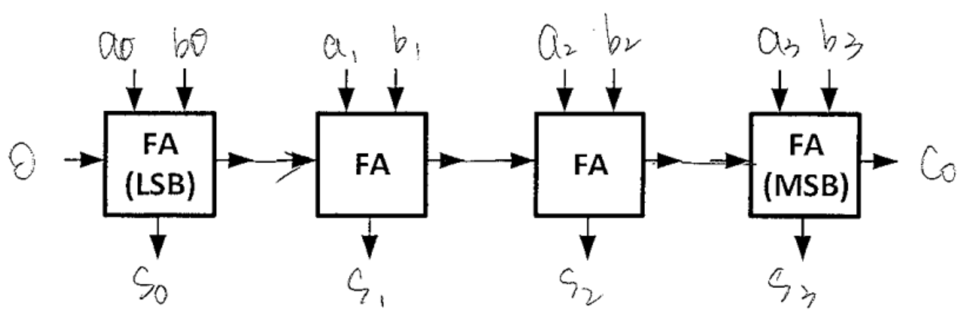


$\therefore t_{FA, ci \to S} = 2ns(8+8f)$

$t_{FA,Ci \to Co}$ = 2ns(11+7f).

$t_{FA,Ci \to S}$ = 2ns(8+8f).

c) Complete the drawing and label all inputs and outputs for an 4-bit ripple-carry adder in figure below by using full-adder (FA) cells from part b). Inputs are a[3:0] and b[3:0] and there is no carry-in to the least-significant bit.



d) Find the critical path delay for the circuit in part c), if the capacitance driven by the sum and the carry bits is equal to the input a_i, b_i capacitance. If you are not confident in your answer in part b), you can express the delay in terms of $t_{FA,Ci\to Co}$, and $t_{FA,Ci\to S}$

$t_{delay} = 3t_{FA,Ci\to Co} + t_{FA,Ci\to S}$

Critical path = ______________________.