Homework 5

Boolean algebra and two-level design

1. Simplifying with Don’t Cares

The functions $f(w,x,y,z)$ and $g(w,x,y,z)$, specified by their Karnaugh maps below, are almost the same as the ones in Homework 4 Problem 4, except that they have don’t cares in some cells. That means that the expressions in Homework 4 solution work for these functions too, but are not necessarily minimal.

\[ f(w,x,y,z) \]

\[
\begin{array}{cccc}
  & 00 & 01 & 11 & 10 \\
\hline
00 & 0 & 1 & X & 0 \\
01 & 1 & X & 0 & 1 \\
11 & 1 & X & 0 & 1 \\
10 & 0 & 1 & X & 0 \\
\end{array}
\]

\[ g(w,x,y,z) \]

\[
\begin{array}{cccc}
  & 00 & 01 & 11 & 10 \\
\hline
00 & 1 & 1 & X & 0 \\
01 & 1 & X & 0 & 0 \\
11 & 1 & X & 0 & 1 \\
10 & 0 & 0 & X & 0 \\
\end{array}
\]

For each function:
1. Circle on the K-map and list all prime implicants that you must consider for a minimal SOP.
2. Find a minimal SOP expression. Is it unique? Explain your answer.
3. Find a minimal POS expression. Is it unique? Explain your answer.

If you want to have more practice on K-maps, please feel free to do the problems on this page.

2. Duality and NAND/NOR Circuits

Let \( m(a,b,c,d) = a'bc + ac'd' \) and let \( n(a,b,c,d) \) be its dual.

1. Write the expression \( n(a,b,c,d) \).
2. Implement \( m(a,b,c,d) \) using a 2-level network of NAND gates only. Do not simplify the expression. Assume complements of the inputs are available.
3. Implement \( n(a,b,c,d) \) using a 2-level network of NOR gates only. Do not simplify the expression. Assume complements of the inputs are available.

3. Six-Sided Die Display Circuit

In this problem, you will design a logic circuit to display the six faces of a die using 7 LEDs. For example, the value 5 is shown on the display below:

Your circuit takes three bits XYZ as input and outputs the patterns shown in the table below, by lighting up the corresponding LEDs. For example, for the value 5, the input is XYZ=101 and the output is ABCDEFG=1011101.

<table>
<thead>
<tr>
<th>Value</th>
<th>Input XYZ</th>
<th>Output Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>001</td>
<td>●</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>●●</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>●●●</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>●●●●</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>●●●●●</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>●●●●●●</td>
</tr>
</tbody>
</table>

Design the logic circuit following the steps below. Note that it is possible to simplify the Boolean expressions if we assume that we do not care about the behavior for invalid input cases.
1. Write the **complete** truth table for input variables X, Y and Z to output variables A, B, C, D, E, F and G. Remember that for 3 input variables, the truth table must have 8 rows.
2. Draw K-Maps for all 7 output variables.
3. Use the above K-Maps to write minimal SOP expressions for the output variables. Show your work.
4. Using **only two 2-input logic gates in total**, draw the complete logic circuit diagram based on the SOP equations for your system, with inputs X, Y and Z entering on the left side, and outputs A, B, C, D, E, F and G on the right side. Assume complemented inputs are not available.
5. Redraw your logic circuit diagram using **2-input NAND gates only**. Just like before, assume complemented inputs are not available.
6. How many 7400 TTL chips do you need to implement this circuit on a protoboard? See Lab 3.
7. Draw the output patterns that will be displayed if the invalid inputs XYZ=000 and XYZ=111 are entered into your logic circuit.

### 4. Logic Design Word Problem

Ye Olde 120 Pizza Shoppe has the following 5 (FIVE) pizza toppings available: anchovies (a), broccoli (b), mushrooms (m), pepperoni (p), and sausage (s).

- Let a represent “the pizza has anchovies”.
- Let b represent “the pizza has broccoli”.
- Let m represent “the pizza has mushrooms”.
- Let p represent “the pizza has pepperoni”.
- Let s represent “the pizza has sausage”.

a. For each of the following conditions (C1 - C3), give a **simple** boolean expression which equals 1 precisely when the given condition is satisfied. Your expression may use only AND, OR, and NOT operations, and may **not be a brute force listing of lots of terms**.

   Ex: "The pizza must have anchovies and broccoli" would be denoted by ab. It would **not** be acceptable to write ab+pp'+mm'

   C1. The pizza must have anchovies or broccoli.
   C2. The pizza must have anchovies or mushrooms but not both.
   C3. The pizza must have neither pepperoni nor sausage.

b. Define the function \( F(a, b, m, p, s) = 1 \) if and only if all the pizza conditions stated in C1, C2 and C3 are satisfied.

   1. Give a Boolean expression for F.
   2. Express F in canonical SOP form. Hint: If you have more than 5 minterms, you made a mistake!
   3. List all acceptable pizza combinations. For example, your answer(s) should be of the form: no anchovies, no broccoli, mushrooms, pepperoni, sausage

### 5. Programming in C
#include <stdio.h>
int main()
{
    unsigned int a, b, c, d;
    unsigned int f;

    /* Print header for K-map. */
    printf(" bc        
");    // bc
    printf(" 00 01 11 10 
");  // 00 01 11 10
    printf(" ______________
");  // 

    /* row-printing loop */
    for (a = 0; 2 > a; a = a + 1) {
        printf("a=%u | ", a);
        // Loop over input variable b in binary order. */
        for (b = 0; 2 > b; b = b + 1) {
            // Loop over d in binary order.*/
            for (d = 0; 2 > d; d = d + 1) {
                /* Use variables b and d to calculate *
                 * input variable c (iterated in *
                 * Gray code order). */
                /* CALCULATE c HERE. */

                /* Calculate and print one K-map entry *
                 * (function F(a,b,c) ). */
                /* INSERT CODE HERE. */
            }
        }
    }

    /* End of row reached: print a newline character. */
    printf("\n");
}

return 0;
}

The above program (hw5p5.c) provides some initial code to generate a 3-variable K-Map for a 3-variable Boolean function. For example, for function \( F(a,b,c) = ab + b'c' \) the program should print the following:

<table>
<thead>
<tr>
<th>bc</th>
<th>00 01 11 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a=0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>a=1</td>
<td>1 0 1 1</td>
</tr>
</tbody>
</table>

Complete the program above. For your portion of the code at "CALCULATE c HERE", you must begin by ensuring that the variable c follows
Gray code order and runs from 0 to 1 when \( b=0 \), but from 1 to 0 when \( b=1 \). At "INSERT CODE HERE", you must then calculate the function \( F(a, b, c) = (a'+b') (a+b'+c) (b+c') \). Do not otherwise change the program. For full points, use C's bitwise operators (\&, |, ^ and ~) to perform both of these calculations.

1. Print your code and attach it to your written homework
2. Print and attach the output of your program; that is, the K-Map for \( F(a, b, c) = (a'+b') (a+b'+c) (b+c') \).

**Hint 1:** You may find it helpful to verify the output on a paper version of the K-map.

**Hint 2:** Remember that bitwise operators operate on the entire pattern of 32-bits (here the bit pattern corresponds to unsigned int). It might be useful for you to print the output \( F \) to see for yourself what is happening with the bits. Then, before having your code printing \( F \), you must ‘discard’ all bits other than the 1’s position (one way would be to use an & operation with an appropriate value).

Again, you may not hardcode the K-map or change the program other than as specified. Your program must compute \( F(a, b, c) = (a'+b') (a+b'+c) (b+c') \).