A Visual-Based Logic Minimization Method

Eungi Kim*

Abstract In many instances a concise form of logic is often required for building today's complex systems. The method described in this paper can be used for a wide range of industrial applications that requires Boolean type of logic minimization. Unlike some of the previous logic minimization methods, the proposed method can be used to better gain insights into the logic minimization process. Based on the decimal valued matrix, the method described here can be used to find an exact minimized solution for a given Boolean function. It is a visual based method that primarily relies on grouping the cell values within the matrix. At the same time, the method is systematic to the extent that it can also be computerized. Constructing the matrix to visualize a logic minimization problem should be relatively easy for the most part, particularly if the computer-generated graphs are accompanied.

Key Words : Boolean algebra, logic, minterm, Quine-McCluskey, Karnaugh Map

1. Introduction

Today logic is often used extensively in one form or another for building complex applications. Minimization of Boolean expression is essential for implementing applications such as circuit design. A shorter symbolic representation also means more efficiency with fewer errors. Since Boole's [1] classic work on logic and Shannon's [10] work on the application of Boolean algebra, two influential techniques were developed: Karnaugh map [4] and Quine-McCluskey [6]. These two methods are still widely used for finding minimized logic expressions. The former Karnaugh map is a popular graphical method, but is not suitable beyond 5-input variables. In contrast, the Quine-McCluskey's method can be conveniently used for solving beyond 6-input variable minimization expressions.

Numerous algorithms have been suggested subsequently. See references in [2, 3, 5, 7, 9] for some of the methods in dealing with Boolean logic minimization. Some of these methods proved to be effective in finding minimized Boolean expressions. However, none of these methods were developed with the intent to visualize a logic minimization process.

In this paper, the matrix that supports visual aid in detecting minimal expression is described in detail. The author calls the proposed method as the 'Decimal-Valued Matrix' method. Unlike Karnaugh map, it can handle beyond 5-input variables. Compared to the Quine-McClukey's method, the selection of prime implicants is visual based and eliminates the need for establishing a separate chart to examine the potential Boolean logic terms. A prime implicant is commonly referred to as a minimum product term that cannot be combined with any other product term to eliminate a variable [6]. For discussion on prime implicant chart see also [8].

As a result of using the Decimal-Valued Matrix

method, users can gain insights into the logic minimization problems. Otherwise, it can be used to verify the Quine-McCluskey's method. Most important of all, recognizing certain patterns associated with the logic problems should become more intuitive for users.

2. Minterms in Respect to Numerical Values

Before presenting the Decimal Valued Matrix method, the notion of minterm in respect to numerical values is briefly introduced here. This foundational concept is widely covered in literature such as [4] [10]. In logic minimization related problems, a literal is usually defined as a single variable within a term. For example, the letter $\overline{A}$ may stand for a literal that is in the complemented form while the letter A may stand for an uncomplemented form of a literal. The combination of literals in a product form is referred as a minterm. Further, a product of literals requires the logical operation 'AND'. In Table 1, each variable, in a complemented or uncomplemented form, is used exactly only once in the minterm column.

The truth table shows the 24 possible combinations for 4 variables. Here, for n number of variables, the list will be expanded to 2n possible combinations. Using this form of representation, a truth table can represent any type of Boolean function using minterms.We may write the above listed possible minterms as m1, m2, m3, etc. Using a truth, if m3 is true, the following condition will result: A=0, B=0, C=1, and D=1. A function of a given problem can be represented by using only minterms. From the above list, if only minterm m3 and m5 are represented with the value of 1 as according to a truth table and if all other have the value of 0, then the Boolean function can be expressed differently. For example, the following $Y=\overline{A}\,\overline{B}\,C\,D+\overline{A}\,B\,\overline{C}\,D$ can be simply denoted as $F(ABCD)=\Sigma$ (m3,m5). The above Boolean expression is often said to be a sum of product (SOP) expression since every term contains all input variables. A non-standard SOP is a case where a term does not contain all literals.

<Table 1> Minterms for 4 Variable Minimization Problem

Minterm Decimal Value	Binary Value	# of 1's in the Binary Form	Minterms
0	0000	0	$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$
1	0001	1	$\overline{A}\,\overline{B}\,\overline{C}\,D$
2	0010	1	$\overline{A}\,\overline{B}\,C\,\overline{D}$
3	0011	2	$\overline{A}\,\overline{B}\,C\,D$
4	0100	1	$\overline{A}\,B\,\overline{C}\,\overline{D}$
5	0101	2	$\overline{A}\,B\,\overline{C}\,D$
6	0110	2	$\overline{A}\,B\,C\overline{D}$
7	0101	2	$\overline{A}\,B\,\overline{C}\,D$
8	1000	1	$A\,\overline{B}\,\overline{C}\,\overline{D}$
9	1001	2	$A\,\overline{B}\,\overline{C}\,D$
10	1010	2	$A\,\overline{B}\,C\,\overline{D}$
11	1011	3	$A\,\overline{B}\,C\,D$
12	1100	2	$A\,B\,\overline{C}\,\overline{D}$
13	1101	3	$A\,B\,\overline{C}\,D$
14	1110	3	$A\,B\,C\,\overline{D}$
15	1111	4	$A\,B\,C\,D$

As for another example, a Boolean expression $Y=\overline{A}+B\,C$ involves three variables. Even though it is in minimized form, this is not a standard SOP. This expression can be converted to a standard SOP expression using an expansion method. That is, $Y=\overline{A}+B\,C$ can be converted into the following expression: $Y=\overline{A}\,\overline{B}\,\overline{C}+\overline{A}\,\overline{B}\,C+\overline{A}\,B\,\overline{C}+\overline{A}\,B\,C+A\,B\,C$.

The latter logic expression can be also derived using the laws of Boolean algebra, but alternatively it can be expanded more easily using a series of binary numbers. For a variable A, a binary value 1 can be used. Also, a binary value 0 can be used for the variable $\overline{A}$. The expansion rule counts the total number of variables and expands the minterm based on combinatorial binary value. Considering the Boolean function $Y=\overline{A}+B\,C$, the first term $\overline{A}$ can be expanded as a series of binary numbers 000 001 010 011. The variable $\overline{A}$ alone means that any one of the following combination can occur: $\overline{A}\,\overline{B}\,\overline{C}$, $\overline{A}\,\overline{B}\,C$,

$\overline{A}B\overline{C}$, or $\overline{A}BC$. Translating the binary number to the decimal number, the result would be 0, 1, 2 and 3. The second term BC can be expanded as 011 and 111. Thus, the term BC can be expanded to $\overline{A}BC$ or ABC. Using the above decimal based number scheme, the minterm BC can be denoted as m3 and m7. Combining values, we have m0, m1, m2, m3, and m7 as decimal based minterm values. The minterms represent the Boolean expression $Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + \overline{A}\,BC + ABC$.

In turn, these values are equivalent to the Boolean expression $Y = \overline{A} + BC$, which is our original minterm expression. Also, note that a logic function with n variables has 2n minterms. In the above example, the 4-variable input function has a total of 16 minterms. The Quine-McCluskey method is based on binary number bit patterns that arise among the minterms. As shown in Table 1, the number of 1's in the binary form that ranges from 0 to 4. The method starts with grouping minterms based on the number of 1's. As an example, for the binary values 0001, 0010, 0100, and 1000, there is exactly a single "1" in every binary form of number. As a consequence, group 1 consists of minterm 2, 8, and 16. In the Quine-McCluskey's method the value of each group is compared with each other using a binary form of minterm numbers. Overall, in this proposed method, the minterm patterns are observed with decimal numbers instead of
binary numbers.

3. Illustration of the Minimization Procedure

The following section describes the Decimal Valued Matrix method in detail. Let's consider a simple example with 4-variable logic minimization problem Σ (0,2,4,8,9,10,12).

Note that the letter m, which stands for minterm, was omitted for simplicity. The maximum minterm value, which is 15, implies that this is a 4-variable minimization problem. Using the previously mentioned cell matrix layout scheme, Figure 1 can be produced.

3.1 Step I. Layout a Matrix with Minterm Values

The first step in finding a minimized Boolean expression is to construct a matrix as shown in Figure 1. The matrix needs to layout minterm values according to the matrix construction rule. Figure 1 shows the minterms based on counting the 1's and the corresponding cell values after performing the calculation. The line that partitions the matrix into different sections is said to be a symmetry line. Whether a symmetry line is shown on the matrix or not, readers should assume that there is a symmetry line that divides the matrix into different sections.

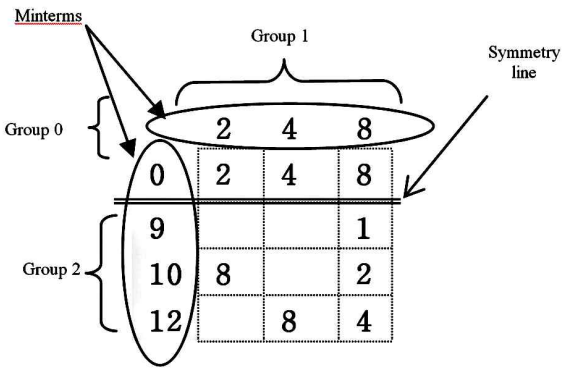
Also, the ordering of group and the layout of these groups on the matrix is significant in this approach. Same as the Quine-McCluskey method, sequential numbered groups can be formed based on number of 1's that exist in each binary based minterm. Also, the sequence of comparing group is same as the Quine-McCluskey's method. In this method, the top most row and the left most column are used for labeling the minterms. Notice that the minterms are ordered in an alternating sequence. The top side minterms are based on every odd number of 1's while the left side minterm are based on every even number of 1's. Therefore, the top side of matrix will be Group 1, Group 3, Group 5, etc. Meanwhile, the left side of the matrix will be Group 0, Group 2, Group 4, etc. We refer to this type of grouping scheme which is based on the frequency count of 1's in the binary form of minterms as a bit-numbered group. Additionally, obtaining these cell values is described in the next section.

3.2 Step II. Record the Cell-Coordinate Values

Once the matrix is laid out with different partitions, the cell-coordinate value needs to be calculated and

recorded. In Figure 1, each cell can represent a value where the corresponding row minterm and column minterm intersect. To avoid confusion, three terms (cell coordinate, cell value, cell coordinate value) are defined in the following.

Cell-coordinate refers to an individual cell element which is represented as an intersecting pair of minterm row and column number. For example, the cell coordinate (8,0) indicates that the cell intersects with the minterm value 8 and minterm value 0. Cell coordinate is a minterm pair (M_{i+1}, M_i) where i=l to z. Here, l is the lowest index number of the bit-numbered group while z is the highest bit group value. The entire matrix will contain cell coordinates $\{(M_{i+1}, M_i),(M_{i+2}, M_{i+1})\ldots (M_z, M_{z-1})\}$.



<Figure 1> The Matrix with Subtracted Values

Cell value, which is denoted as C, is a difference between two cell-coordinate elements and must be the power of 2 (e.g., 1,2,4,8, etc.). For example, C =8 for the cell coordinate (8,0). C is a subtracted value which can be obtained by $M_{i+1} - M_i$. C is only valid if it is a power of 2 and M_{i+1} must be greater than M_i. In this case, the cell value which is obtained by subtracting 0 from 8 is valid since $M_{i+1} > M_i$ and it is a power of 2. Starting with the lowest bit-numbered group i, we compare M to the next bit-numbered group M_{i+1} in order to obtain the cell value C. When comparing and subtracting minterm values for the cell values, if the next higher bit-numbered group is lower than the current bit-numbered group, we can skip performing the subtraction operation

Combining the above definitions, a cell coordinate value is defined as (M_{i+1}, M_i)C. For example, (8,0)8 is a cell coordinate value which represents the cell coordinate (8,0) and the cell value 8.

3.3 Step III. Group Cells Based on Symmetry

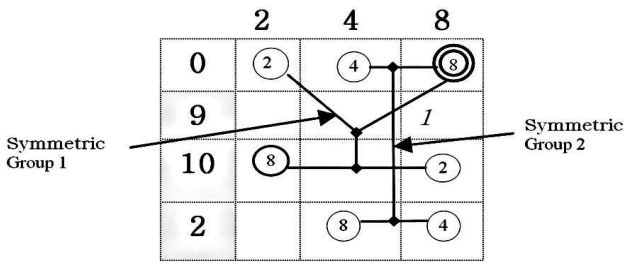
The next step is to group the cells based on the notion of symmetry. A *symmetric group* consists of four cell values in such a way that two of cell values from one bit-numbered group are symmetric to its adjacent bit-numbered group. As Figure 2 indicate, two symmetric groups can be identified with the above minterm function Σ(0,2,4,8,9,10,12). These are

Symmetric Group 1: (2,0)2,(8,0)8,(10,2)8, (10,8)2
Symmetric Group 2: (4,0)4, (8,0)8, (12,4)8, (12,8)4

The cell values of circled elements are perfectly diagonal to each other by crossing the symmetry line. In this matrix, there are two types of cell values: ones with small circle, and ones without any circle. The cells with a small circle are the type of cell values which can be combined with another cell value. The cell values without the circles are the cell values that cannot be combined as a symmetric group. For example, 1 from cell coordinate (9,8) is left alone as it can not belong to any symmetric group.

As a result of grouping the cell values, some cells can have more than one circle. This type of circle indicates the cell values is a congruent as it belongs to more than one symmetric group. This is a junction point where other groups share the same cell values together. It is possible to have multiple circles sometimes since the cell value can be a member of multiple symmetric groups. For

example, because the cell coordinate (8,0) is a congruent cell coordinate, the cell value in this case is enclosed by a double circle.



<Figure 2> The Matrix after Grouping

3.4 Step IV. Cover Minterms with Cell Values

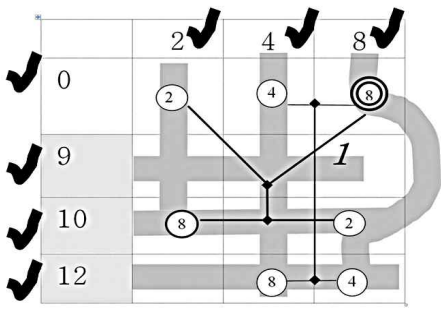
After the matrix is constructed, we need to find an optimal way to cover minterms by using the prime implicants. Since more than one prime implicant can cover a minterm, we start the process by locating *essential prime implicants* that was not enclosed by any circles. Non-circled elements do not necessary mean they are essential prime implicant so close visual inspection is necessary for each non-circled element. The reason for start covering a non-circled cell value first is that a non-circled cell values is easier to determine whether it is an essential implicant or not.

Sometimes minterms can be covered more than one way when aligned with cell values. If competing grouped elements which have equal number of cells that can cover a minterm, there could be more than one solution. A minterm needs to be examined vertically as well as horizontally since some minterms are listed on the top while other minterms are listed on the left side. Figure 3 shows the overall minterm covering process for the above minimization problem.

The process of covering minterms can start with selecting *essential* prime implicants. Minterm 9 can be covered by the cell-coordinate value (9,8)1. Here the cell value 1 must be used and no other cell value can cover minterm 9. To this end, it is an essential prime implicant. Essential prime implicants can be identified by recognizing the only one cell values in the row or the column. Each time a prime implicant is selected to cover a minterm, the corresponding minterms should be marked so that only remaining minterms can be examined to complete the procedure.

Next, since there are two symmetric groups, both groups need to be inspected in order to cover the rest of the minterms. Figure 3 shows the symmetric groups, which are prime implicants, covering the minterms. Minterm 8 and 9 were already covered by cell value 1. Even though double covering can occur, we can use the symmetric group 1 and 2 in an attempt to cover the rest of the minterms. In reference to Figure 2 and 3, the symmetric group 1 will cover minterm 0, 2, 8, and 10. Then symmetric group 2 will cover 0, 2, 4, and 8. A trial and error process may be required to cover every minterm with minimum number of prime implicants.

In general, if a non-circled cell value is not an essential prime implicant, then we often need to select a symmetric group rather than a non-circled cell values in order to obtain small number of literals at the end. If none of cell values can cover a minterm, then the minterm itself is essential, and it must be translated and included as a part of final Boolean expression at the end.



<Figure 3> An Example Showing Minterm Covering Process

Also, in order to determine the best possible prime implicants, redundant (non-essential) symmetric groups should be identified. In Figure 3, there is no redundant group so an example of this type of group is provided later.

3.5 Step V. Convert the Prime Implicant Selection

Once all the prime implicants in the form of cell coordinate values are selected for covering minterms, the final step is to transform cell coordinate values to a Boolean expression. For given number of cell coordinate value, we need to select the highest minterm value *M* regardless the number of cell coordinate values. Thus, we select the highest number among the inside of the parenthesis. *Cell value C*, which is outside of the parenthesis, will be also used for the final translation.

In Figure 3, if a cell coordinate value (9,8)1 is selected as a prime implicant, then the highest number inside the parenthesis is 9, or 1001 in binary form. In addition, since *C* is 1, the binary number 0001 will be used. Now examining the highest *M* with *C*, 1001 intersects with 0001 at the point of right most bit. By marking the intersection with 'x', 100x would be produced as a result. Here, 100x is equivalent to the Boolean term $A\overline{B}\overline{C}$.

For the prime implicant symmetric group (2,0)2 ,(8,0)8,(10,2)8,(10,8)2, 10 is the highest number among the group. Now each cell value C needs to be intersected with the decimal value 10 since 10 is the highest M. The values of C are 2 and 8. The decimal number 10 is 1010 in binary form; the values of C are 0010 and 1000 in binary form Thus, we have

1010
0010 (mark with x where 1 intersect)
1000 (mark with x where 1 intersect)

---

x0x0

The equivalent Boolean term for x0x0 is $\overline{B}\overline{D}$. For the prime implicant (4,0)4, (8,0)8, (12,4)8 and (12,8)4, we apply the same procedure. 12, which is 1100 in binary, is the highest *M*. Since 4 and 8 are the value of *C*, 1000 and 0100 must be used. By intersecting these two cell values with the decimal value 12, the result would be xx00. xx00 is equivalent to $\overline{C}\overline{D}$.

Therefore, the following will result:

(9,8)1 → 100x → $A\overline{B}\overline{C}$

(2,0)2,(8,0)8,(10,2)8,(10,8)2 → x0x0 → $\overline{B}\overline{D}$

(4,0)4,(8,0)8,(12,4)8,(12,8)4 → xx00 → $\overline{C}\overline{D}$

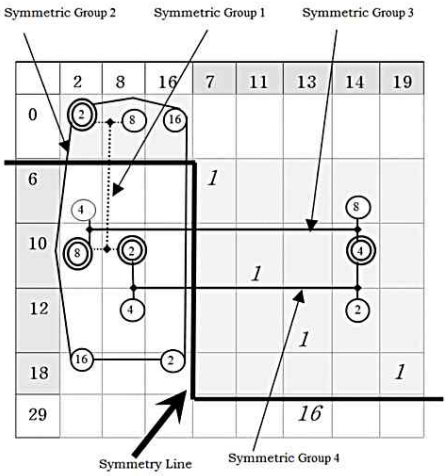
Combining the above Boolean terms together, we have $Y = A\overline{B}\overline{C} + \overline{B}\overline{D} + \overline{C}\overline{D}$ as a final minimized Boolean function $F(A,B,C,D) = \Sigma(0,2,4,8,9,10,12)$.

4. Identifying Redundant Symmetric Groups

For an illustration of steps to identify a redundant symmetric group, let us consider a logic minimization function $F(A,B,C,D,E) = \Sigma(0,2,6,7,8,10,11,12,13,14,16,18,19,29)$. This problem involves input 5-input variables since the decimal values of minterm ranges from 0 to 29. Also, it implies the bit-numbered group which will range of 0 to 5. After obtaining subtracted values by comparing each lower bit-numbered group to the next higher bit-numbered group, Figure 4 would be produced.

In this example, since (2,0), (10,2), (10,8), and (14,10) are congruent cell coordinates, double circles are used for each of these cell values. Three out of four cell coordinates in the symmetric group 1 have two circles surrounding each of the cell value. Double circled cell coordinates are (2,0), (10,2), and (10,8). The symmetric group 1 can be eliminated from the potential prime implicant list that can convert the minterm since other connected symmetric groupings would become dominant in covering the minterms. Since selecting non-redundant symmetric

groups as a prime implicant may lead to an optimal solution in covering minterms, redundant symmetric groups must be avoided. The rule of thumb is that for a given symmetric group, if more than 2 of its cell values have multiple circles, then the symmetric group is redundant. Since redundant groupings can be visually identified, the process of selecting implicants for covering can become somewhat simpler after removing the redundant groups.

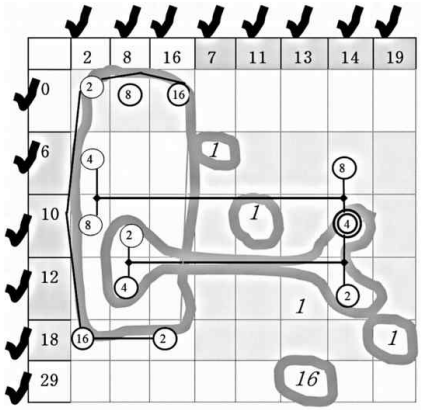


<Figure 4> A Grouping Example for
Σ (0,2,6,7,8,10,11,12,13,14,16,18,19,29)

In this figure, four symmetric groups can be identified. These are
Symmetric Group 1: (2,0),(8,0),(10,2),(10,8)
Symmetric Group 2: (2,0),(18,2),(16,0),(18,16)
Symmetric Group 3: (6,2),(10,2),(14,6),(14,10)
Symmetric Group 4: (10,8),(12,8),(14,10),(14,12)

The covering process can start with selecting essential prime implicants. In this case, the cell-coordinate values (7,6)1, (11,10)1, (19,18)1, and (29,13)16 are essential implicants. Here, these single cell-coordinate values do not have any circles. Also, the cell values are the only values which are aligned diagonally or vertically with the corresponding minterms. Therefore, the cell-coordinate value (13,12)1 can be left out since it can be (29,13)16 and the symmetric group 4 can cover this cell-coordinate.

The remaining symmetric group 2, 3 and 4 must be inspected so that remaining minterms can be covered. However, in covering the minterms, the symmetric group 1 and 3 are not required. The reason for this is that minterm 6 and 10 were already covered by the cell-coordinate (7,6), (11,10), and minterm 14 can be covered by just selecting the symmetric group 4.



<Figure 5> Minterm Covering for
Σ (0,2,6,7,8,10,11,12,13,14,16,18,19,29)

After the entire process, Figure 5 will be produced. Every minterm is covered as the figure indicates. The prime implicants that are selected to cover the minterms are indicated on the matrix. Selecting the following prime implicants are the most effective ways to cover the minterms:
Symmetric Group 2: (2,0)2,(18,2)16,(16,0)16,(18,16)2
Symmetric Group 4: (10,8)2,(12,8)4,(14,10)4,(14,12)2
Cell-coordinate Values:
(7,6)1,(11,10)1,(19,18)1,(29,13)16

These are the final prime implicant selection which needs to be converted. Table 2 shows the final translating step. Finally, the minimized expression will result in the following: $\overline{B}\overline{C}\overline{E} + \overline{A}B\overline{E} + \overline{A}\overline{B}CD + \overline{A}B\overline{C}D + A\overline{B}\overline{C}D + BC\overline{D}E$.

<Table 2> Translating Prime Implicants to Boolean Terms

Selected Prime Implicants	Highest M	C values (x)	Binary-based Term	Boolean Term
(2,0)2,(18,2)16,(16,0)16,(18,16)2	18	2,16	x00x0	$\overline{B}\overline{C}\overline{E}$
(10,8)2,(12,8)4,(14,10)4,(14,12)2	14	2,4	01xx0	$\overline{A}B\overline{E}$
(7,6)1	7	1	0011x	$\overline{A}\overline{B}CD$
(11,10)1	11	1	0101x	$\overline{A}B\overline{C}D$
(19,18)1	19	1	1001x	$A\overline{B}\overline{C}D$
(29,13)16	29	16	x1101	$BC\overline{D}E$

5. Paired Symmetric Groups

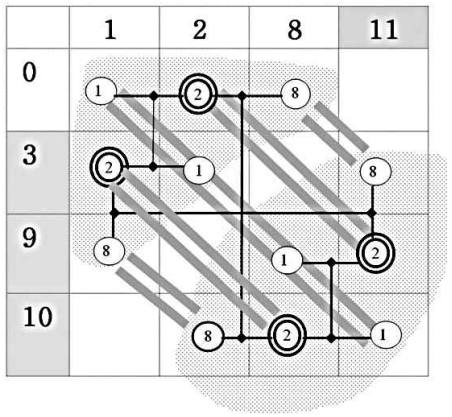
Sometimes a symmetric group itself can "pair off" with another symmetric group. In such case, all of the cell-coordinates are treated as a single symmetric group. We refer this type of group as a *paired symmetric group*. Now Consider a Boolean minimization problem

Σ(0,1,2,3,8,9,10,11). See Figure 6 for the matrix construction and its corresponding cell values.

In this figure, all of the cell-coordinate values (1,0)1, (2,0)2, (8,0), (3,1)2, (3,2)1, (9,1)8, (11,3)8, (9,8)1, (11,9)2, (10,2)8, (10,8)2, and (11,10)1 are connected in some fashion. In the previous example, every symmetric group was treated separately. Unlike the previous example, the symmetric connected groups collectively form a larger symmetric group and form a paired symmetric group. The left upper region as a whole are reflected to right lower corner. In this case, these cell-coordinate values need to be treated as if they are combined together as a single prime implicant.

In the above case, all of its cell-coordinate values are considered as a single prime implicant. Consequently, the symmetric groups must be treated as a single group in covering the minterms.

Since identifying a large number of cell values can form a paired symmetric group, visually identifying the paired symmetric groups can be difficult sometimes. Another way to identify paired symmetric groups is to examine its cell values diagonally after an initial grouping. As shown in the figure, all of the cell numbers line up diagonally.



<Figure 6> A Paired Symmetric Group

Now we need to translate the result to an equivalent Boolean expression. The Decimal value 11 is the *M*, and the values of *C* would be: 1, 2, 8, 2, 1, 8, 8, 1, 2, 8, 2, and 1. After removing the duplicate values, we are left with 1, 2, and 8. These are values that need to be aligned with 11 and marked x where they align. Thus, after 1011 is being intersected by the values 0001, 0010, and 1000, we would have x0xx. Since 0 refers to the second variable of the term, the final minimized term is $\overline{B}$. Therefore, the output minimized expression can be written as Y=$\overline{B}$.

6. Unspecified (Don't Care) Minterms

Sometimes unspecified, or "don't care", minterms of a function can occur. This is a situation where certain input specifications have no bearing on the

output of a function. In most cases, this situation can be utilized to be shorter expression. For example, consider a Boolean function

$F(ABCD) = \Sigma(0,2,4,8,9,10,12) + d(13)$.

Except for the notation *d(13),* this Boolean function is the same as the previous Boolean function which we used for Figure 2. In this case, *d(13)* denotes the don't care minterm value 13. If a don't care condition with minterms is specified, we list the don't care minterm and calculated values in the matrix, and mark the don't care values. This step is required in order to correctly select prime implicants. However, during the minterm covering step as shown in Figure 7, don't care minterms do not have to be covered by any prime implicant.

The following will be produced as a result of selecting the prime implicants.

(9,8)1, (13,9)4, (12,8)4, (13,12)1 → *1x0x* → $A\overline{C}$

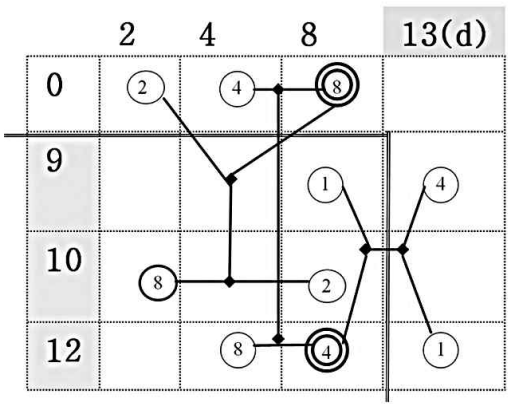
(2,0)2, (8,0)8, (10,2)8, (10,8)2 → *x0x0* → $\overline{B}\,\overline{D}$

(4,0)4, (8,0)8, (12,4)8, (12,8)4 → *xx00* → $\overline{C}\,\overline{D}$

Although the symmetric group can cover the minterm, it is not required to cover minterm 13 since this is a don't care minterm. Combining the above terms together, we have

$Y = A\overline{C} + \overline{B}\,\overline{D} + \overline{C}\,\overline{D}$ as a final minimized Boolean function

$F(ABCD) = \Sigma(0,2,4,8,9,10,12) + d(13)$.



<Figure 7> The Matrix for the Minterm $\Sigma(0,2,4,8,9,10,12)+d(13)$

7. Summary of the Procedure

The following steps summarize the procedures which were described earlier.

Step 1. Layout a matrix with minterm values. Minterms are organized based on the bit-numbered group. The matrix contains minterms in such way that the top side minterm is based on every odd number of 1's while the left side minterm is based on every even number of 1's. Starting from the lowest possible number, alternating sequence of minterm values has to be labeled on the top most and left most of line of the matrix.

Step 2. Record the cell values. Every cell value needs to be calculated by subtracting the lower bit-numbered minterm group from the next higher bit-numbered minterm group. The values are recorded only if the numbers are the power of 2.

Step 3. Group the cell values based on symmetry. Based on every four members as one group, some cell values will form a symmetric group. Each cell values must be circled when it belongs to a symmetric group. A congruent cell value can be enclosed with more than one circle. Every symmetric group needs to be identified in this step. Then, any redundant symmetric group from the graph can be removed.

Step 4. Identify paired symmetric groups, if any. Some symmetric groups itself can pair off with another symmetric group. These groups, if any, need to be identified so that they can be treated as a single group.

Step 5. Select prime implicants to cover minterms. Starting with cell values which has no circles, determine whether these are essential prime implicants by examining its row and column. Once these essential prime implicants are identified and minterms that they cover are clearly marked. The remaining minterms should be covered by symmetric groups or other non-circled cell values.

If none of the prime implicant can cover a minterm, then the minterm itself is a term that must be included in the final expression. A trial and error process may be required to find a minimal minterm covering prime implicants. Remove any prime implicants that fail to cover any minterm. Also, the don't care minterms must be included and labeled in the matrix construction, but the minterms do not have to be covered during the minterm covering process.

Step 6. Convert the selected prime implicant values in to a final Boolean expression. Collectively, the highest *M* among the cell-coordinate and each of the cell value *C* are used to obtain the final Boolean term. Some binary bits are marked with 'x' after intersecting binary number of 1's from each cell value with the highest minterm *M*.

8. Conclusion

In this paper, a relatively easy, visual-based method is presented for finding minimized logic expressions. The Decimal-Valued Matrix method can be used to visualize well beyond 5-input variable logic minimization problems. Since generating graphs and performing calculations by hand seem impractical for logic problems that involve large input variables, computer generated solutions based on this method are definitely more suitable. In such a case, the user would be able to visually manipulate a logic pattern and gain further insights into to the Boolean logic minimization process. Therefore, a software based implementation of the Decimal-Valued Matrix and additional research are needed in order to advance this approach.

References

[1] G. Boole, *An Investigation of the Laws of Thought*, London: Macmillan, at the Internet Archive, 1854.

[2] R.E. Bryant, "Graph based Algorithms for Boolean Functions Manipulation," *IEEE Trans.Computers*, vol. C 35, pp. 677-691, Aug. 1986.

[3] O. Coudert, "Doing Two Level Logic Minimization 100 Times Faster," in *Proceeding SODA '95 Proceedings of the sixth annual ACM SIAM symposium on Discrete algorithms*.

[4] M. Karnaugh, "A map method for synthesis of combinational logic circuits," *Transactions of the AIEE, Communications and Electronics*, vol 72, pp. 593-599, 1953.

[5] H. Mahmoud, "A New Method for Two Level Logic Minimization," in *46th IEEE International Midwest Symposium on Circuits and Systems*, Magdy Bayoumi (Chairman), Cairo, Egypt, December 2003.

[6] E. J. McCluskey, "Minimization of Booleanfunctions," *Bell System Tech. Journal*,Vol. 35, No. 5, pp. 1417-1444, 1956.

[7] P. C. McGeer, J. Sanghavi, R.K. Brayton,A.L.Sangiovanni Vincentelli, "ESPRESSOSIGNATURE: A New Exact Minimizer for Logic Functions," *IEEE Transactions on VLSI*, vol. 1, no. 4, pp. 432-440, December 1993.

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

[9] R. L. Rudell, "Multiple Valued Logic Minimization for PLA Synthesis," *Memorandum No. UCB/ERL M86-65* (Berkeley), 1986.

[10] C. E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits," *Trans. AIEE*. pp 713-723, 1957.

김 은 기 (Eungi Kim)

- 정회원
- Indiana Univ. of Pennsylvania (BS) (Computer Science)
- Illinois State University (MS) (Applied Computer Science)
- University of North Texas (Ph.D Candidate in Information Science)
- 남서울 대학교 정보통신공학과 외국인교수
- 관심분야: Information Retrieval, Digital Systems