ECE 120: Introduction to Computing Spring 2019 ZJUI

due at start of section on Friday 29 March 2019

Homework 5: Boolean Algebra

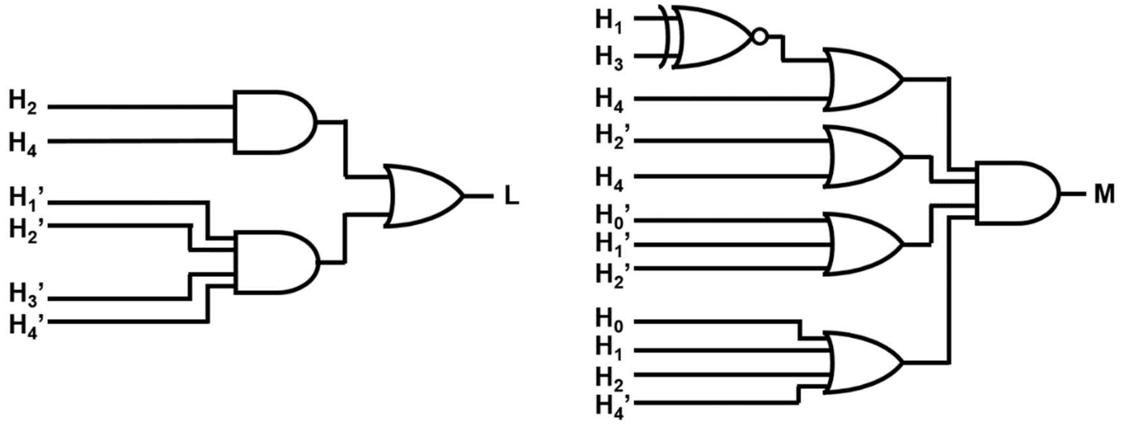
As with Homework #4, in this assignment, minimal SOP and POS refer to the K-map solutions with minimal area heuristic discussed in class (literals + operations, not counting complements on literals). When we ask for SOP and POS expressions, you need only consider expressions in these forms, and solving the K-map as shown in class produces the correct answer. The online exercises will help you to practice these skills.

1. Helping Your Friends

Your friends have developed a circuit to control the lights and music in their shared apartment, but they want you to check that the circuit works correctly for them.

The input to the circuit is a 5-bit number $H=H_4H_3H_2H_1H_0$ hour number ranging from 0 to 23, which changes value once per hour to match the current time.

Your friends have designed the logic shown below to output a signal L that turns the lights on (L=1) and off (L=0) at their usual hours. Similarly, the signal M turns their music on (M=1) and off (M=0) at their usual hours.



a. Analyze the circuit that calculates L and identify at what hours the lights turn on throughout the day (when L=1). For your answer, simply write the hour numbers (in decimal). For example, if the lights are on when H=4 to 7, you could write “4-7” or “4, 5, 6, 7,” or just “4 to 7.” Show your work.

For the upper AND gate to produce 1, we need H_2H_4 =1.
In this case, numbers with H_3=1 are at least 24, so mean nothing.
The upper AND gate thus produces 1 when H=101xx, or 20 to 23.

For the lower AND gate to produce 1, we need $H_1'H_2'H_3'H_4'$=1.
So H=0000x, or 0 to 1.

Overall, we have L=1 when H=20-23 or 0-1.

b. Analyze the circuit that calculates M and identify at what hours the music turns on throughout the day (when M=1). For your answer, simply write the hour numbers (in decimal). For example, if the music is on when H=16 to 20, you could write “16-20” or “16, 17, 18, 19, 20,” or just “16 to 20.” Show your work.

$M = (H_0+H_1+H_2+H_4')(H_0'+H_1'+H_2')(H_2'+H_4)((H_1 \oplus H_3)'+H_4)$

Let’s start with $H_4=1$, in which case $H_3=0$, since H must be less than 24.

Plugging in, we obtain $(H_0+H_1+H_2)(H_0'+H_1'+H_2')$

So M=1 when H is from 10001 to 10110, or 17 to 22.

Now consider the case of $H_4=0$. M now reduces to $M = (H_0'+H_1'+H_2')H_2'(H_1 \oplus H_3)'$

So $H_3=H_1$, $H_2=0$, and $H_2H_1H_0$ cannot be 000.

When $H_3=0$, we have only H=00001, or 1.

When $H_3=1$, we have H=0101x, or 10 to 11.

Putting it all together, we have M=1 when H=0-1, 10-11, or 17-22.

2. Analyzing NOR Structures

Based on the circuit shown to the right,

a. Write POS expressions for all four output variables: W, X, Y, and Z.

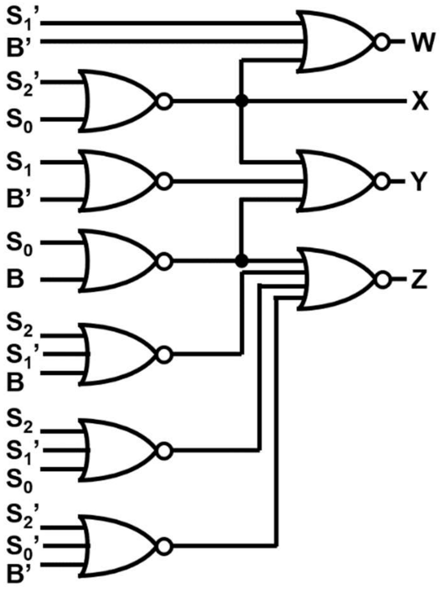
$X = (S_2' + S_0)' = S_2S_0'$

The rest are two-level logic, so we can simply read them from the circuit…

$W = S_1B(S_2' + S_0)$

$Y = (S_2' + S_0)(S_1 + B')(S_0 + B)$

$Z = (S_0 + B)(S_2 + S_1' + B)(S_2 + S_1' + S_0)(S_2' + S_0' + B')$



b. Write a truth table showing the values of all four outputs as a function of the four inputs B, S_2, S_1, and S_0.

B	S_2	S_1	S_0	X	W	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	1
0	0	1	0	0	0	0	0
0	0	1	1	0	0	1	0
0	1	0	0	1	0	0	0
0	1	0	1	0	0	1	1
0	1	1	0	1	0	0	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	1
1	0	1	0	0	1	1	0
1	0	1	1	0	1	1	1
1	1	0	0	1	0	0	1
1	1	0	1	0	0	0	0
1	1	1	0	1	0	0	1
1	1	1	1	0	1	1	0

3. VR Motion Waveforms

You are responsible for implementing the two waveforms shown below for controlling different motions in your company's new virtual reality (VR) simulation engine. You must design a circuit that emits the two waveforms based on a 1-bit waveform selector input F and on a counter value $C_2C_1C_0$. One waveform lasts six cycles, and the other lasts four. Based on these four bits of input, your circuit must produce a 2-bit unsigned "height" value H_1H_0.

a. Write a truth table (using inputs $FC_2C_1C_0$) for outputs H_1 and H_0. Don't forget to include the "don't-cares" in the table!

F	C_2	C_1	C_0	H_1	H_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	X	X
0	1	1	1	X	X
1	0	0	0	1	1
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	1	0	0
1	1	0	0	X	X
1	1	0	1	X	X
1	1	1	0	X	X
1	1	1	1	X	X

b. Turn your table from **part (a)** into a Karnaugh Map for each output variable.

H_1

		FC_2			
		00	01	11	10
C_1C_0	00	0	1	X	1
	01	0	0	X	1
	11	1	X	X	0
	10	1	X	X	0

H_0

		FC_2			
		00	01	11	10
C_1C_0	00	0	0	1	X
	01	1	1	0	X
	11	1	X	0	X
	10	0	X	1	X

c. Write a minimal SOP expression for both H_1 and H_0.

$H_1 = F'C_1 + C_2C_0' + FC_1'$

$H_0 = F'C_0 + FC_0'$

d. Write a minimal POS expression for both H_1 and H_0. Are your expressions equivalent to your answers to **part (c)**? Why or why not?

$H_1 = (F + C_2 + C_1)(C_2' + C_0')(F' + C_1')$

$H_0 = (F + C_0)(F' + C_0')$

The two are equivalent for H_0, but not for H_1, because the "don't cares" take on different values for the two solutions. For H_0, all "don't cares" have the same value in the two expressions.

e. Now replace all x's in your K-map for H_0 from **part (b)** with 0s and write a minimal SOP expression from the resulting K-map.

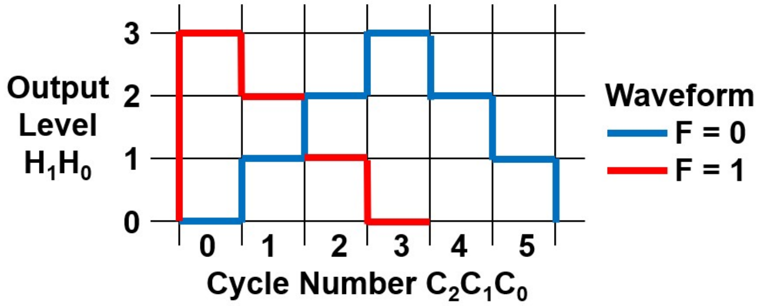
H_0

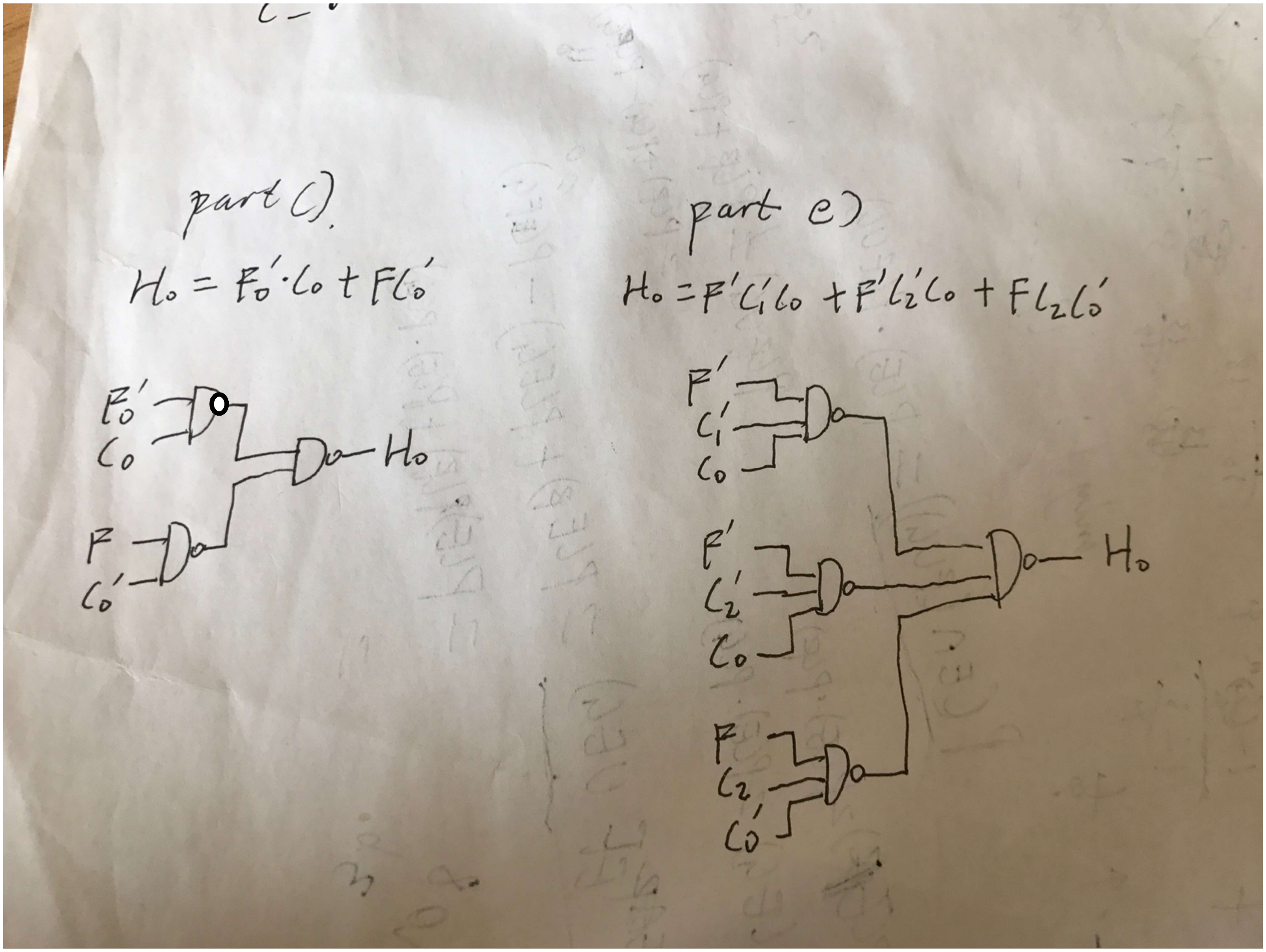
		FC_2			
		00	01	11	10
C_1C_0	00	0	0	1	0
	01	1	1	0	0
	11	1	0	0	0
	10	0	0	1	0

$H_0 = F'C_1'C_0 + F'C_2'C_0 + FC_2C_0'$

f. Implement your expressions for H_0 from **parts (c)** and **(e)** using only NAND gates. Assume that complemented inputs are available. Which circuit is simpler to implement? Why?

Circuit from **part (c)**. Using don't cares rather than picking a specific function allows us to choose a simpler function (with fewer literals and fewer operators).





part c).

$H_0 = F_0' \cdot C_0 + FC_0'$

part e)

$H_0 = F'C_1'C_0 + F'C_2'C_0 + FC_2C_0'$

4. **Simplifying with Don't Cares**
The function G(A,B,C,D) is specified by the Karnaugh map to the right.

G		AB			
		00	01	11	10
CD	00	1	1	0	x
	01	1	x	0	x
	11	1	0	1	0
	10	0	0	x	x

   a. List all prime implicants that include at least one '1' of function G(A,B,C,D).
   A'C', A'B'D, ABC, and B'C'
   b. Find a minimal SOP expression for G(A,B,C,D).
   G = A'C' + A'B'D + ABC
   c. Find a minimal POS expression for G(A,B,C,D).
   G = (C' + D)(A + B' + C')(A' + C)(A' + B)
   d. Are the functions that you found for **parts (b)** and **(c)** equivalent? Justify your answer.
   No, they are not equivalent. When ABCD=1110, the function from **part (b)** evaluates to 1, while the function from **part (c)** evaluates to 0.

5. **Serving Coffee: Another Word Problem**
Prof. Lumetta has heard that the library café may close. In an act of desperation, he copied down the menu of caffeinated beverages. You job, should you choose to accept it (and engineers always accept jobs that involve points towards a grade!), is to design the equations that control release of various ingredients for the drinks.

Given a four bit drink number $D=D_3D_2D_1D_0$, draw a K-map and calculate a minimal SOP expression for each of the following eight ingredients: W, water; S, steamed milk; F, foamed milk; H, hazelnut syrup; V, vanilla syrup; C, chocolate; R, caramel syrup; and I, ice. Each ingredient's output should equal 1 iff the drink D includes that ingredient.

Drinks and their contents are as follows. Drink #0 doesn't exist:

#1 espresso	espresso
#2 caffe Americano	espresso and water
#3 cold caffe Americano	espresso, water, and ice
#4 cappucino	espresso and foamed milk
#5 cold cappuccino	espresso, foamed milk, and ice
#6 cocoa cappuccino	espresso, foamed milk, and chocolate
#7 caramel macchiato	espresso, foamed milk, and caramel syrup
#8 caffe latte	espresso and steamed milk
#9 cold caffe latte	espresso, steamed milk, and ice
#10 hazelnut caffe latte	espresso, steamed milk, and hazelnut syrup
#11 cold hazelnut caffe latte	espresso, steamed milk, hazelnut syrup, and ice
#12 vanilla caffe latte	espresso, steamed milk, and vanilla syrup
#13 cold vanilla caffe latte	espresso, steamed milk, vanilla syrup, and ice
#14 caffe mocha	espresso, steamed milk, and chocolate
#15 cold caffe mocha	espresso, steamed milk, chocolate, and ice

One more question: if espresso is put into every drink, what is produced if D=0 is used with your equations?

W

		D_3D_2			
		00	01	11	10
D_1D_0	00	x	0	0	0
	01	0	0	0	0
	11	1	0	0	0
	10	1	0	0	0

W = $D_3'D_2'D_1$

S

		D_3D_2			
		00	01	11	10
D_1D_0	00	x	0	1	1
	01	0	0	1	1
	11	0	0	1	1
	10	0	0	1	1

S = D_3

F

		D_3D_2			
		00	01	11	10
D_1D_0	00	x	1	0	0
	01	0	1	0	0
	11	0	1	0	0
	10	0	1	0	0

F = $D_3'D_2$

H

		D_3D_2			
		00	01	11	10
D_1D_0	00	x	0	0	0
	01	0	0	0	0
	11	0	0	0	1
	10	0	0	0	1

H = $D_3D_2'D_1$

V

		D_3D_2			
		00	01	11	10
D_1D_0	00	x	0	1	0
	01	0	0	1	0
	11	0	0	0	0
	10	0	0	0	0

$V = D_3D_2D_1'$

C

		D_3D_2			
		00	01	11	10
D_1D_0	00	x	0	0	0
	01	0	0	0	0
	11	0	0	1	0
	10	0	1	1	0

$C = D_3D_2D_1 + D_2D_1D_0'$

R

		D_3D_2			
		00	01	11	10
D_1D_0	00	x	0	0	0
	01	0	0	0	0
	11	0	1	0	0
	10	0	0	0	0

$R = D_3'D_2D_1D_0$

I

		D_3D_2			
		00	01	11	10
D_1D_0	00	x	0	0	0
	01	0	1	1	1
	11	1	0	1	1
	10	0	0	0	0

$I = D_3D_0 + D_2D_1'D_0 + D_2'D_1D_0$

D=0 produces espresso.

6. **Helping Your Classmates**

Let’s be honest: the canteen is just plain overcrowded at noon (1200). Help your classmates by producing a circuit that helps them decide when to leave for the canteen.

Input variables include the following:

E—end of service; 1300 is coming soon if E=1
N—noon; if they leave when N=1, they will arrive too close to 1200
R—rain; if R=1, it’s raining outside (again!)
C—conversation; if C=1, there’s a good conversation in progress

Draw a K-map and calculate minimal SOP and POS expressions for the function L(E,N,R,C) based on the following rules:

1. If the time is approaching 1300, leave immediately.
2. Otherwise, if leaving means arriving close to 1200, wait for a while.
3. Otherwise, if the rain has stopped or conversation has stalled (no good conversation in progress), leave immediately.
4. Otherwise, wait.

Note that not all input combinations are possible. Handle impossible cases appropriately.

L		EN			
		00	01	11	10
	00	1	0	X	1
	01	1	0	X	1
RC	11	0	0	X	1
	10	1	0	X	1

SOP: L = E + N’R’+ N’C’

POS: L = N’(E + R’ + C’)

7. Printing a K-map

Update your Subversion repository to obtain the **hw5** subdirectory. In that directory is a file called **kmap.c**, which is intended to print a 3-input K-map to the monitor. Unfortunately, the code is not quite complete. You must add two pieces to the code.

a. First, calculate the input variable **c** as a function of the variables **b** and **d**. Read the code for more details.

   c = b ^ d;

b. Next, use C operators to calculate the function

   f(a,b,c) = abc′ + c(a′ + b′) + b′c′.

   Be sure that **f** can take values of only 0 and 1. Then print the value of **f** to the monitor using **printf**. Use enough spaces so that the K-map output looks correct.

Code:

```
f = ( ( (a & b & (~c)) | (c & ( (~a) | (~b) ) ) | ( (~b) & (~c) ) ) & 1);
printf ("%d  ", f); // two spaces
```

Output:

```
          bc
      00 01 11 10
    +-------------
a=0 | 1  1  1  0
a=1 | 1  1  0  1
```

Turn in a copy of your modified code along with a copy of your program's output.