COMS W1004 MW 4:10-5:25 PM
Columbia University Spring 2014

Homework 3: Problem Set
March 3, 2014

Your CUNIX ID: ami2119

Your Last Name: Iwamizu

Your First Name: Akiko

Circle the range that includes your UNI:

Group 1	(aa3473-am4051)	Group 8	(kea2134-lvt2107)
[✓ **Group 2**]	[✓ **(ama2231-av2425)**]	Group 9	(lz2371-mry2109)
Group 3	(ay2289-cme2133)	Group 10	(msv2121-pc2627)
Group 4	(cmh2194-emh2213)	Group 11	(pfa2103-sc3719)
Group 5	(emm2224-hv2169)	Group 12	(sch2148-tat2133)
Group 6	(hwk2106-jk3667)	Group 13	(tb2498-zn2116)
Group 7	(jl4161-kdj2109)		

Do the following exercises in Schneider and Gersting (6th Edition):
Chapter 4: 1 (6 points), 11 (6 points), 19 (12 points), 25 (16 points), 26 (16 points)
Chapter 5: 2 (8 points), 18 (6 points), 19 (6 points), 21 (16 points)
Challenge Work: 1 (8 points)

4.1) Find the decimals of the following:

a. 133 (base 4)

$(133)_4 = 1(4^2) + 3(4^1) + 3(4^0) = \mathbf{31}$

b. 367 (base 8)

$(367)_8 = 3(8^2) + 6(8^1) + 7(8^0) = \mathbf{247}$

c. 1BA (base 16) (Note: A represents 10 and B represents 11)

$(1BA)_{16} = 1(16^2) + 11(16^1) + 10(16^0) = \mathbf{442}$

4.11) Find the decimal value of the following (based on the method of 4.10):

a. 0111000000000111 → 0 | 111000000 | 0 | 00111 → $(+.111 \times 2^{+7})$ = **<u>112</u>**

b. 1010001000100001 → 1 | 010001000 | 1 | 00001 → $(-.010001 \times 2^{-1})$ = **<u>0.1328</u>**

4.19) a = 1, b = 2, and c = 2. What is the value of the Boolean expressions?

a. $(a > 1)$ OR $(b = c)$ → F OR T → **True**

b. $[(a+b) > c]$ AND $(b \leq c)$ → T AND T → **True**

c. NOT $(a = 1)$ → NOT(T) → **False**

d. NOT $[(a = b)$ OR $(b = c)]$ → NOT[T] → **False**

e. $(a = 1)$ AND $(b = 1)$ AND $(c = 2)$ → T AND F AND T → **False**

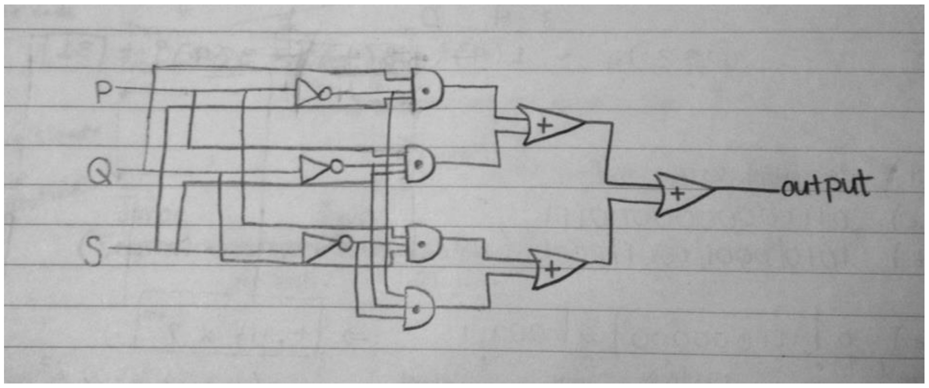
4.25) Design an odd-parity circuit

P	Q	R	Output
1	1	1	0
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	1

Boolean expression:

Output = $\mathbf{[(P \wedge Q \wedge \sim R) \text{ OR } (P \wedge \sim Q \wedge R) \text{ OR } (\sim P \wedge Q \wedge R) \text{ OR } (\sim P \wedge \sim Q \wedge \sim R)]}$

where " ~ "= NOT , " ^ " = AND



4.26) Design a 1-bit subtraction circuit

A	B	B_{in} (borrow in)	D (A-B-B_{in})	B_{out} (borrow out)	Process
0	0	0	0	0	0-0-0=0; no borrow
0	0	1	1	1	0-0-1=-1; borrow 2: 2-1=1
0	1	0	1	1	0-1-0=-1; borrow 2: 2-1=1
0	1	1	0	1	0-1-1=-2; borrow 2: 2-2=0
1	0	0	1	0	1-0-0=1; no borrow
1	0	1	0	0	1-0-1=0; no borrow
1	1	0	0	0	1-1-0=0; no borrow
1	1	1	1	1	1-1-1=-1; borrow 2: 2-1=1

Boolean expressions:

D = (~A ^ ~B ^ B_{in}) OR (~A ^ B ^ ~B_{in}) OR (A ^ ~B ^ ~B_{in}) OR (A ^ B ^ B_{in})

Which can be simplified by using the XOR rules listed below.

$$D = \big[\mathrm{Bin} \wedge [(\sim A \wedge \sim B) \vee (A \wedge B)]\big] \vee \ [\mathrm{Bin} \wedge [(\sim A \wedge B) \vee (A \wedge \sim B)]$$

$$D = [\mathrm{Bin} \wedge \sim(A \oplus B)] \vee [\sim \mathrm{Bin} \wedge (A \oplus B)]$$

$$\underline{\mathbf{D = Bin \oplus (A \oplus B)}}$$

1st line: I used (*) on the left side of the OR, and (**) on the right side of the OR
2nd line: I used (**) to get line 3.

B_{out} = (~A ^ ~B ^ B_{in}) OR (~A ^ B ^ ~B_{in}) OR (~A ^ B ^ B_{in}) OR (A ^ B ^ B_{in})

Which can be simplified by using the XOR rules listed below.

$$\mathrm{Bout} = \big[\mathrm{Bin} \wedge [(\sim A \wedge \sim B) \vee (A \wedge B)]\big] \vee \ [\sim A \wedge [(B \wedge \sim \mathrm{Bin}) \vee (B \wedge \mathrm{Bin})]$$

$$\mathrm{Bout} = [\mathrm{Bin} \wedge \sim(A \oplus B)] \vee [\sim A \wedge [(B \wedge B) \vee (\sim \mathrm{Bin} \wedge \mathrm{Bin})]$$

$$\underline{\mathbf{Bout = [Bin \wedge \sim(A \oplus B)] \vee [\sim A \wedge (B)]}}$$

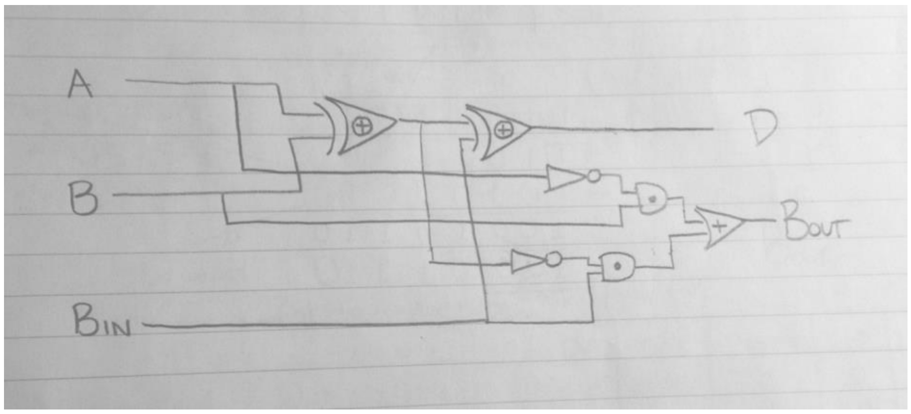
1st line: I used (*) on the left side of the OR, and complementary on the right side of the OR
2nd line: I used idempotency to get line 3.

Where " ~ "= NOT , " ^ " = AND

XOR Rules:

$p\oplus q = \sim[(p\wedge q)\vee(\sim p \wedge \sim q)]$ (*)

$= (p \wedge \sim q) \vee (\sim p \wedge q)$ (**)



5.2) At a minimum, how many bits are needed in the MAR with each of the following memory sizes?

a. 1 million bytes → **20 bits** (because 2^{20} = 1,048,576)
b. 10 million bytes → **24 bits** (because 2^{24} = 16,777,216)
c. 100 million bytes → **27 bit**s (because 2^{27} = 134, 217, 728)
d. 1 billion bytes → **30 bits** (because 2^{30} = 1,073,741,824)

5.18) Assume that we are receiving a message across a network using a modem with a rate of 56,000 bits/second. Furthermore, assume that we are working on a workstation with an instruction rate of 500 MIPS. How many instructions can the processor execute between the receipt of each individual bit of the message?

$$56{,}000\frac{bits}{second}\ x\ \frac{1}{500{,}000{,}000}\frac{second}{instructions} = 1.12\ x\ 10^{-4}\frac{bits}{instructions}$$

I want $\frac{instructions}{bits}$, so I divide by 1 and get: $\frac{1}{1.12\ x\ 10^{-4}} = \mathbf{8,928}\ \frac{\boldsymbol{instructions}}{\boldsymbol{bits}}$

5.19)

a. What is the maximum number of distinct operation codes that can be recognized and executed by the processor on this machine?

2^6 = **<u>64 op codes</u>**

b. What is the maximum memory size on this machine?

2^{18} = **<u>262,144 memory cells</u>**

c. How many bytes are required for each operation?

6+18+18 = 42 bits → 42 bits x $\frac{1\,byte}{8\,bits}$ = 5.25 bytes (must round) = **<u>6 bytes</u>**

5.21) Translate the following algorithmic operations into their machine language equivalents.

a. Set v to the value of x-y+z

Address	Contents	Comments
	…	
50	LOAD 202	Put the value of x into register R.
51	SUBTRACT 203	Subtract y to register R. It now holds x-y.
52	ADD 204	Add z to register R. It now holds x-y+z.
53	STORE 200	Store the contents of register R into v.
54	…	

b. Set v to the value (w+x)-(y+z)

Address	Contents	Comments
	…	
50	LOAD 203	Put the value of y into register R.
51	ADD 204	Add z to register R. It now holds y+z.
52	STORE 203	Store the contents of register R into y.
53	LOAD 201	Put the value of w into register R.
54	ADD 202	Add x to register R. It now holds w+x.
55	SUBTRACT 203	Subtract y to register R. It now holds w+x-(y+z).
56	STORE 200	Store the contents of register R into v.
57	…	

c. If, else loop

Address	Contents	Comments
	…	
50	COMPARE 200,201	Compare v and w and set condition codes.
51	JUMPGE 54	Go to location 54 if v≥w.
52	MOVE 202,204	Get here if v<w, so move x into z.
53	JUMP 55	And skip the next instruction.
54	MOVE 202,203	Move x into y.
55	…	

d. While do loop

Address	Contents	Comments
	…	
50	COMPARE 203,204	Compare y and z and set condition codes.
51	JUMPGE 56	Jump to location 56 if $y \geq z$.
52	ADD 203,201, 203	Add y and w, store this into y. → y+w
53	ADD 203,204, 203	Add y and z, store this into y. → y+w+z
54	ADD 204,200, 204	Add z and v, store this into z. →z+v
55	JUMP 50	Repeat until loop ends.
56	…	

5C1) Design a parallel algorithm that utilizes these additional resources to speed up the solution to the previous computation. Exactly how much faster would your parallel summation algorithm execute than the sequential one? Did you need all 100 processors? Could you have used more than 100?

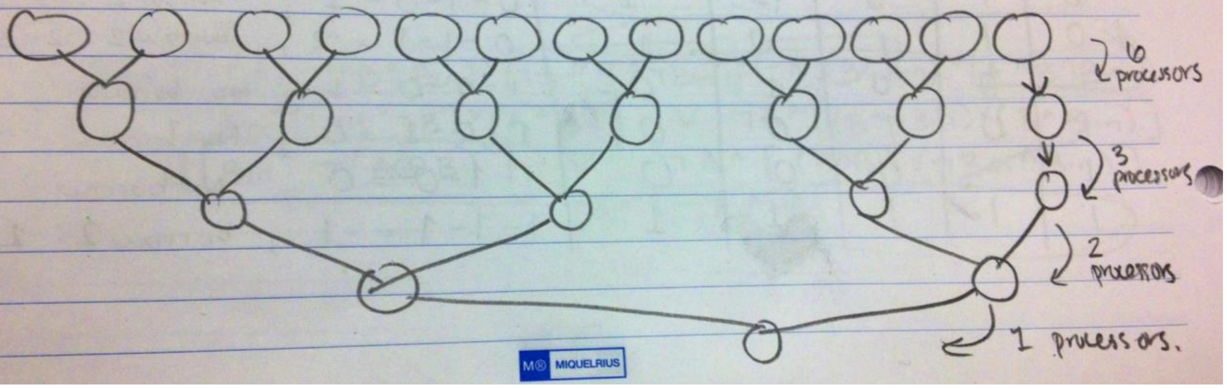
My parallel algorithm would allow 50 processors to add up distinct pairs from a1 to a100, then another 25 processors for the next round of additions, then 12 processors for the next round of addition (there is 1 number that did not have a pair, so there are 13 numbers left to be added), then 6 (there is 1 number that did not have a pair, so there are 7 numbers left to be added), then 3 processors for the next round of addition, then 2 processors for the next round, and then 1 processor for the final round of addition. Thus, there are only 7 steps and this would only take 7 units of time as opposed to 100 units of time in the original algorithm.

At the most you can use 99 processors, but not more than that using this algorithm in which you assume that each processor can only 1 addition throughout the entire algorithm. Under the assumption that a processor can be "recycled" and do more than multiple additions throughout the entire algorithm, then you can use 50 processors (the processors that calculated the first round of addition). Hence, you do not need all 100 processors.

For this algorithm, you could not have used more than 100 processors because there are only 100 processes in the original algorithm with a loop that would take about 100 units of time.

An example of this algorithm's last 4 steps looks like the image below:

· where "\ /" = 1 processor and symbolizes an addition:



6 processors
3 processors
2 processors
1 processors.