King Fahd University of Petroleum and Minerals
College of Computer Science and Engineering
Computer Engineering Department

COE 202: Digital Logic Design (3-0-3)
Term 111 (Fall 2011)
Major Exam 2
Thursday December 8, 2011

Time: 120 minutes, Total Pages: 10

Name: Key **ID:** _____ **Section:** _____

Notes:

- Do not open the exam book until instructed
- **Calculators are not allowed** (*basic, advanced, cell phones, etc.*)
- Answer all questions
- All steps must be shown
- Any assumptions made must be clearly stated

Question	Maximum Points	Your Points
1	16	
2	20	
3	20	
4	34	
Total	90	

Question 1. **(points)**

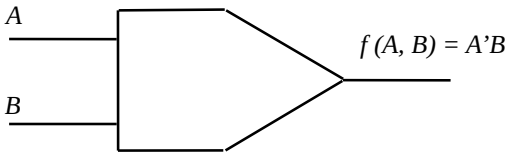
I.

a. What are the conditions for a gate to be universal? [**1 Point**]

Can implement Any Boolean expression without need for any other type gates. This is equivalent to ability to perform any of the fo;;owing:

1. {AND, NOT}
2. {OR, NOT},
3. {AND, OR, NOT}

Show that a two input gate performing the function $f(A, B) = A'B$ is a universal gate. [**3 Points**]



1. Can Implment {NOT} → B=1, $f = A'$
2. Can Implment {AND} → Use two such gates; one acts as inverter generating A' and a 2nd with it's a-input = A', B= B and, $f = AB$

II. *Let* $F(x, y, z) + G(x, y, z) = H(x, y, z)$, *where F, G* and *H* are Boolean functions of three variables *x*, *y*, and *z*. Given that

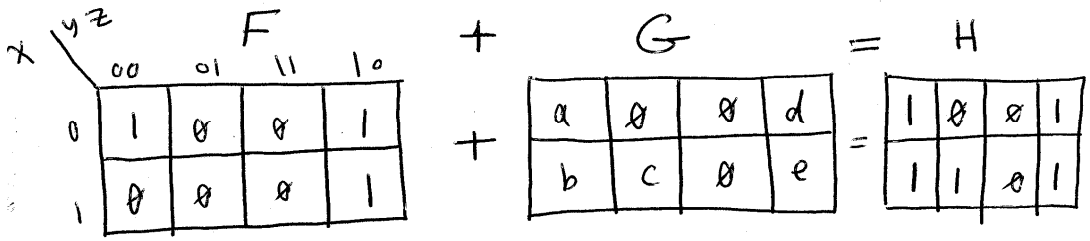
$$F(x, y, z) = x`z` + yz` \qquad \underline{\text{and}} \qquad H(x, y, z) = z` + xy`$$

a- Find *a* possible function $G(x, y, z)$ that will satisfy the above relation. [**3 Points**]

b- Is the solution unique? If not, what is the number of possible *G(x, y, z)* functions that can satisfy the above relation? [**2 Points**]

(*Hint: Draw the K-maps of the functions above*)

(*Hint: Draw the K-maps of the functions above*)



• Since $F + G = H$, then

1) IF $H = \emptyset$ then $F = \emptyset$ & $G = \emptyset$

2) IF $H = 1$ and $F = \emptyset$ then

G must be 1 $\Rightarrow b = c = 1$

3) IF $H = 1$ and $F = 1$ Then

G can be either ∅, or 1

i.e $G = X$ = Don't Care

$\Rightarrow a = b = e = X$

∴ G =

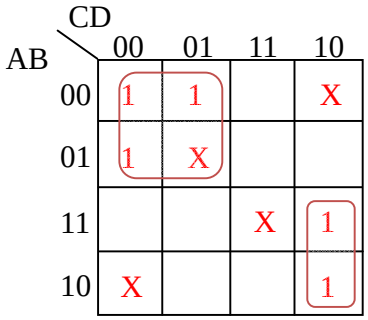
x \ yz	00	01	11	10
	X			X
	1	1		X

$\Rightarrow$ 3-don't Cares means 8-possible expressions for G will satisfy the above relation.

III. Using the K-Map method, give a simplified SOP expression for the Boolean function F(A, B, C, D) =∑(0, 1, 4, 10, 14) subject to don't care conditions d(A,B, C, D) = ∑(2, 5, 8, 15). [**8 Points**]

<SHOW your WORK>

F(*A, B, C, D*) = A'C' + ACD'



Question 2. **(20 Points)**

Given a **4-bit** *unsigned* number X $(x_3\, x_2\, x_1\, x_0)$, the function $F(X)$ is defined as:

$$F(X) = \begin{cases} X - 1, & X \text{ is odd} \\ \frac{X}{2}, & X \text{ is even} \end{cases}$$

You are to design a digital circuit that takes X as an input and produces $Z = F(X)$ as output;

(a) How many bits are required to represent the output Z? [2 points]

The maximum unsigned # representable in 4 bits is 15 which is an odd #

$\Rightarrow F(15) = 14$ which is also representable in 4 bits

$\Rightarrow$ # of bits for Z is 4

(b) Derive the truth table for the output function Z. [5 points]

X_3	X_2	X_1	X_0	Z_3	Z_2	Z_1	Z_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	0	0
1	0	0	1	1	0	0	0
1	0	1	0	0	1	0	1
1	0	1	1	1	0	1	0
1	1	0	0	0	1	1	0
1	1	0	1	1	1	0	0
1	1	1	0	0	1	1	1
1	1	1	1	1	1	1	0

(c) If **Z** is to be implemented using only NAND gates, derive minimized expressions for the output bits of the circuit. [8 points]

Z_0

AB \ CD	00	01	11	10
00				1
01				1
11				1
10				1

$$Z_0 = A\bar{B}$$

CD \ AB	00	01	11	10
00			1	
01	1		1	1
11	1		1	1
10			1	

$$Z_1 = B\bar{D} + CB + CD$$

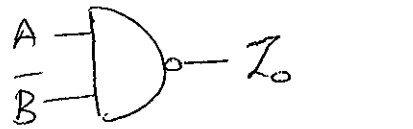
CD \ AB	00	01	11	10
00				
01			1	1
11			1	1
10				

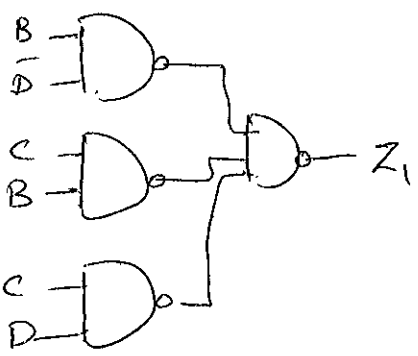
$$Z_3 = AD$$

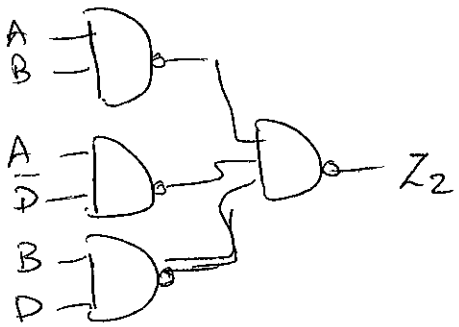
AB \ CD	00	01	11	10
00				
01		1	1	
11	1	1	1	1
10	1			1

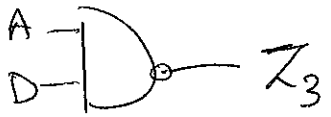
$$Z_2 = AB + A\bar{D} + BD$$

(d) Implement the output expressions obtained in (c) using **NAND** gates only [5 points]





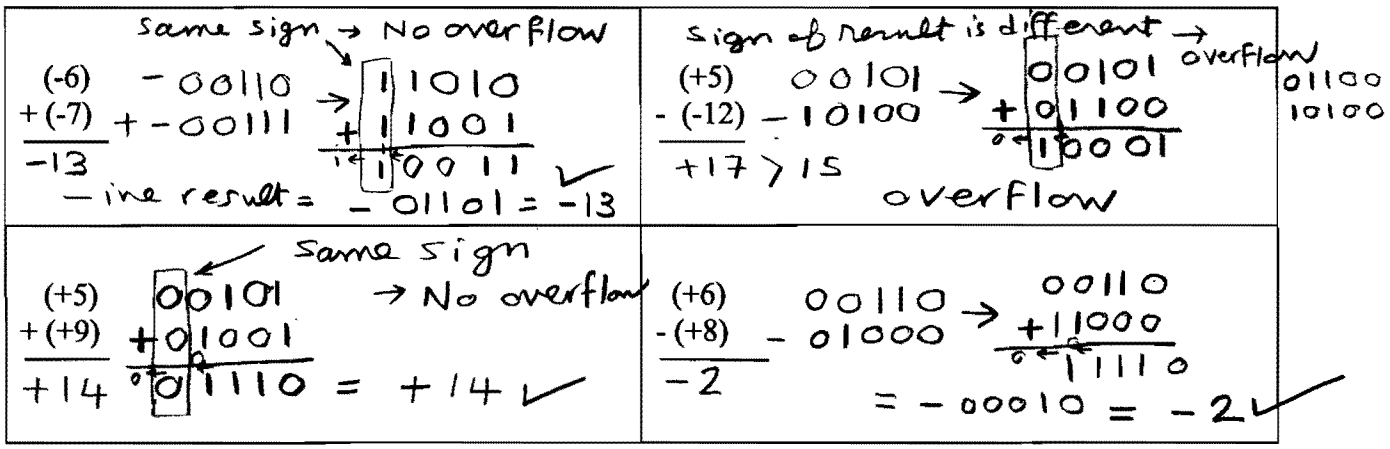




Question 3. **(20 Points)**

a. In each of the following problems, first represent the numbers in brackets in binary in the signed-2's complement notation **using 5 bits** then perform the indicated arithmetic operation using only binary addition. [8 points]

In each case check if you have obtained the correct results and indicate clearly if overflow occurred or not.



b. Interpret each of the following 5-bit binary numbers in the format indicated: [6 points]

Binary Number	Is equal to (in decimal)	When interpreted as:
11011	−11	Signed-magnitude
01101	+13	Signed-1's complement
10110	01010 −10	Signed-2's complement
11010	26	Unsigned

c. When doing signed-2's complement arithmetic in 6-bits: (Fill in the spaces **with signed decimal values**) [6 points]

- The range of numbers that can be represented extends from -32 to +31.

31 − 13 = 18

- The largest positive number that can be added to +13 without causing an overflow is +18.

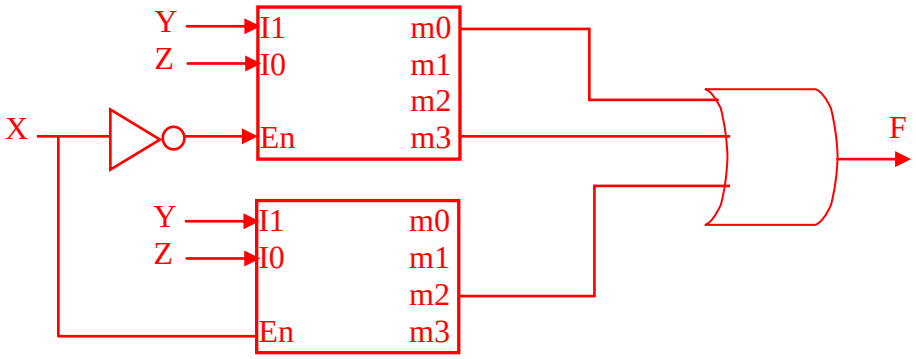
- Overflow may occur when (circle all that applies):
  - i. Adding a positive number to a negative number
  - ii. Subtracting a negative number from a negative number
  - iii. Subtracting a negative number from a positive number

Question 1. Given the function

F (X,Y,Z) = Π M (1,2,4,5,7)

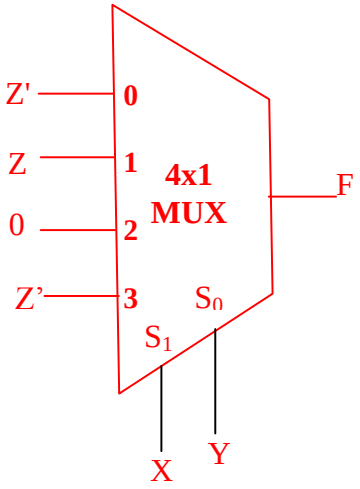
a. Implement F using 2 2-to-4 decoders and any other gates you need

F (X,Y,Z) = ∑ m (0,3,6)



b. Implement F using the 4-to-1 MUX shown below (see steps for obtaining the solution)

X	Y	Z	F	F
0	0	0	1	Z'
0	0	1	0	
0	1	0	0	Z
0	1	1	1	
1	0	0	0	0
1	0	1	0	
1	1	0	1	Z’
1	1	1	0	

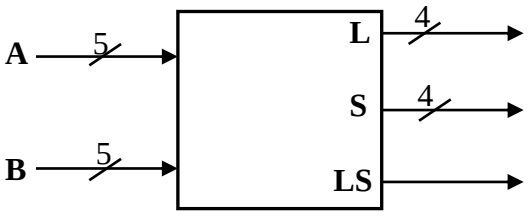


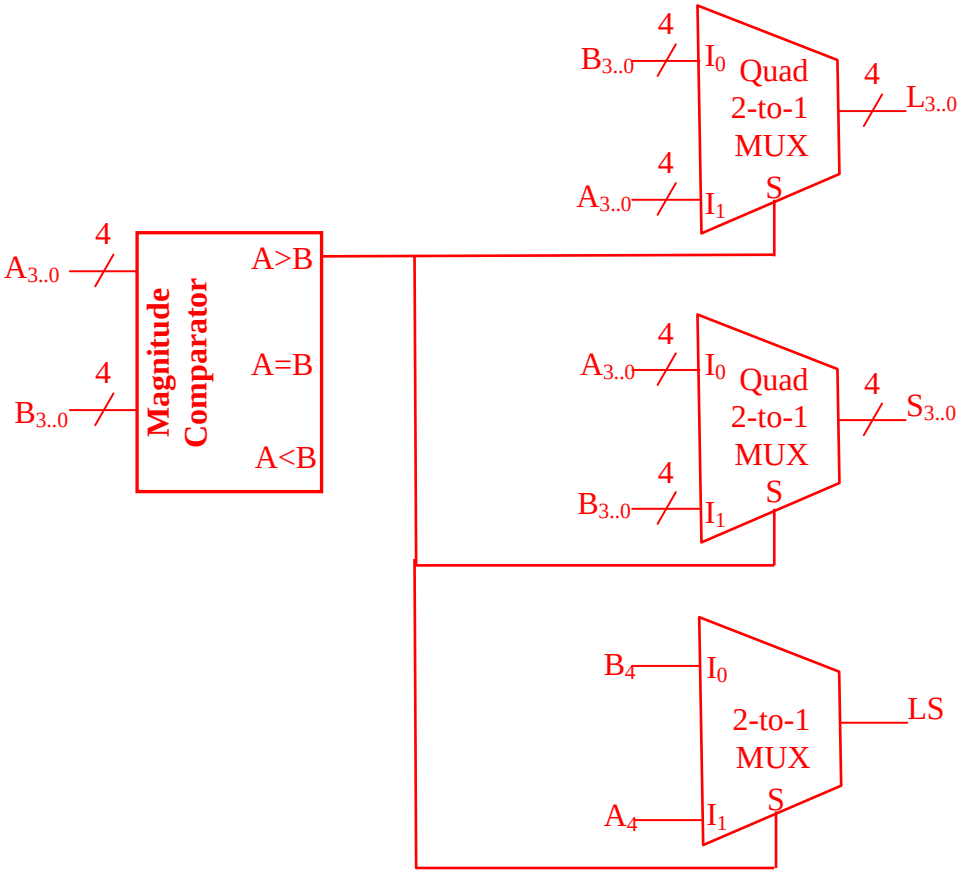
c. It is required to design a circuit that adds two **5-bit numbers**, A and B, that are represented in **signed-magnitude**. The result ***O***, also 5-bits, should also be represented in signed-magnitude notation. **Ignore overflow. You can Design this circuit in anyway you like or follow the suggested sequence below.**

(I) Using MSI parts, design a circuit that receives two 5-bit signed numbers A &B (represented in **signed-magnitude**) and produces three outputs **L**, **S** and **LS**, *where*:

- **L:** a 4-bit number which equals the larger *magnitude* of either A or B irrespective of their signs (e.g, A=-5 and B=+4 then L=5 but if A=-3 and B=+7 then L=7)
- **S:** a 4-bit number which equals the smaller *magnitude* of either A or B irrespective of their signs (e.g, A=-5 and B=+4 then S=4 but if A=-3 and B=+7 then S=3)
- **LS:** a single bit which equals the sign of larger magnitude number of either A or B (e.g, A=-5 and B=+4 then LS=1 or if A=-3 and B=+7 then LS=0).

Magnitude of A = $A_3A_2A_1A_0 = A_{3..0}$
same with B …
Sign of A is A_4 … same with B …





(II) Using MSI parts, design a Signed-Magnitude Adder that will take as input **L**, **S, LS** (produced by the circuit in I above), **A** and **B**, and generates **A+B**

Hint: to add two signed-magnitude numbers ***A*** and ***B*** we can do the following:

1. If sign(**A**) = Sign(**B**) → then magnitude(**O**) = magnitude(**A**) + magnitude(**B**), sign(**O**) = Sign(**A**)
2. else (i.e. sign(**A**) ≠ Sign(**B**)) → subtract the smaller number from the larger number magnitude(**O**) = max[magnitude(**A**), magnitude(**B**)] - min[magnitude(**A**), magnitude(**B**)], sign(**O**) = Sign of the number with the larger magnitude

Sign of output (O_4) is LS
Magnitude is either L+S (when signs are equal)
or L-S when signs are different
The Adder/Subtractor performs A+B if
the Add/Sub input is 1 and A-B otherwise