

CSE 140 Homework Two

August 20, 2013

*Only **Problem Set Part B** will be graded. Turn in only **Problem Set Part B** which will be due on August 27, 2013 (Tuesday) at 4:00pm.*

1 Problem Set Part A

- Roth&Kinney, 6th Ed: 2.5 - 2.8
- Roth&Kinney, 6th Ed: 2.12
- Roth&Kinney, 6th Ed: 2.22
- Roth&Kinney, 6th Ed: 2.29 - 2.30
- Roth&Kinney, 6th Ed: 3.9 - 3.10
- Roth&Kinney, 6th Ed: 3.12
- Roth&Kinney, 6th Ed: 3.16 - 3.17
- Roth&Kinney, 6th Ed: 3.19 - 3.21
- Roth&Kinney, 6th Ed: 3.29 - 3.30
- Roth&Kinney, 6th Ed: 3.33 - 3.34
- Roth&Kinney, 6th Ed: 3.37
- Roth&Kinney, 6th Ed: 4.5 - 4.6
- Roth&Kinney, 6th Ed: 4.8
- Roth&Kinney, 6th Ed: 4.17 - 4.18
- Roth&Kinney, 6th Ed: 4.25
- Roth&Kinney, 6th Ed: 4.32
- Roth&Kinney, 6th Ed: 4.42
- Roth&Kinney, 6th Ed: 5.10 - 5.12
- Roth&Kinney, 6th Ed: 5.21
- Roth&Kinney, 6th Ed: 5.28 - 5.29
- Roth&Kinney, 6th Ed: 5.32 - 5.33
- Roth&Kinney, 6th Ed: 5.42
- Roth&Kinney, 6th Ed: 6.4 - 6.5
- Roth&Kinney, 6th Ed: 6.11
- Roth&Kinney, 6th Ed: 6.17

1 (Binary-Encoded Ternary Adder) A number of students taking this course were confused by the complexities of the multiplication of signed 3's complement numbers. In order to test their various theories about the mechanics of this operation, one pair of partners decided that they would invent a 3's complement multiplication circuit. However, they have become stuck on the implementation of a Binary-Encoded Ternary Adder, and are arguing about the best way to implement that. (Note: while ideally their adder should take in 5 bits – 2 for each summand, and one carry in – the pair is uncomfortable with using 5-input Karnaugh maps, and so they are focused on constructing a carry-independent adder that takes in only 4 inputs, two for each summand, which they intend to later modify to incorporate the carry-in.) Seeing that you are doing fairly well in the course, this befuddled pair approaches you and asks you to help them decide who is right about certain aspects of the implementation.

(Part A) At present, the pair has agreed to use the binary sequence 00 to encode “0” and 01 to encode “1”, but they disagree about how they should encode “2”. One thinks that using **10** will produce the most efficient **sum-of-products** logic. To justify his point, he has filled out the following Karnaugh maps for the functions for the sum functions, S_0 and S_1 (where S_1 is the most significant bit of the sum), and the carry-out generation function C :

$S_0: a_1a_0 \backslash b_1b_0$	00	01	11	10
00	0	1	X	0
01	1	0	X	0
11	X	X	X	X
10	0	0	X	1

$S_1: a_1a_0 \backslash b_1b_0$	00	01	11	10
00	0	0	X	1
01	0	1	X	0
11	X	X	X	X
10	1	0	X	0

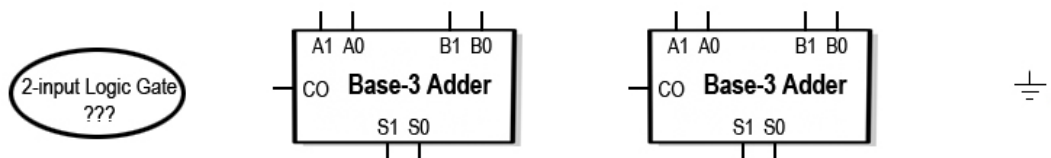
$C: a_1a_0 \backslash b_1b_0$	00	01	11	10
00	0	0	X	0
01	0	0	X	1
11	X	X	X	X
10	0	1	X	1

The other partner, who has not had time to write up his Karnaugh maps, nonetheless asserts that while the **11** encoding does not generate as efficient a sum-of-products implementation, the **product-of-sums** implementation is actually more efficient.

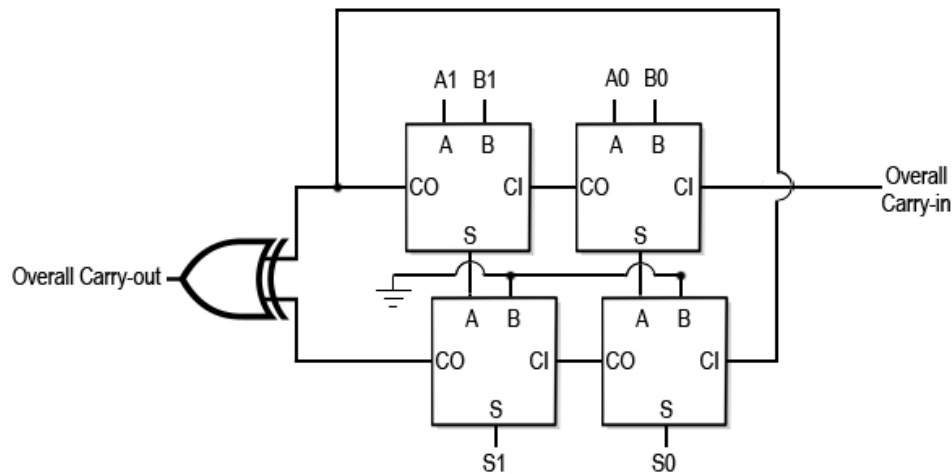
In the space provided next to the already completed Karnaugh maps, help the second partner draw his Karnaugh maps for the functions S_0 , S_1 , and C under the 11 encoding for the value “2” and obtain the minimal **product-of-sums** implementation. Then compare the product-of-sums implementation of each of the functions with the first partner’s sum-of-products implementation. Decide which partner’s idea is superior, or, if they are equivalent, scold them for wasting time on a pointless fight.

(Part B) After you have made your demonstration, the first partner has a thought (finally) and suggests that perhaps using a loose encoding, using either 10 or 11 to encode “2”, will produce an even more efficient logic than either of the two previous encodings. The second partner says that this is a horrible idea, and that the first partner must have some loose encodings in his head to think of a silly idea like that. Is the second partner right about the first one, or is he unwittingly projecting his own mental incompetencies onto his partner? Draw the new Karnaugh maps for the functions S_0 , S_1 , and C , and compare the efficiency of the logic for each (you may use either a product-of-sums or a sum-of-products implementation, and this may vary from function to function).

(Part C) After they have finished implementing their carry-independent adder, the partners need to figure out how to modify it to produce a full adder. They think about it for a moment, and the first partner says that you can accomplish this without modifying their existing component; they just have to use two carry-independent adders to accomplish it, along with a logic gate and a circuit ground. The second partner is skeptical of this assertion. After a moment, you realize that the first partner is right, and chime in on their conversation. You and the first partner make your case to the second for several minutes, but even with your blessings, the idea is not germinating in his mind. Please use your wonderful visual arts skills to finish the drawing below of the implementation of the carry-sensitive adder composed of the carry-insensitive adders that gives the correct 2-bit result and correct carry-out. Make the appropriate wiring connections, label which pins need to take the input summands, which pin(s) take(s) the carry-in, label the pins which output the sum and carry-out, and identify the mysterious unknown logic gate.



(Part D) Another group overheard the idea of the first group, and has attempted to implement their own binary-encoded ternary adder. However, they noticed early on that if they used the 00, 01, 10 encoding for “0”, “1”, and “2”, respectively, they could construct a Ternary full-adder that simply used single-bit binary full-adders. They came up with the following design:



However, it's not quite working right for all inputs. They come to you and ask you to identify and correct the error for them. Identify the input combinations that produce incorrect results, and explain why the circuit outputs the wrong results in these cases. After that, fulfill the great responsibility that comes with the great power of artistic competence to draw a corrected circuit, **making as few changes as necessary to ensure the circuit's correct functionality.**

2 (Static Hazards) As you have learned in class, static hazards manifest themselves when adjacent minterms (or maxterms) are not covered by a common implicant (or implicate). For example, the function $x'y' + xz$ contains a static-1 hazard because minterms $x'y'z$ and $xy'z$ are not covered by a common implicant, as shown in the K-map below, where one of the prime implicants in a minimal cover is bolded and the other is italicized:

x/yz	00	01	11	10
0	<i>1</i>	1	0	0
1	0	1	1	0

(Part A) For the following functions, please specify if

- The function will generate a static-1 hazard
- There are other minimal representations of the same function that will also generate a static-1 hazard
- If there is another equivalent function that will also generate a static-1 hazard, report the function

$$xy' + y'z + x'y$$

The function will/will not (circle one) generate a static-1 hazard.

There are/aren't (circle one) other minimal representations of the same function that will generate a static-1 hazard.

One other static-1 hazard representation is: (put N/A if there are none)

$$xy' + wz + yz$$

The function will/will not (circle one) generate a static-1 hazard.

There are/aren't (circle one) other minimal representations of the same function that will generate a static-1 hazard.

One other static-1 hazard representation is: (put N/A if there are none)

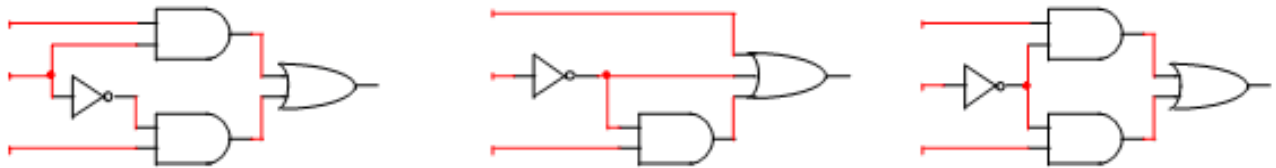
$$w'z + wz' + xy'z$$

The function will/will not (circle one) generate a static-1 hazard.

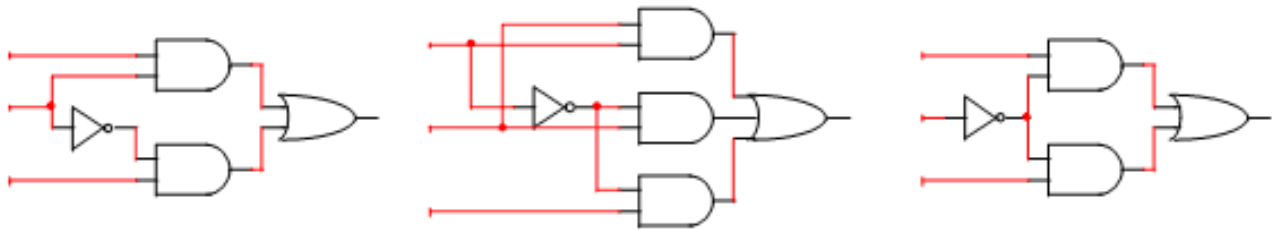
There are/aren't (circle one) other minimal representations of the same function that will generate a static-1 hazard.

One other static-1 hazard representation is: (put N/A if there are none)

(Part B) How many 3-variable 3-minterm functions contain static-1 hazards in their minimal implicant cover? Please explain your reasoning. If there are static-1 hazards, please circle the general schematic form in which such a hazard will take place.



(Part C) How many 3-variable 4-minterm functions contain static-1 hazards in their minimal implicant cover? Please explain your reasoning. If there are static-1 hazards, please circle the general schematic form in which such a hazard will take place.



(Part D) How many 3-variable 4-minterm functions that contain static-1 hazards in their minimal implicant cover will also manifest static-0 hazards in their minimal *implicate* cover?

3 (Tabulation & Quine-McCluskey) (Part A) A prime implicate table was derived based on the Tabulation method consisting of a set of unspecified maxterms. At this point, we observed that the maxterm that you see in the last column of the following table, **f**, was really not a maxterm but a *don't care*. Please make the appropriate modifications to the prime implicate table to reflect this new knowledge and then solve the problem by performing the row/column domination aspects of the Quine-McCluskey method. Write the final cover, F , as a function of the prime implicants selected.

Q-M Table

	a	b	c	d	e	f
P_1	X	X				
P_2		X	X			
P_3			X	X		
P_4				X	X	
P_5					X	X
P_6	X					X

$F =$ _____

(Part B) The Tabulation method is applied to the **minterms** of a 4-variable Boolean function. In the list of the 0-subcubes, the number of elements in each G_i group is reported as follows:

	G_0	G_1	G_2	G_3	G_4
Elements	0	2	6	2	1

To jolt your memory (unnecessary though it probably is), we remind you that an 'i-subcube' denotes a subcube with 2^i minterms, while each G_k group contains subcubes which are denoted in the tabulation method by entries whose 1's count equals k .

Given the information outlined above, a student is trying to apply the Tabulation method to the **maxterms** of the same function. How many **prime implicants** (not implicants) should (s)he get at the end? What is the cube size of each prime implicate (e.g., 0-subcube, 1-subcube, 2-subcube)? Please provide a brief reasoning.

The use of the Quine-McCluskey method on a large circuit is typically a quite sizable and time consuming enterprise. One possible idea to reduce its complexity is to adopt a *divide and conquer* approach. If the prime implicants can be partitioned into disjoint sets and the minterms covered by these sets are also disjoint, the larger prime implicant table can be obviously divided into smaller tables with all the off-diagonal submatrices being null. The Quine-McCluskey method can therefore be applied onto the non-empty submatrices individually, thus reducing the amount of work that needs to be done in row and column comparisons. This approach is concretely shown in the left one of the following two figures.

	Minterm columns	
PI rows		empty
	empty	

Divide and conquer approach

	Minterm columns	
PI rows		empty
	X empty	

Single 'X' in off-diagonal submatrix

A group of CSE 140 students are trying to decompose a large prime implicant table using the approach described above. Unfortunately, life is not always perfect. They have managed to partition the initial table into 4 pieces. Yet only one of the off-diagonal submatrices happens to be null, while the other one has a single 'X' in it, as shown on the right one of the above two figures.

Rather than giving up decomposition and going back to the initial table, these students are wondering whether it is legal to delete this single entry denoted by 'X' in the figure. They are debating among the following three possibilities:

- (a) Deleting the 'X' could possibly make the resultant final cover incorrect.
- (b) Deleting the 'X' will result in a final cover that is correct but possibly non-minimal.
- (c) Deleting the 'X' has no impact on either the correctness or the cost of the resultant final cover.

Thankfully, there seems to be a bit more information about the situation as outlined in the three parts on the next page. Please choose among the three alternatives just outlined in addressing the problems in Parts C, D, & E.

(Part C) Assume that the column that contains this 'X' **dominates** a column to its left. Under this assumption, please help the students determine whether it is legal to delete this 'X' by selecting among the three outlined statements. Please provide a brief reasoning for your answer.

(Part D) Assume that the row that contains this 'X' **dominates** a row to its bottom. Under this assumption, please determine whether it is legal to delete this single 'X' (by selecting among the three outlined statements). Please provide a brief reasoning for your answer.

(Part E) Assume that the row that contains this 'X' **is dominated** by the **union** of two disjoint rows, one to its top and one to its bottom. Under this assumption, please determine whether it is legal to delete this single 'X' (by selecting among the three outlined statements). Please provide a brief reasoning for your answer.