Solutions Provided by Audrey and Li

2/11/16

CSC 270

Homework 2

Problem 1

The top gate (gate A) with the AND gate and an inverter input (figure 1) is universal, but not the bottom gate (gate B) with an exclusive OR gate with an inverted input (figure 2). Table 1 shows the truth table for gate A, where the input b is assumed to be connected to the inverted input. As can be seen from figure 1, if we can make a NOT gate from gate A, we can easily make a NAND gate from gate A by inverting the inverted input, and also by inverting the output. Therefore, if we can make a NOT gate from gate A, we can prove that gate A is universal since NAND gates are universal. From table 1, we can see that if we supply a voltage to the un-inverted input (input a), we can make a NOT gate since when b is 0, the output is 1, while when b is 1, the output is 0. Since we have proven that a NOT gate can be made with gate A, NAND gates can be made from gate A, and hence gate A is universal.

Table 2 shows the truth table for gate B, where the input b is assumed to be connected to the inverted input. From table 2, we can see that there are two positive outputs out of the four possible outputs. Since there is no exclusivity in the output where we are able to differentiate or single out a particular a and b combination based on the output, we cannot make a universal gate out of gate B.

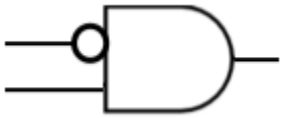

Figure 1. AND gate with an Inverted Input: Gate A

Figure 2. XOR gate with an Inverted Input: Gate B

Table 1. Truth Table for an AND gate with an Inverted Input (Gate A)

a	b	ab'
0	0	0
0	1	0
1	0	1
1	1	0

Table 2. Truth Table for an XOR gate with an Inverted Input (Gate B)

a	b	a⊕b'
0	0	1
0	1	0
1	0	0
1	1	1

Problem 2

Table 3 is a reference to figure out where the different minterms are in the Karnaugh map for 4 inputs.

Table 3. Karnaugh Map Numbering for Minterms

CD\AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

Table 4 shows the Karnaugh map for f(a, b, c, d) = Σ(0, 1, 2, 3, 12, 13, 14, 15). From table 4, f(a, b, c, d) can be simplified to f(a, b, c, d) = A'B' + AB = (A + B)' + AB.

Table 4. Karnaugh Map for f(a, b, c, d) = Σ(0, 1, 2, 3, 12, 13, 14, 15)

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

Table 5 shows the Karnaugh map for g(a, b, c, d) = Σ(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15). From table 5, g(a, b, c, d) can be simplified to g(a, b, c, d) = A' + B + C' + D' = (AB)' + (CD)'.

Table 5. Karnaugh Map for g(a, b, c, d) = Σ(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15)

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

h(a, b, c) = Π(0, 1, 2, 3) = Σ(4, 5, 6, 7). Table 6 shows the Karnaugh map for h(a, b, c) = Π(0, 1, 2, 3). From table 6, h(a, b, c) can be simplified to h(a, b, c) = A.

Table 6. Karnaugh Map for h(a, b, c) = Π(0, 1, 2, 3)

C\AB	00	01	11	10
0			1	1
1			1	1

k(a, b, c, d) = Π(0, 1, 7, 8) = Σ(2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15). Table 7 shows the Karnaugh map for k(a, b, c, d) = Π(0, 1, 7, 8). From table 7, k(a, b, c, d) can be simplified to k(a, b, c, d) = BC' + AD + AC + CD + B'C = B⊕C + A(D + C) + CD.

Table 7. Karnaugh Map for k(a, b, c, d) = Π(0, 1, 7, 8)

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

Problem 3

Table 8 shows the truth table for the majority voter, while table 9 shows the truth table for a 3-to-8 active-high output decoder. From table 8, the Boolean expression of the majority voter can be expressed as O(a, b, c) = Σ(3, 5, 6, 7). Figure 3 shows the implementation of the majority voter using the 3-to-8 active-high decoder shown in table 9.

Table 8. Truth Table for the Majority Voter

a	b	c	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 9. Truth Table for a 3-to-8 Active-High Output Decoder

A0	A1	A3	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
L	L	L	H	L	L	L	L	L	L	L
L	L	H	L	H	L	L	L	L	L	L
L	H	L	L	L	H	L	L	L	L	L
L	H	H	L	L	L	H	L	L	L	L
H	L	L	L	L	L	L	H	L	L	L
H	L	H	L	L	L	L	L	H	L	L
H	H	L	L	L	L	L	L	L	H	L
H	H	H	L	L	L	L	L	L	L	H

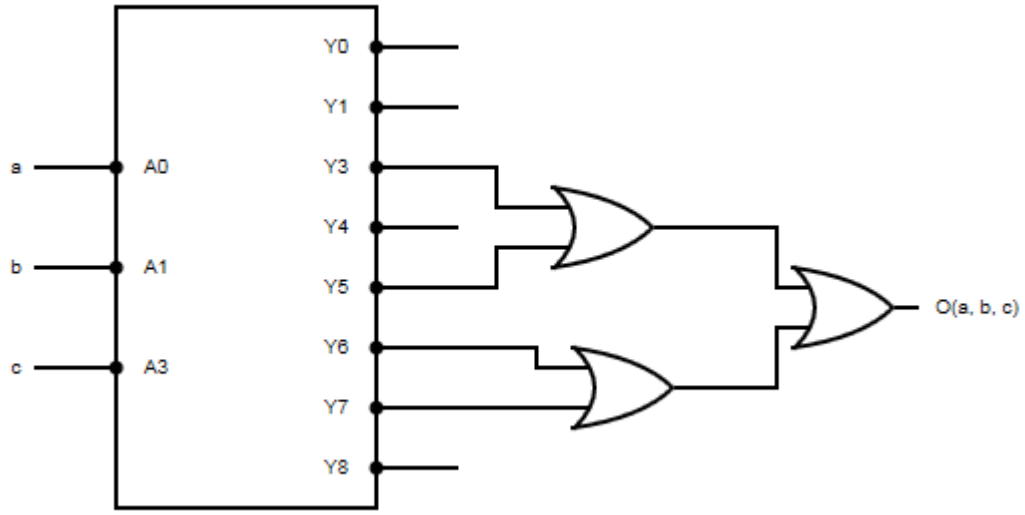

Figure 3. Circuit Diagram of a Majority Voter Implemented with a 3-to-8 Active-High Decoder

Table 10 shows the truth table for a 3-to-8 active-low, active-low enable decoder. Using this decoder and only NAND gates, a majority voter circuit can be implemented as shown in figure 4.

Table 10. Truth Table for a 3-to-8 Active-Low, Active-Low Enable, Output Decoder

A0	A1	A3	E	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	X	X	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H
L	H	L	L	H	H	L	H	H	H	H	H
L	H	H	L	H	H	H	L	H	H	H	H
H	L	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	L	H	H
H	H	L	L	H	H	H	H	H	H	L	H
H	H	H	L	H	H	H	H	H	H	H	L

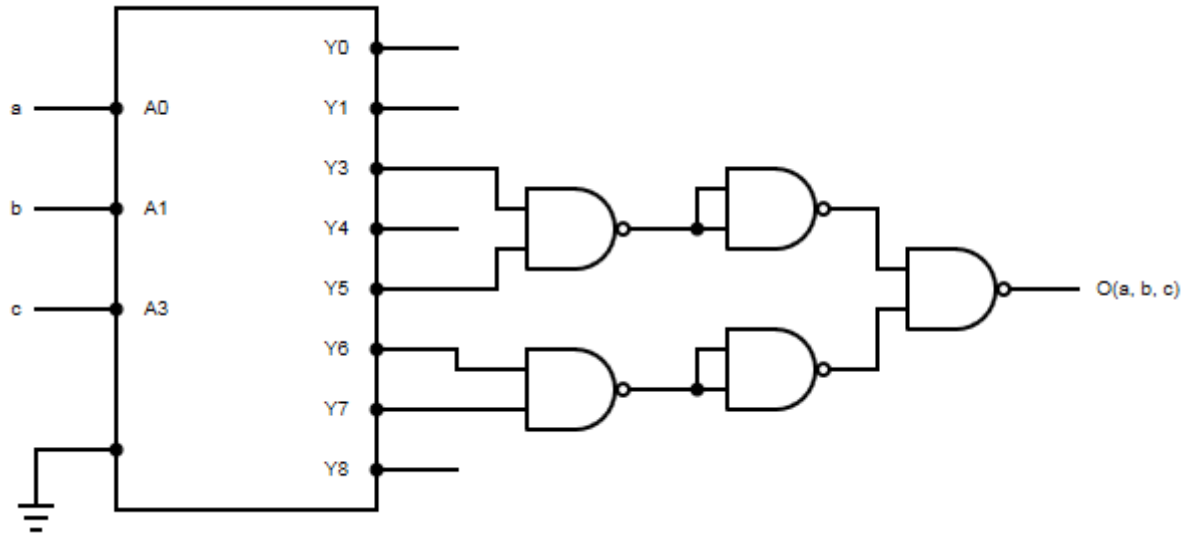

Figure 4. Circuit Diagram of a Majority Voter Implemented with a 3-to-8 Active-Low, Active-Low Enable Decoder and only NAND gates

```
# hw2.py
# Li Chai
#
# Simulates a 3-to-8 decoder with an active-high enable,
# and an active-high outputs.
#
import math
def f( e, a, b, c):
    L = [0,0,0,0,0,0,0,0]
    if e:
        decimal = c*(2**0) + b*(2**1) + a*(2**2)
        L[7-decimal] = 1


    return L

def main():
    print( "  E  A  B  C |  Y7  Y6  Y4  Y3  Y2  Y1  Y0 " )
    print( "-------------+------------------------------" )
    for e in [0, 1]:
        for a in [ 0, 1 ]:
            for b in [ 0, 1 ]:
                for c in [0,1]:
                    print( "%3d%3d%3d%3d  |%3d%3d%3d%3d%3d%3d%3d%3d" %
                        ( e, a, b, c, f(e,a,b,c)[0],f(e,a,b,c)[1],f(e,a,b,c)[2],f(e,a
,b,c)[3],f(e,a,b,c)[4],f(e,a,b,c)[5],f(e,a,b,c)[6],f(e,a,b,c)[7]))


main()
```

the output:

E	A	B	C	Y7	Y6	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0
1	1	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	1	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0