EE 209 Homework 2

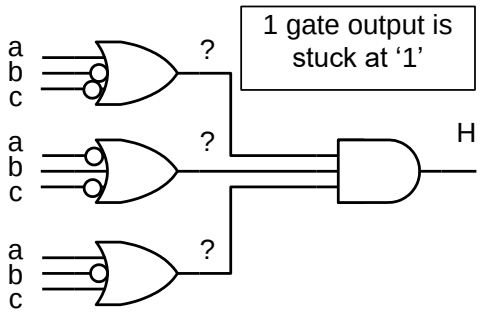
Name: ___Solutions_____
Due: **Score: _____**

Show work to get full credit. Remember, use on only one side of the paper and staple them together. Only use a calculator to CHECK your work, not to DO your work.

1. When the circuit below was fabricated on a chip, a problem occurred. One of the first level OR gates continually outputs a '1' (a.k.a. a "stuck-at-1" fault) no matter what the inputs are.

a. Propose and list a minimal number of input combinations that could be used to determine which gate had the fault (i.e. a,b,c = 1,1,0 and a,b,c = 1,1,1)? Describe what how you could use these values to determine the gate with the error?

b. If instead one of the OR gates was stuck at '0' would any set of input combinations be able to determine which gate had the error? Explain?



a.) If one of the OR gates is stuck at 1 we can figure out which one it is by plugging in combinations that should make one of the OR gates output '0' (while the other OR gates output 1) which should in turn force the AND gate to output 0. However, since an OR gate is stuck at '1' then whatever input combination doesn't force H=0 should identify the faulty gate.

Combination to make the top gate output '0': a,b,c = 0,1,1
Combination to make the middle gate output '0': a,b,c = 1,0,1
Combination to make the bottom gate output '0': a,b,c = 0,1,0

Whichever combination doesn't force H=0 identifies the faulty gate.

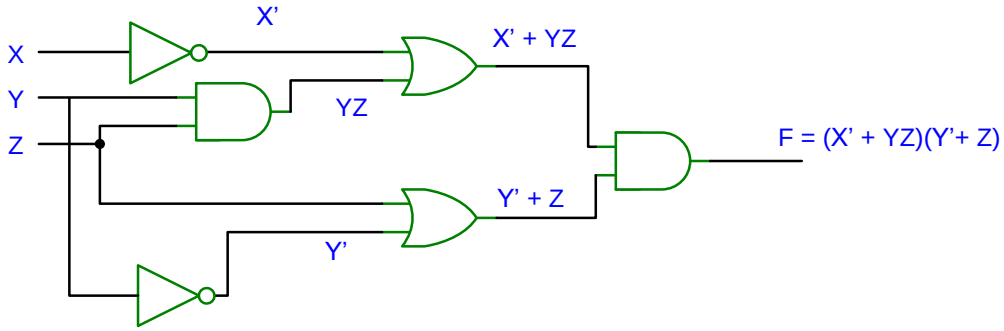
If an OR gate was stuck at '0', the AND gate would continuously be forced to output 0 (look at T2'). Thus, that stuck at '0' would mask any input combinations we tried to put into the other gates. Thus no combination can help us identify the stuck at fault.

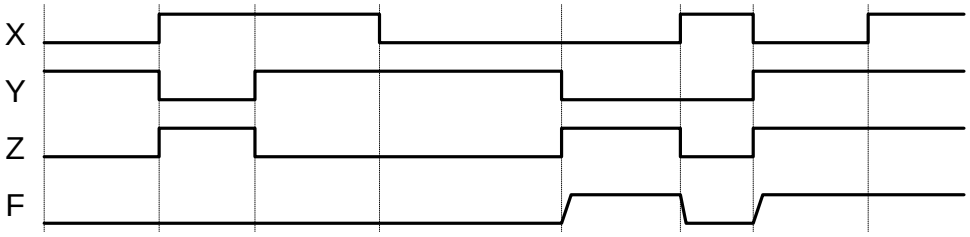
2.a G $= \Sigma_{ABCD}(2,3,6,13,15)$
$= A'B'CD' + A'B'CD + A'BCD' + ABC'D + ABCD$

$= \Pi_{ABCD}(0,1,4,5,7,8,9,10,11,12,14)$
$= (A + B + C + D) \bullet (A + B + C + D') \bullet (A + B' + C + D)$
$\bullet (A + B' + C + D') \bullet (A + B' + C' + D') \bullet (A' + B + C + D)$
$\bullet (A' + B + C + D') \bullet (A' + B + C' + D) \bullet (A' + B + C' + D')$
$\bullet (A' + B' + C + D) \bullet (A' + B' + C' + D)$

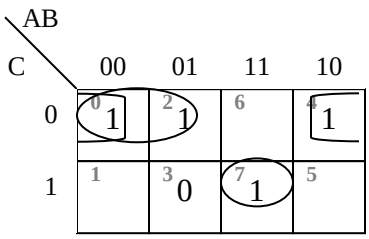
3. F = $\Sigma_{A[3:0]}(0,3,6,9,12,15)$ =
A3’A2’A1’A0’ + A3’A2’A1A0 + A3’A2A1A0’ + A3A2’A1’A0 + A3A2A1’A0’ + A3A2A1A0

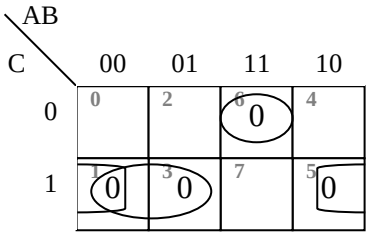
4.





5.a

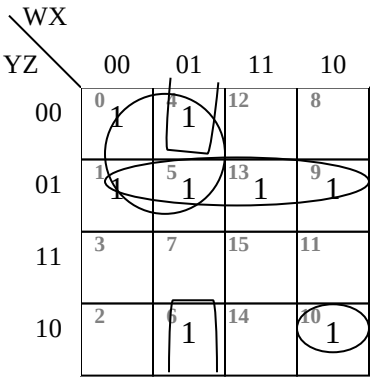




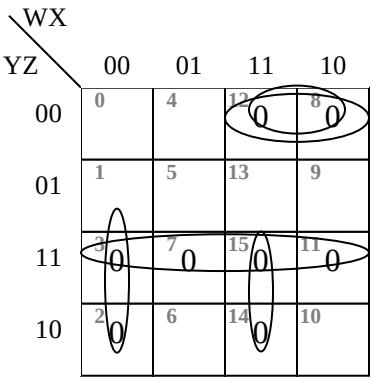
SOP = B’C’ + A’C’ + ABC

POS = (B+C’)(A+C’)(A’+B’+C)

5.b

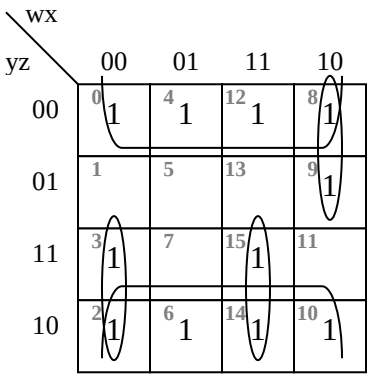


SOP = $\overline{Y}Z + \overline{W}\overline{Y} + \overline{W}X\overline{Z} + W\overline{X}Y\overline{Z}$

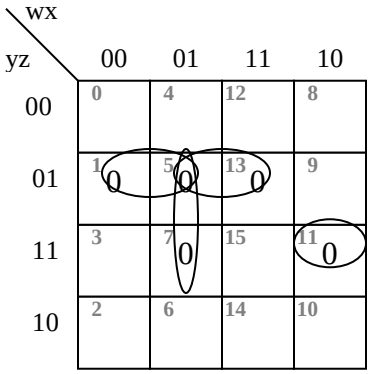


POS = $(\overline{Y} + \overline{Z})(W + X + \overline{Y})(\overline{W} + \overline{X} + \overline{Y})(\overline{W} + Y + Z)$

5.c



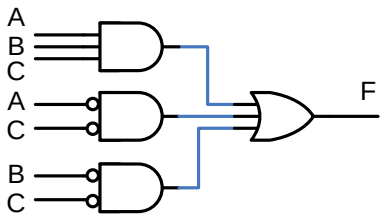
SOP = $\bar{z} + \bar{w}\bar{x}y + wxy + w\bar{x}\bar{y}$



POS = $(W + Y + \bar{Z})(\bar{X} + Y + \bar{Z})(W + \bar{X} + \bar{Z})(\bar{W} + X + \bar{Y} + \bar{Z})$

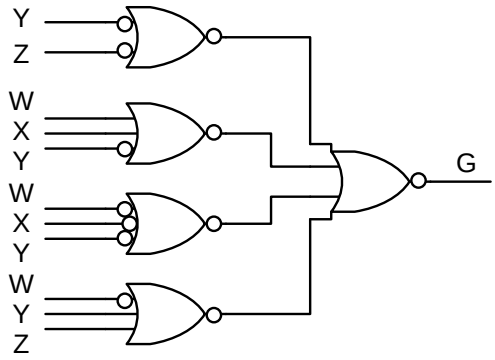
6.a (a) SOP = $\overline{A}B + \overline{B}\overline{C} + BC$ shown, but alternately: B'C' + A'C' + BC

- AND-OR implementation



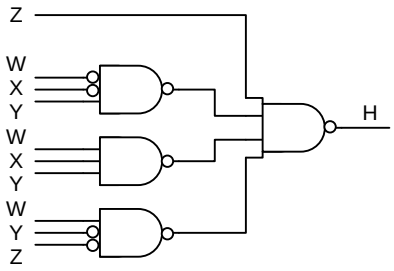
6b.

- NOR-NOR implementation



6c.

- NAND-NAND implementation



7.a

A 3-bit signed-magnitude number can represent a decimal from -3 to +3. Note that we regard "-0" as an illegal input. So we never use "100" as an input. When A is ranging from -3 to +3, Z = A + 2 can represent the decimal number from -1 to +5. We use 4-bit 2's complement number to represent it.

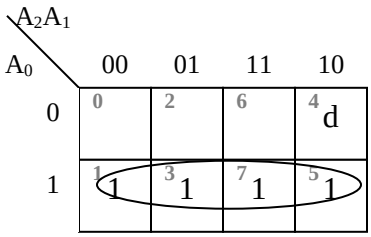
- The block diagram is below:

5.b Truth Table

A	A_2	A_1	A_0	Z_3	Z_2	Z_1	Z_0	Z
+0	0	0	0	0	0	1	0	+2
+1	0	0	1	0	0	1	1	+3
+2	0	1	0	0	1	0	0	+4
+3	0	1	1	0	1	0	1	+5
-0	1	0	0	d	d	d	d	d
-1	1	0	1	0	0	0	1	+1
-2	1	1	0	0	0	0	0	+0
-3	1	1	1	1	1	1	1	-1

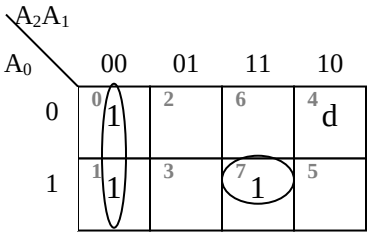
7.b

(1) K-Map of bit Z_0:



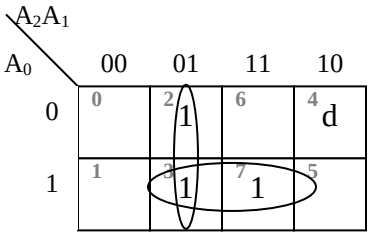
$Z_0 = A_0$

(2) K-Map of bit Z_1:



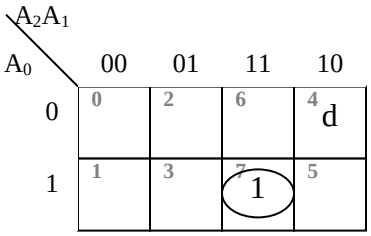
$Z_1 = \overline{A_2}\,\overline{A_1} + A_2 A_1 A_0$

(3) K-Map of bit Z_2:



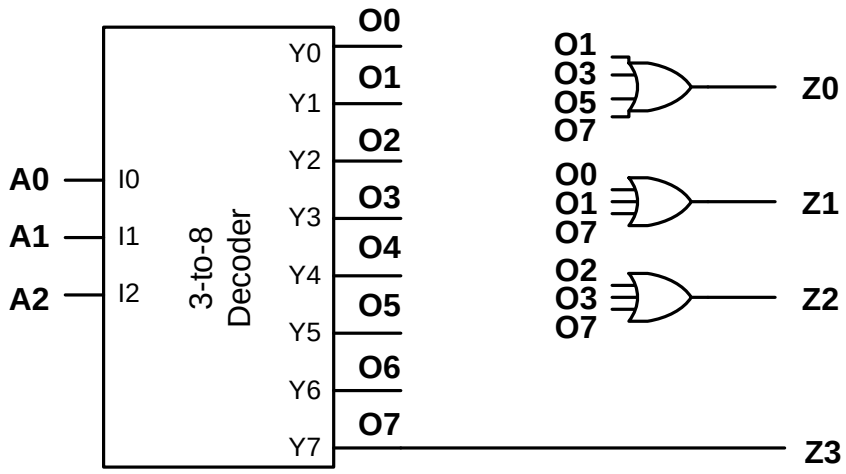
$Z_2 = \overline{A_2} A_1 + A_1 A_0$

(4) K-Map of bit Z_3:



$Z_3 = A_2 A_1 A_0$

7.c


A0
A1
A2
I0
I1
I2
3-to-8
Decoder
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
O0
O1
O2
O3
O4
O5
O6
O7
O1
O3
O5
O7
Z0
O0
O1
O7
Z1
O2
O3
O7
Z2
Z3