

Computer Science 141

Weekly Assignment #2

Due 1 P.M. Monday, Sep. 28

You may hand in your solution via email (cs141-2009-staff@eecs.harvard.edu) or in person at the beginning of lecture.

1 Required Reading

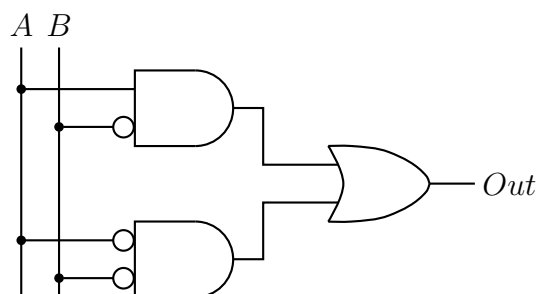
Please read sections 2.6-2.10

2 Problems (50 total points)

Please show your work in your solutions. If you need any help, please contact the TFs (Jian and Svilen) with questions or meeting requests at cs141-2009-staff@eecs.harvard.edu.

1. (9 points) Draw schematics for the following functions using AND, OR and NOT gates (do not simplify or factor the functions). The first example has been done for you; complete the exercise in a similar fashion.

a. $A\bar{B} + \bar{A}B$



- b. $A \bar{B} + \bar{A} B + A B \bar{C}$
 c. $(\bar{A} + B)(A + \bar{B} + \bar{C})(A + C)$

d. $A(\bar{B} + C)\bar{D} + \bar{A} D$

2. (6 points) Design a hall light circuit to the following specification: There is a “Master” switch and three hall switches. If the “Master” switch is off, the hall light is off regardless of the three switches. If the “Master” switch is on, flipping any of the three hall switches toggles the light on and off. List any assumptions you need to make, and show the implementation of the function using inputs A , B and C as hall switches and M as the master switch. You may use any combination of AND, OR, NOR and XOR gates. The following symbol is used to represent XOR gates (see Problem 8 for information on XOR gates with more than 2 inputs):



For five points of extra credit, show how to **modify the existing circuit** so that the “Master” switch can be in one of three states: 1, in which case the light is always on, 0, in which case it’s always off, and T , in which the three hall switches toggle the light on and off. State all assumptions. Use only the components we covered so far.

For the remaining problems, the online truth table generator (found on our website) will come in very handy when verifying whether the expressions you come up with in fact generate the desired truth tables.

3. (4 points) Form the complement of the following function using DeMorgan’s theorem. Do not simplify.

$$f(A, B, C, D) = (\bar{A} + B C D + A \bar{D})(B + \bar{A} C + D) + A \bar{B} \bar{C}$$

4. (6 points) Given the following function f in POS form, not necessarily minimized,

$$f(A, B, C, D) = (A + \overline{C})(B + \overline{D})(\overline{A} + \overline{B} + D)$$

(a) Express f in the canonical SOP form using the minterm list (Σ) notation

(b) Re-express f in minimized SOP form

(c) Express \overline{f} in minimized SOP form

(d) Re-express \overline{f} in minimized POS form

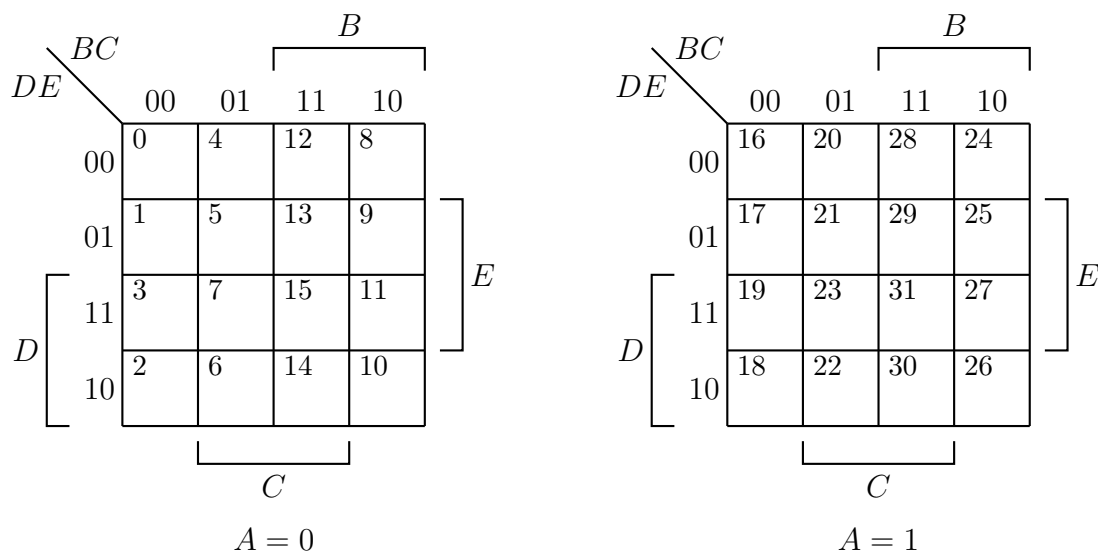
5. (6 points) Using the theorem of Boolean algebra, verify that NAND is functionally complete - that is, any Boolean function can be constructed by a combination of NANDs. To prove that, you need to express each of the following in terms of NAND only, showing your work:

(a) \overline{A}

(b) $A B$

(c) $A + B$

6. (6 points) A Karnaugh map for a 5-variable function can be drawn as shown below. In such a map, cells that occupy the same relative position in the $A = 0$ and $A = 1$ submaps are considered to be adjacent.



(a) Find a minimal sum-of-products expression for f using a 5-variable map (d denotes *Don't Care's*):

$$f = \Sigma_{A,B,C,D,E}(2, 7, 8, 10, 11, 12, 13, 15, 21, 23, 26, 29, 31) + d(5, 9, 14, 18, 27)$$

(b) Try to minimize a 5-input divisible-by-3 function (it should return true when the input number is divisible by 3). Can we do better than the SOP representation?

7. (10 points) Design a combinational circuit with four data inputs B_3, B_2, B_1, B_0 and one output P . P should be 1 if the binary number $B_3B_2B_1B_0$ is prime and 0 otherwise. You may assume that the input number will always be greater than 1. For example, If $B_3B_2B_1B_0 = 1011$, $P = 1$. Fill in a single truth table for the combinational logic function P , draw its Karnaugh map, and find the minimized SOP form. Finally, find the least number of gates necessary to implement P (not necessarily in two-level logic). Assume you have access to **2-input** AND, OR and XOR gates **only**, and that you may negate any input.

8. (3 points) XOR for more than two variables is defined as a parity function (*i.e.* it's equal to 1 if the number of 1's on input is odd). Why is XOR not minimizable (check it out yourself if you don't believe it)? Hint: look at two adjacent cells of a Karnaugh map.