

CSE 140 Homework Three

February 21, 2008

*Only **Problem Set Two** will be graded. Turn in only **Problem Set Two** which will be due on February 27, 2008 (Wednesday) at 11:00am.*

1 Problem Set One

- textbook 3.10(c)(d)
- textbook 3.12(a)(d)
- textbook 3.14(a)(b)
- textbook 3.16
- textbook 3.19(b)
- textbook 4.4(d)
- Using the map method, determine the prime implicants of 4.4(d)
- textbook 4.5(a)(d)
- textbook 4.6(a)(d)
- textbook 4.7(b)
- textbook 4.9(b)
- textbook 4.10(b)
- textbook 4.17(a)
- textbook 4.18

2 Problem Set Two

1 (Tabulation Method) (Part A) State whether the following statements are True (T) or False (F).

- A 0-subcube in G_0 can always be merged with a 0-subcube in G_1 to generate a 1-subcube.
- A 1-subcube in G_0 can always be merged with a 1-subcube in G_1 to generate a 2-subcube.
- A logic function exists such that a subcube in G_i and another subcube in G_{i+2} can be combined into a larger subcube.
- No two subcubes that belong to the same group, G_i , can be combined into a larger subcube.

(Part B) The Quine McCluskey method is applied on the minterms of a 4-variable logic function. You know that the function has 8 minterms and the minimum distance between any two minterms is two. You also know that in the list of 0-subcubes, there is a single element in the G_0 group. How many elements are there in the other 0-subcube groups, namely, G_1 , G_2 , G_3 , and G_4 for this logic function?

(Part C) The Quine McCluskey method is applied on the minterms of a 4-variable function. You know that the function has 8 minterms. The Quine McCluskey algorithm terminates with 4 1-subcube Prime Implicants (PIs). You also know that in the list of 0-subcubes, there is a single element in G_0 . How many elements are there in the other 0-subcube groups, namely, G_1 , G_2 , G_3 , and G_4 ?

2 (Karnaugh Maps & Boolean Decomposition) As you must have noticed from our discussion of minterms and maxterms and also from our discussion of Karnaugh maps to derive minimal sum-of-products and product-of-sums representations, every Boolean function can be expressed by just using AND, OR and NOT gates. However, in terms of gate count, the total number of gates (not counting the NOT gates) that are used in the implementation may not be minimal when compared to non-standard representations. The use of the functions such as XOR and XNOR to generate nonstandard implementations may reduce the total gate count.

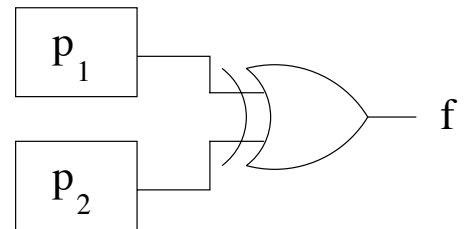
(Part A) Given the following **Karnaugh map** of the function, f , please derive the *minimal sum-of-products* representation.

ab \ cd	00	01	11	10
00	1	1	0	0
01	0	1	0	1
11	1	0	0	1
10	0	0	0	0

$f =$

(Part B) For the same **Karnaugh map** given in (PART A), decompose the function into 2 parts, p_1 and p_2 , that are in sum-of-products form and compose the resulting parts by an **XOR** gate to obtain the original function as depicted in the following graph. Please identify the decomposition that requires minimum total number of AND and OR gates.

ab \ cd	00	01	11	10
00	1	1	0	0
01	0	1	0	1
11	1	0	0	1
10	0	0	0	0



Karnaugh map for p_1 :

ab \ cd	00	01	11	10
00				
01				
11				
10				

Karnaugh map for p_2 :

ab \ cd	00	01	11	10
00				
01				
11				
10				

$f = (\quad) \oplus (\quad)$

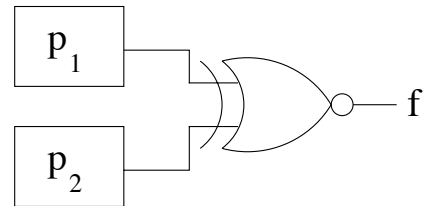
(Part C) Given the following **Karnaugh map** of the function, f , please derive the *minimal sum-of-products* representation.

ab \ cd	00	01	11	10
00	1	1	0	0
01	1	0	1	0
11	1	0	1	0
10	1	1	0	0

$f =$

(Part D) For the same **Karnaugh map** given in (PART C), decompose the function into 2 parts, p_1 and p_2 , that are in sum-of-products form and compose the resulting parts by an **XNOR** gate to obtain the original function as depicted in the following graph. Please identify the decomposition that requires minimum total number of AND and OR gates.

ab \ cd	00	01	11	10
00	1	1	0	0
01	1	0	1	0
11	1	0	1	0
10	1	1	0	0



Karnaugh map for p_1 :

ab \ cd	00	01	11	10
00				
01				
11				
10				

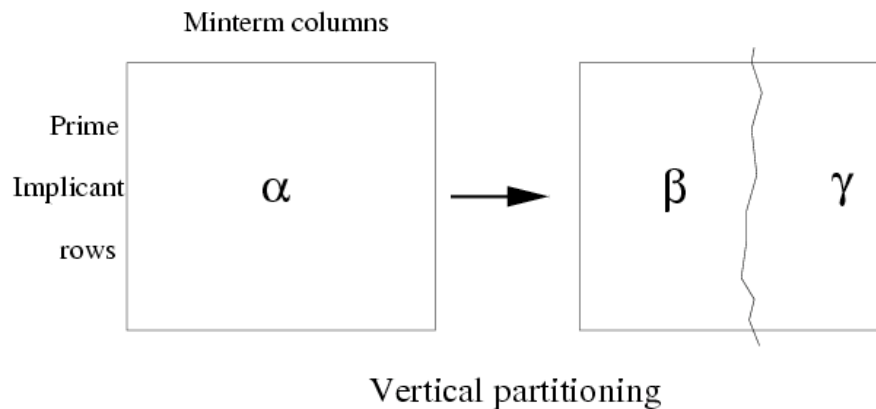
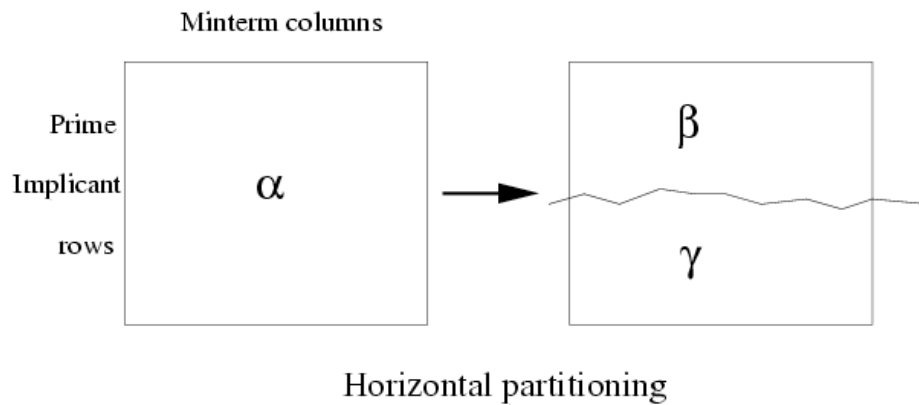
Karnaugh map for p_2 :

ab \ cd	00	01	11	10
00				
01				
11				
10				

$f = (\quad) \odot (\quad)$

3 (Quine-McCluskey method) Boolean function simplification for a large circuit is typically a quite sizable and time consuming enterprise. A CSE 140 intern at a CAD company has suggested that the whole process can be streamlined if a divide and conquer approach is adopted. She recommends that the large prime implicant tables be divided (horizontally and/or vertically) so as to reduce the amount of work that needs to be done in row and column comparisons. A study group has been formed out of fellow CSE 140 students to pass judgement as to whether this idea works. (No collaboration during the midterm, of course!) You are supposed to see if the idea of the intern works by examining whether the resulting functions are a competitive answer sizewise to the function and also to examine if they are indeed functionally equivalent. This is the spirit of the two subparts explored later on in the question.

But first a set of ground rules. We will denote the original prime implicant table as α , and the two horizontal decompositions as β and γ . We will denote the two vertical decompositions also as β and γ . We assume in the case of horizontal partitioning that each of the β 's and γ 's have at least one prime implicant for each minterm. We assume of course that none of the partitions β and γ is trivially empty.



(Part A) In this question, hardware cost in terms of gate counts is considered. (To simplify the model, we assume that decomposition of a multiple input gate into several smaller, for example, two-input gates is equivalent in terms of hardware costs.) Compare the implementations both for the horizontal and the vertical partitionings of the resultant functions denoted in the columns after applying the Quine-McCluskey method. For each of the boxes, four possibilities hold, denoted as (a) through (d). The question asks you to mark the appropriate option in each of the blocks.

Please note that an incorrect answer will subtract a correct answer, resulting possibly in negative points.

- (a) Always costs more than or equal to α
- (b) Always costs less than or equal to α
- (c) Always costs the same as α
- (d) Might cost the same, more or less than α depending on the specific function α and the partitioning

Hardware cost	β	γ	β AND γ	β OR γ
Horizontal partitioning				
Vertical partitioning				

(Part B) Of course, even more important than size is correctness. We are asking you in this part to check if our original logic function α is equivalent to any one of the four possibilities shown in the columns. Determine in the following table, which representation is **functionally** identical to α by filling in true (T) or false (F) accordingly.

Please note that an incorrect answer will once again subtract a correct answer, resulting possibly in negative points.

Functional identical	$\alpha = \beta$	$\alpha = \gamma$	$\alpha = \beta$ AND γ	$\alpha = \beta$ OR γ
Horizontal partitioning				
Vertical partitioning				