Journal of The Korea Society of Computer and Information
Vol. 17 No. 6 , June 2012

2012-17-6-3-4

Multi-Level Groupings of Minterms Using the Decimal-Valued Matrix Method

김 은 기*

십진수로 표현된 매트릭스에 의한 최소항의 다층모형 그룹화

Eungi Kim*

요 약

This paper suggests an improved method of grouping minterms based on the Decimal-Valued Matrix (DVM) method. The DVM is a novel approach to Boolean logic minimization method which was recently developed by this author. Using the minterm-based matrix layout, the method captures binary number based minterm differences in decimal number form. As a result, combinable minterms can be visually identified. Furthermore, they can be systematically processed in finding a minimized Boolean expression. Although this new matrix based approach is visual-based, the suggested method in symmetric grouping cell values can become rather messy in some cases. To alleviate this problem, the enhanced DVM method that is based on multi-level groupings of combinable minterms is presented in this paper. Overall, since the method described here provides a concise visualization of minterm groupings, it facilitates a user with more options to explore different combinable minterm groups for a given Boolean logic minimization problem.

▸ Keyword: Karnaugh Map, Quine-McCluskey, Minterm, Prime Implicant

Abstract

이 논문에서는 십진수의 매트릭스 방법 (DVM) 을 이용한 새로운 방법으로 불리언 논리를 최소화할 때 최소항을 그룹화 하여 표시하는 방법을 제안하고 있다. DVM 방법은 매트릭스 방법을 이용하여 최소항에 관한 이진수의 차이를 십진수 형태로 변환하는 과정을 거치고, 결합할 수 있는 최소항을 직접 확인할 수 있다. 십진수의 매트릭스

• 제1저자 : Eungi Kim (김은기)
• 투고일 : 2012. 03. 09, 심사일 : 2012. 04. 21, 게재확정일 : 2012. 05. 04.
* 남서울대학교 정보통신공학과(Dept. of Information Communication Engineering, NamSeoul University)

방법은 시각적 접근에 따른 새로운 매트릭스이지만, 경우에 따라 주어진 셀 값을 그룹화 하는데 있어서 도형이 복잡해지기도 하는 문제점이 있다. 이 논문은 이러한 문제점을 해결하기 위한 연구로, 십진수의 매트릭스 방법에 최소항의 다단계 그룹을 포함하는 기법을 제안하고 있다. 이 연구에서 제시하는 방법은 최소항의 그룹을 간결한 시각적인 방법으로 표현 하였으므로, 관련된 최소항을 구체적으로 파악하는 수단으로 사용할 수 있다.

▸ Keyword : 카르노맵, 퀸 매클러스키, 최소항, 후보항

I. Introduction

Logic minimization process is often difficult to comprehend in detail. Gaining insights into the logic minimization problem is not easy since the process of minimizing logic can become tedious and complex. To this end, Karnaugh map [1] is still popular today for minimizing Boolean functions due to its convenient constructable format. Although Karnaugh map is predominantly visual-based and simple to use, the major limitation is that it is only efficient for problems that are less than 5-input variables.

In contrast, the Quine-McCluskey method [2] is able to find an "exact" solution for larger input variable problems. Quine-McCluskey's basic approach is to produce all allowable maximum-size groups of minterms. In other words, the objective is to find prime implicants that can cover the minterms first.

Simple definitions need to be provided first. A minterm is a product term that contains all of the Boolean function's variables exactly once, and minterms always contain the decimal numbers of rows of the truth table in which the output is equal to 1. A prime implicant is a product term which cannot be further reduced by combining with other terms.

The Quine-McCluskey method which was the first systematic method that introduced the notion of prime implicants. Based on the generated prime implicants, finding minimum Boolean functions involve selection of a minimum set of prime implicants to cover a given Boolean function.

Although the Quine-McCluskey's method is relatively simple to implement, it lacks visual support in finding all of the required prime implicants. Tracing prime implicants using Quine-McCluskey's method is cumbersome in most cases. The reason is that it is often difficult to instantly see detailed relationships among different minterm numbers.

The Quine-McCluskey's method is cumbersome in most cases. The reason is that it is often difficult to instantly see detailed relationships among different minterm numbers. After the initial introduction of the Quine-McClukey's method, new innovative logic minimization methods based on heuristics such as binary decision diagram (BDD) [3][4] and ESPRESSO [5] appeared in the literature. Most of the algorithms that have been investigated in the last two decades or so still focus on efficiently finding a minimized solution for a given Boolean logic function. In the recent years, a wide range of methods that have been developed are either heuristic-based or variants of the Quine-McCluskey's methods [6][7][8][9][10][11][12]. While some of them claim to solve large variable problems more efficiently than the Quine-McCluskey's method, a visual aspect of logic minimization problem has been largely ignored all together.

Therefore, an alternative logic minimization method that not only finds an exact solution to the minimization problem but also visually depict relationships among different minterms is needed. The main purpose is to gain insights into the Boolean logic minimization problem. For a larger Boolean function in particular, being able to visualize the process means an entire logic related options can be better explored at a design level.

Previously, a novel approach to Boolean

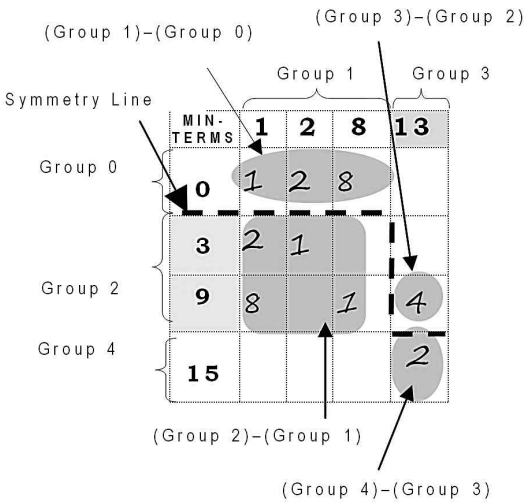
minimization problem called the Decimal-Value Matrix (DVM) method [13] was suggested by this author. This method minimizes the Boolean function based on the unique matrix layout. The matrix format represents the concise relationship between different groups of minterms. Thus, based on this minterm based matrix, the DVM method relies on visual groupings of combinable Boolean terms. Unlike Karnaugh Map, as the input variables become larger, the size of matrix extends gracefully beyond 5-input variables in the DVM method.

However, the major bottleneck of the suggested method is that in some cases too many interconnected loops could be formed as a result of connecting all of the cell values. Consequently, the graph can look cluttered and messy at times. Therefore, this paper improves the basic looping and presents a method to find higher symmetric groups for the matrix. The method presented in this paper focuses on facilitating a multi-level grouping feature. The basic idea behind using the multi-levels of grouping scheme is that the higher level of grouping can eliminate the lower level grouping details.

In order to fully understand the author's multi-level grouping procedures for the DVM method, readers are first encouraged to become familiar with Kanaugh map and Quine-McCluskey's method. Karnaugh maps and Quine-McCluskey's methods are widely covered in logic design related textbooks such as [14][15][16]. Also, readers should be familiar with basic terms and notions related to logic minimization. A brief survey of Boolean minimization methods, and the theoretical treatment of current issues are provided in [17].

II. Preliminaries

The basic approach to logic minimization using the DVM was described in [13]. This section summarizes the essential background first in order to present enhanced features in the subsequent sections. The basic layout of DVM is shown in Figure 1. The numbers at the most upper part of the row and the left most part of the column represent minterms. The odd bit-numbered groups are on the top and even bit-numbered groups are on the left side. Here, we have minterm (1,2,8,13) on the top row and (0,3,9,15) on the left side column.



Note: The above labeled groups are bit-numbered groups.
Fig 1. The DVM Layout for Boolean function

$$\sum (0,1,2,3,8,9,13,15)$$

Using the binary equivalent number, minterms are grouped based on the frequency count of 1. For example, the binary number '0011', which is equivalent to decimal minterm number 3, belongs to group 2. The reason is that there are two 1s in the binary number '0011'. This type of grouping based on the frequency count of 1s is referred as a bit-numbered grouping. In order to show each bit-numbered group is different from another, some of the corresponding matrix areas are highlighted.

Based on the Boolean input-variables, a finite number of bit-numbered groups are formed since the number of bit-numbered is proportional to the input-variables in general. As the number of input variable increases, the number of minterms to compute the prime implicants are likely to increase as well. Using the matrix, a subtracted value that

is a power of 2 (e.g., 1,2,4,8, etc.) can be recorded since each bit-numbered group intersects with next higher bit-numbered group.

The symmetry line which is shown on the figure divides each intersecting bit-number group areas. As the bit-numbered groups expand for larger input variable problems, the symmetry line will continue to expand resembling downward stair steps. For discussion purposes, from this point on we will assume there is a symmetric line even though it will not appear on the figure. The following terms are defined for clarity.

Cell coordinate is a term used to denote a pair of row and column minterms. For example, (8,0) indicates the intersecting cell of minterm 8 and 0. In Figure 1, since the matrix is in a 4x4 form, the matrix contains 16 cell coordinates. For consistency, we use a higher number first followed by a lower number.

Cell value is a positive decimal number that is power of 2 and is produced by intersecting two minterms. Specifically, a cell value can be calculated by subtracting a bit-numbered minterm value from the next higher bit-numbered minterm value. For example, a cell value of 8 can be derived from cell coordinate (8,0) where 8 and 0 are minterms. In Figure 2, minterm 0 is in bit-numbered group 0, and minterm 8 is in the next higher bit-numbered group, which is 1.

Cell coordinate value is a collective notation that indicates a cell coordinate and a cell value. For example, (3,1)2 is a cell coordinate value which indicates that minterm 3 and minterm 1 have a difference of cell value 2.

III. Levels of Symmetric Groups

As previously noted, this paper improves the notion of the DVM method by applying the multi-level grouping of minterms. Details of suggested enhancement is described in the remaining section.

In essence, we use various levels of grouping scheme to represent the basic types of prime implicants. Bit-numbered groups are the primary basis for establishing different categorical levels: Level 0 grouping connects 2 bit-numbered groups together, Level 1 connects 3 bit-numbered groups, Level 2 connects 4 bit-numbered groups together, and Level 3 connects 5 bit-numbered groups together. Any grouping of Level 3 or higher will exhibit a similar pattern in connecting bit-numbered groups.

1. Level 0 Grouping

The most primitive level in the DVM method is a Level 0 Group. In Level 0 grouping, every uncombined individual cell value belongs to a group. In Figure 1, individual cell values are not surrounded by any circles. Therefore the cell values in a Level 0 group should not belong to any symmetric groups.

2. Level 1 Grouping

In Level 1 grouping, the objective is to identify Level 1 symmetric groups. Level 1 symmetric groups can be formed by grouping the cell values from three successive bit-numbered groups. More specifically, two cells, which need to be in the same column or in the same row, crosses a symmetric line and form symmetry with another two cells in next lower or next higher bit-numbered groups. Figure 2 shows three Level 1 symmetric groups which can be formed for Boolean function $\sum$ (0,1,2,3,8,9,13,15). Cell values from successive bit-numbered groups can be combined together if the cell values form a symmetry group as shown in Figure 2.

Each symmetric group in Level 1 consists of 4 cell values. For example, in group 1, the cell value 1 and 2 are symmetric to another cell value 1 and 2 by crossing the symmetric line. For any member cell that belongs to a symmetric group, identical cell values can be located diagonally. To make certain

that cell values are visually more distinguishable from others, a cell value that belongs to more than one symmetric group are enclosed by a doubled circle.

Some cell values do not belong to a Level 1 symmetric group. As shown in Figure 2, cell coordinate values (13,9)4 and (15,13)2 are not linked to other symmetric groups since the cell values do not form symmetry with other cell values.

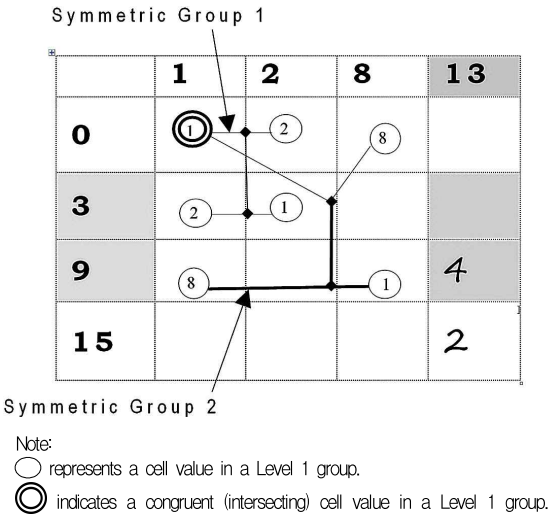


Fig 2. Level 1 and Level 0 Grouping

In the previous version of the DVM method [13], this type of grouping scheme was uniformly used for every combinable cell value. Only Level 0 and Level 1 were essentially used for the grouping of minterms. The shortcoming of this approach is that a single combinable group may involve more than just 4 cell values, and the connected lines can become quiet messy sometimes. Because of this reason, if only Level 0 and Level 1 groupings are used for the Figure 3 Boolean function

$\sum$ (0,1,2,3,4,8,9,10,11,1213), it is less convenient to see certain combinable groups after the complete graph is shown on the matrix. To remedy this representational shortcomings that deals with beyond 4 combinable cell values, we use the Level 2 and higher grouping schemes.

3. Level 2 Grouping

In Level 2 grouping, a total of 12 cell values are linked to the symmetric group. It is still easy to visually detect a Level 2 symmetric group. In Figure 3, the cell-values which are labeled as the pivot cell-value contain 1 as a numeric value. The pivot cell is located in the upper left corner of the matrix. The same pivot cell can be also found in the lower right corner. Cells within the Level 2 grouping form a symmetric group. Then, in Level 2, a symmetric group as whole forms another symmetry with another symmetric group. Highlighting the areas as in Figure 3 is not necessary since it is used merely to show particular patterns of Level 2 group.

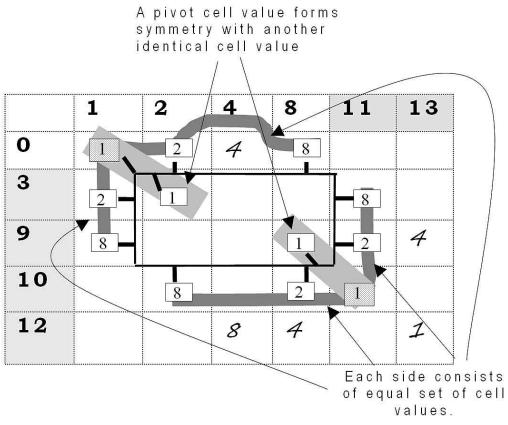


Fig. 3. Level 2 and Level 0 Grouping

For a more systematic approach in identifying symmetric groups, we can look for a common set of 3 cell values which can occur in 4 successive bit-numbered groups. The topology of cell values that form a common set is important in order to detect a valid Level 2 symmetric group. That is, a rectangular shape that represent sub-matrix can be formed by connecting four sides of the Level 2 symmetric groups. Within this sub-matrix form, some cell values form symmetry with the pivot cells. These cell values are also linked with Level 2 group

as they are determined as valid Level 2 members. Consequently, recognizing sub-symmetry pattern within this rectangular shaped sub-matrix is necessary in order to correctly identify all of the combinable values.

Suppose we have four successive bit-numbered groups (***W,X,Y,Z***) and the side of the sub-matrix is defined as ***S*** = {***a,b,c***}, where ***a***, ***b***, and ***c*** are cell values in a matrix. If a is a pivot cell value in bit-numbered group, then ***a*** sub-matrix will be formed by intersecting 4 successive bit-numbered groups.

A valid Level 2 group is formed based on meeting the following conditions :

(a) ***a*** must exist in W-X sub-matrix area AND ***a*** must also be in Y-Z sub-matrix area. They must form in a single row or in a single column.

(b) The cell values {***b,c***} as a pair appear twice in *X-Y* sub-matrix area, and both cell values line up with the pivot cell.

(c) From each pivot cell value which is located at two opposing corners, additional symmetry can be formed with another cell. For example, in Figure 3, the cell coordinate values (1,0), (2,0), (3,1), and (3,2) form a symmetric group within the Level 2 group.

(d) A pivot cell contains the lowest numeric cell value in ***S***.

(e) Each side of Level 2 symmetric group consists of ***S***. For example, Figure 3 Level 2 group contains {1,2,8} or {2,1,8} on each side the rectangular-shaped loop. The cell members ***a,b,c*** do not have to be adjacent to each other, but each side ***S*** need to contain all the cell values ***a,b,c*** according to its rectangular shape.

(f) For every row and column that has a Level 2 group link, complete members of ***S*** can be found in that column or row.

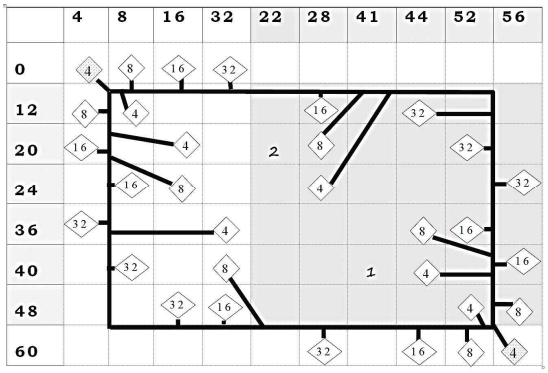
If the above conditions are met, we can state that the cell values are in a Level 3 group. For simplicity, the member cell value can be linked to any side of the rectangular shaped box.

Now the process of detecting the side and the pivot cell value is slightly more involved. Since there is no fixed location of pivot cell value, the most primitive detection strategy is to compare cell values from *W-X, X-Y, Y-Z* areas in order to find valid participating cell members. If we treat the entire horizontal cell-values as one set and the treat entire vertical cell-values as another set, then we can calculate the common cell-values between the two sets by intersecting rows and columns.

4. Level 3 Grouping

A similar pattern can be found in groups higher than level 2. Level 3 grouping has similar cell value participation as in Level 2. Level 3 grouping will comprised of a total of 32 cell values, which all of these will be linked to a single Level 3 group. Figure 4 shows an example that involves a Level 3 group with 6 input variables. The matrix shown in the figure is for a Boolean function

$\sum$ (0,4,8,12,16,20,22,24,28,32,36,40,41,44,48,52,56,60).



Note:
represents a cell value of a Level 3 group.
indicates a pivot cell of a Level 3 group.

Fig. 4. Level 3 and Level 0 Grouping

As exemplified in this figure, Level 3 grouping involves combining 5 consecutive bit-numbered groups and at the same time each column or row set consists of 4 cell values. Every participating row or

column in Level 3 group will consist of 4 equal set of cell values, hough not in a specific order. Particularly in minimization problem that has large input-variables, a number of different levels of groups will co-exist simultaneously in a matrix. To visually distinguish other levels, different shape of notation such as a diamond is being used.

In comparison to the Quine-McCluskey's method, the matrix shown on Figure 4 is much more concise and convenient since it depicts relationships between every minterm numbers. If the Quine-McCluskey's method is used for the Boolean function $\sum$ (0,4,8,12,16,20,22,24,28,32,36,40,41,44,48,52,56,60), more efforts are required to recognize the relationships among minterm values. It is much less instantaneous since some tracing of minterm values are required by a user.

IV. Minterm Covering Process

Once different levels of symmetric groups are identified, the next task is to find a minimum number of prime implicants which can cover the minterms. The process of covering minterms need to start with identifying essential prime implicants. Then, in general, we can start with the highest symmetric groups to the lowest groups. The higher the level, the more minterms the symmetric group will be able to be cover.

In Figure 5, all the highlighted groups are essential prime implicants, and their corresponding minterms that the prime implicant can cover are shown. The highest level in this figure is only 1. Since Level 1 symmetric group consists of four cell values as its members, the maximum number of minterms that a single symmetric group can cover is 4.

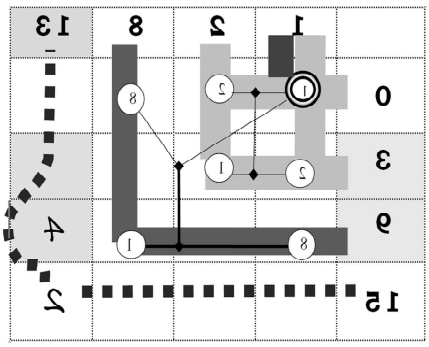


Fig 5. Minterm Covering with Level 0 and Level 1 Groups

Using the cell values which belong to the symmetric groups, minterms can be covered vertically and horizontally. Once its minterms are covered, it needs to be marked accordingly. Occasionally redundancy can be formed in above Level 1 type of grouping. Such redundancy can be removed as shown in [13].

V. Conversion of Final Selection

At the end of the process, the final prime implicant selection needs to be converted into a final Boolean expression. For Figure 5 Boolean function minimization problem $\sum$ (0,1,2,3,8,9,13,15), the following prime implicants must be selected:

(a) (1,0)1,(2,0)2,(3,1)2,(3,2)1
(b) (1,0)1,(9,1)8,(8,0)8,(9,8)1
(c) (15,13)

For each prime implication selection (a) to (c), using the binary equivalent number, the following two translation steps are required:

1. Determine the single most highest value by examining all cell values inside the parenthesis.
2. Intersect numbers outside of the parenthesis and then mark with 'x' for every numbers that have both 1's.

Since a more detailed examples of conversion

process is provided in [13], the process of conversion is rather briefly described here. For selection (b) the highest value inside the parenthesis is 9 which is equivalent to '1001'. The numbers outside of the parenthesis are 1, 8, 8, and 1. Thus, both '0001' (decimal number 1) and '1000' (decimal number 8) needs to intersect with '1001' (decimal number 9). By marking the intersected 1's with an 'x' mark, the result would be ***x00x***. A Boolean equivalent expression for ***x00x*** is $\overline{B}\overline{C}$.

After applying the same procedure for the remaining selection (a) and (c), ***00xx + x00x + 11x1*** will be produced. By translating this expression into an equivalent Boolean SOP (sum-of-product) form, we have $\overline{A}\,\overline{B} + \overline{B}\overline{C} + ABD$. This is the minimized solution for the Boolean function $\sum$ (0,1,2,3,8,9,13,15).

VI. Multi-Level Covering Process

In order to cover minterms, a proper selection of prime implicants from multi-level groups is necessary. The minterm covering process needs to be illustrated with slightly more complex example. Figure 6 is based on Boolean function $\sum$ (0,1,2,3,4,8,9,10,11,12,13). Using Level 1 and Level 2 grouping, the selected prime implicant group will be the following:

(a) (1,0)1, (2,0)2, (8,0)8, (3,1)2, (3,2)1, (11,3)8, (9,1)8, (9,8)1, (11,9)2, (10,2)8, (10,8)2, (11,10)1

(b) (4,0)4,(8,0)8,(12,4)8,(12,8)4

(c) (9,8)1,(13,9)4,(12,8)4,(13,12)1

The prime implicant group (a) is a Level 2 group. The prime implicant group (b) and (c) are Level 1 groups. In this example, all of the groups (a), (b) and (c) are essential prime implicants since every group must be selected. Essential prime implicants are ones that definitely appear in the final prime implicant selection. If (a) was selected to cover minterms on the top (1,2,8,11), and minterms on the left side (0,3,9,10), then (b) and (c) can be selected to cover the remaining minterms (4,12,13). Essential prime implicants can be easily determined by counting the number of row or columns that the each level line up with minterms.

In general, after essential prime implicants are selected, the highest group level can be selected to cover minterms since it will cover more minterms. In Figure 6, notice how the Level 1 group and the Level 2 group prime implicants appear differently in terms of covering the minterms. Particularly, the Level 2 group consists of 3 cell values in its row and column, where the Level 1 group consists of 2 in its row and column.

For the all of prime implicant groups in SOP form, Boolean expression $\overline{B} + \overline{C}\,\overline{D} + A\overline{C}$ which is a solution for the Boolean function $\sum$ (0,1,2,3,4,8,9,10,11,12,13).

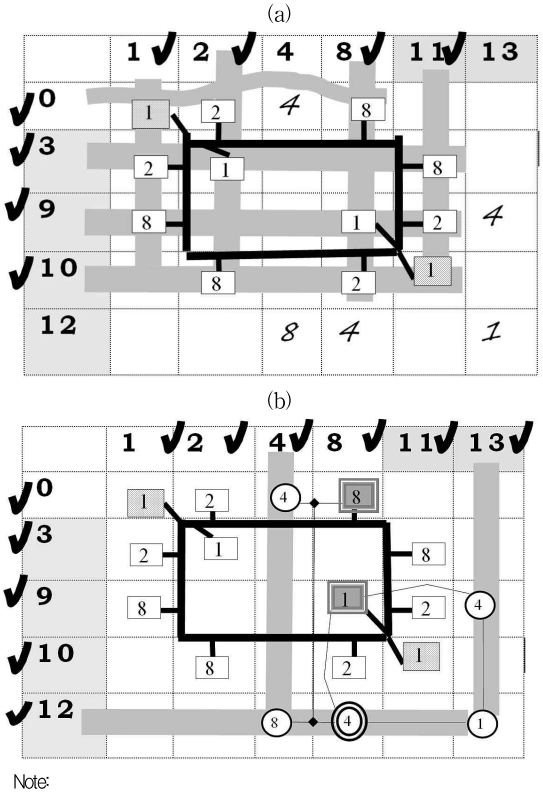


Fig 6. (a) Minterm Covering by Level 2 Group
(b) Minterm Covering by Level 1 Group

Referring back to Figure 4, the Level 3 group

covers 16 minterms. The remaining minterms 22 and 42 have to be covered by the Level 0 groups, which are (22,20) and (41,40). Because 22 and 41 can be covered solely by (22,20) and (41,40), respectively, both of these cell values are the essential prime implicants. If a minterm is covered by Level 3 essential prime implicant, there will be a total 4 cell values in one of its row or column. In this example, since most rows and columns only have 4 cell values in its row or column, the entire Level 3 group is an essential prime implicant.

If a matrix contains a valid Level 3 grouping, then the maximum number of minterms that a single Level 3 group can cover is 16. Notice that the matrix can be represented in Level 0, 1, 2, 3, or any combination of these levels.

There are other pertinent minterm covering issues such as cyclic prime implicants, which deals with more than one possible solution. This topic is widely provided in the literatures that discuss the Quine-McCluskey's method [2][14][15][16]. These issues will not be discussed in this paper, but the basic difference from the Quine-McCluskey's minterm covering strategies is that the number of participating cell values are different in the DVM method. Also, minterms which need to be covered are on the left side and on the top side. Compared to the Quine-McCluskey's method, because the minterm covering process involves different numbers of cells, for the DVM method, existing algorithms that are based on the Quine-McCluskey's minterm covering strategies have to be modified.

VII. Conclusion

Until now, this paper illustrated the techniques to enhance the DVM method, primarily focusing on the multi-level groupings of the combinable minterm. For users, such a grouping support provides necessary means to recognize inter-related minterm patterns for a larger input variable problem. Because each level provides a different level of abstraction, a birds-eye view of entire process is possible with this method. The approach described in this paper could bring increased productivity since a logic based design can easily become highly complex. Besides requiring a software based on the method, to become a viable tool, more algorithms that are based on the DVM method are also needed. In particular, detecting multiple levels of symmetric groups and more minterm covering strategies using the DVM method should be investigated so that the minterm covering process can be further optimized.

References

[1] M. Karnaugh, "A Map Method for Synthesis of Combinational Logic Circuits," Transactions of the AIEE, Communications and Electronics, Vol. 72, pp. 593-599, 1953.

[2] E. J. McCluskey, "Minimization of Boolean functions," Bell System Tech. Journal, Vol. 35, No. 5, pp. 1417 - 1444, 1956.

[3] R. E. Bryant, "Graph-based algorithms for Boolean functions manipulation," IEEE Trans. on Computers, C-35, No 8, pp. 677-692, Aug., 1986.

[4] O. Coudert and J. C. Madre, "Implicit and incremental computation of primes and essential primes of Boolean functions," Proc. 29th DAC, CA, USA, pp. 36-39, June 1992.

[5] R. L. Rudell, Logic Synthesis for VLSI Design, Ph.D. Dissertation, UCB/ERL M89/49, 1989.

[6] F. Basciftci, "Simplification of Single-Output Boolean Functions by Exact Direct Cover Algorithm Based on Cube Algebra," EUROCON, pp. 427-431, Sept. 2007.

[7] A. Dusa, "Enhancing Quine-McCluskey," 2007. http://www.compasss.org/files/WPfiles/Dusa2007a.pdf (Accessed: 9 February 2012).

[8] A. Dusa, " A Mathematical Approach to the

Boolean Minimization Problem," Quality & Quantity, Vol. 44, No. 1, pp. 99-113. 2010.

[9] P. Fiser, P. Rucky, and I. Vanova, "Fast Boolean Minimizer for Completely Specified Functions", Proc. 11th IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop 2008, Bratislava, SK, pp. 122-127.

[10] J. Hlavicka and P. Fiser. "BOOM-A Heuristic Boolean Minimizer," Computers and Artificial Intelligence Vol. 22, No 1. pp. 19-51, 2003.

[11] P. W. C. Prasad, A. Beg, and A. K. Singh, "Effect of Quine-McCluskey Simplification on Boolean Space Complexity, IEEE Proceeding 2009 Conference on Innovative Technologies in Intelligent Systems and Industrial Applications, pp. 165-170, Monash University, July, 2009.

[12] D. Toman and P. Fiser. "A SOP Minimizer for Logic Functions Described by Many Product Terms Based on Ternary Trees," In Proceedings of 9th International Workshop on Boolean Problems (IWSBP), 2010.

[13] E. Kim. "A Visual-Based Logic Minimization Method," Journal of the Korea Industrial Information Systems Research, Vol. 16, No 5, Dec., 2011.

[14] B. Holdsworth and C. Wood, Digital Logic Design, 4th ed., Newnes, 2002.

[15] C. H. Roth, Jr., Fundamentals of Logic Design, 5th ed., Thomson Engineering, 2004.

[16] P. K. Lala, Principles of Modern Digital Design, John Wiley & Sons, Inc., Hoboken, NJ, USA., 2006.

[17] C. Umans, T. Villa, and A.L. Sangiovanni-Vincentelli, "Complexity of Two-Level Logic Minimization," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems. Vol. 25, No. 7, pp. 1230 - 1246, 2006.

저 자 소 개

Eungi Kim (김은기)

1991: Indiana Univ. of Pennsylvania 학사 (Computer Science)

1993: Illinois State University 석사 (Applied Computer Science)

2012: University of North Texas (Information Science) 박사학위취득(예정)

현 재: 남서울대학교 정보통신공학과 외국인교수

관심분야: Info. Retrieval, Logic Design

Email: eungikim68@daum.net