Generation of Prime Implicants of Logic Function by Three-Branch-Tree Expansion Method Using Matrix Operation

Kimio GOTO*, Kee-Seng CHIN**, Takashi ITO**
and Xiao-Ping LING*

Abstract

The two programs for the old IONEX Method and the Three-Branch-Tree Expansion Method Using Matrix Operation in this paper were writtern in C language. They were run on the SUN SPARC station 5 computer and the computation times were measured. As a result, it was proven the method using matrix operation is not inferior to the old IONEX method in terms of computation time and measurable range. The computation time has a tendency to improve for the higher minterm densities beyond fifty percent and the computation times were able to be measured to thirty or more variables.

1. Introduction

For generating all prime implicants of logic function, we have proposed a kind of Three-Branch-Tree Expansion method, such as IONEX (the abbreviation of Improved Consensus Expansion) method[1] and MULES (the abbreviation of Multi-Level-Synthesis) method[2]. The former is the Three Branch-Tree Expansion method, itself. The latter is the method where more performance improvement in terms of computation time was brought by the addition of the subdivisions on original function and those syntheses to this former method.

In this paper, we return back to the first Three-Branch-Tree Expansion method such as IONEX and study again the improvement of that method, itself. For the Three-Branch-Tree Expansion method, originally, the minterm numbers' strings were intended or in order to deal with the inputs shown by the sum-of-products form, that is, the sum-of-minterms form. But for the method described in this paper, except the sum-of-minterms form, the arbitrary products of literals are intended for in order to deal with the inputs.

In order to make such dealing with inputs easier, the matrix operation is used to implement the Tree-Branch-Tree Expansion method.

Therefore, in this paper, first, the adjacency confirmation operation is described as the characteristic of the matrix operation. Second, the algorithm of this Three-Branch-Tree Expansion method using matrix operation and its example are shown. Third, the conputation results and those discussions are described.

Received, 1995. 9. 16

* 情報工学科
** 情報工学科専攻修士学生

2. Three-Branch-Tree Expansion Method Using Matrix Operation

In this section, first, the adjacency confirmation operation is described. Then, the algorithm is explained.

2.1 Adjacency Confirmation Operation

The given function is shown as the sum-of-products form. These products heve been arranged as the row elements in the matrix. These row elements are used in order to ascertain the adjacency between each other row element included in the own matrix or in the defferent matrices and then in order to generate a new row element from the two adjacent row elements. Such adjacency confirmation operations are performed between two corresponding digits of two row elements. Each of these digits has been set up by the ternary value, 0, 1, or 2. This setting-up is determined in accordance with what status the literal corresponding to its digit has, namely, negative, affirmative, or nonexistent. Therefore, these adjacency confirmation operations are performed according to the ternary value operation having the special operation rule.

The operation table of adjacency confirmation operation $\triangledown$ and conversion table from ternary number to binary number are shown by Table 1 and Table 2, respectively. One example is shown in the folliwing equations by using these tables.

$$[1020]\triangledown\begin{bmatrix}1111\\1201\end{bmatrix}=\begin{bmatrix}1212\\1002\end{bmatrix}\Rightarrow[1002] \qquad (1)$$

In Equation (1), 1212 is deleted because this row element contains two new 2's. And 1002 is adopted because this row element contains only one 2. In fact, such operation of Equation (1) is excuted by the binary number operations according to Table 2, for example.

Table 1. Operation Table of Adjacency Confirmation Operation $\triangledown$

$\triangledown$	0	1	2
0	0	2	0
1	2	1	1
2	0	1	2

Table 2. Binary Number Expression of Numbers shown in Table 1

Numbers in Table 1	0	1	2
Identifying Bit	0	1	0
Presentation Bit	1	1	0

2.2 Algorithm

The algorithm of this method is as follows:

[Step 1] First, the order of literals in any product of given sum-of-products form function is arranged from smaller to larger according to the order of variable-subscripts. Second, each literal is rewritten to 0 if the literal is the negation of the corresponding variable, to 1 if it is the affirmative of the corresponding variable, or to 2 if it does not exist. Then, each number sequence corresponding to each product is arranged in each row of matrix M_{or}.

[Step 2] One variable x_1 is chosen. From the preceeding matrix M_{or}. we extract every row where the value of digit corresponding to x_1 is 0, and put them into a new matrix M_{or}. called the negation term matrix. In the same way, from the matrix M_{or}. we extract every remaining row where the value of digit corresponding to x_i is 1, and put them into a new matrix M_a called the affirmative term matrix. By comparing every row in matrix M_n and matrix M_a with each other row of its own matrix, the adjacency confirmation operations (described afterwards) war performed.

[Step 3] Between all row elements of negation matrix M_n and all row elements of affirmative matrix M_a the adjacency confirmation operations are performed and all new row

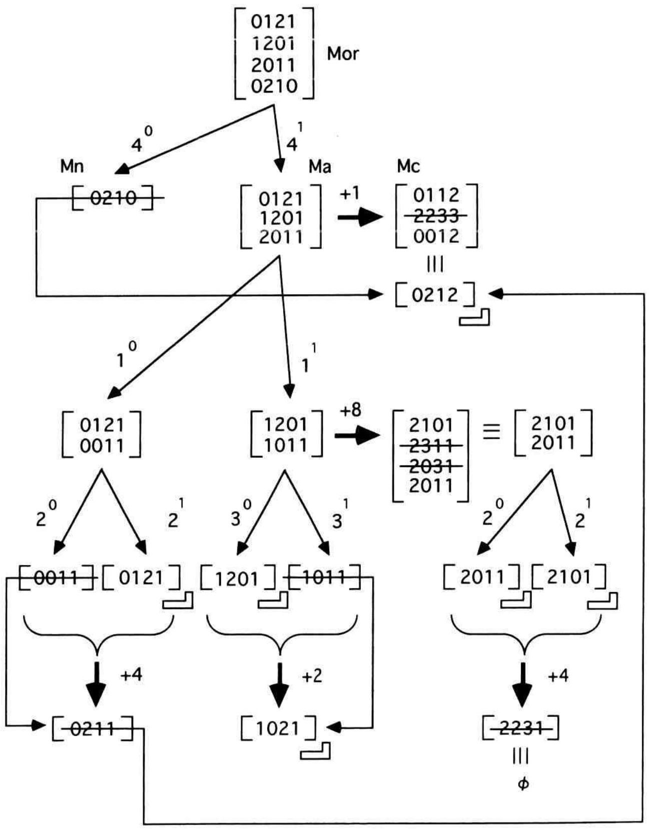

Fig. 1. An example of application of Three-Branch-Tree Expansion Method Using Matrix Operation.

elements are generated from all pairs of adjacent row elements. The new matrix M_c is constructed by these new row elements. This matrix is called the consensus term matrix.
[**Step 4**] If each matrix of these three matrices. M_n, M_a and M_c has only one row element, the matrix is a candidate of prime implicant. If it is so, it has to be checked whether this matrix is covered by the other prime implicant obtained at the other upper expansion level or by the consensus term matrix at the same expansion level.
[**Step 5**] This expansions are to be continued until all matrix has only one row element.

$$f=\overline{x1}x2x4+x1\overline{x3}x4+\overline{x2}x3x4+\overline{x1}x3\overline{x4} \tag{2}$$

3. Computation Results and Those Discussions

The two programs for the IONEX method and the Three-Branch-Tree Expansion method using Matrix Operation in this paper were written in C language. They were run on the SUN SPARC station 5 computer (Memory 64 MB) by inputting the minterm numbers generated randomly for each minterm density in the range from 10% to 90% for each variable in the range from 4 variables to 15 variables and in the appropriate short range chosen inside the range from 0.00005% to 20% for 16 variables to 31 variables.

The average time of ten measurements was adopted as the conputation time taken to generate all prime implicants for each minterm density. As a result, it was proven the method using matrix operation in this paper is not inferior to the old IONEX method in terms of computation time by Fig. 2 and Fig. 3. In addition, from these two figures, for the

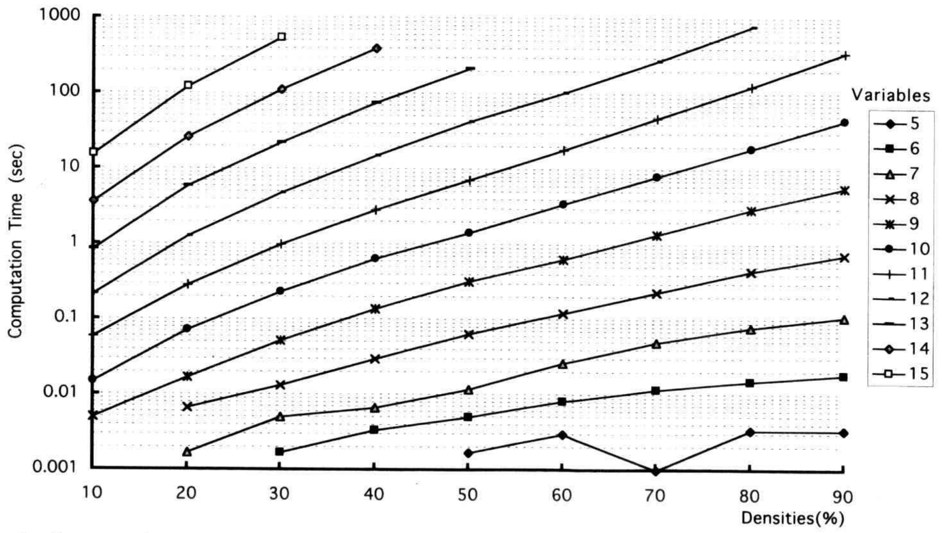

Fig. 2. Computation time taken in generating all prime implicants by the original Three-Branch-Tree Expansion method (IONEX method). Number of variables, $n=5\sim15$, Densities=Number of Minterms/2^n.

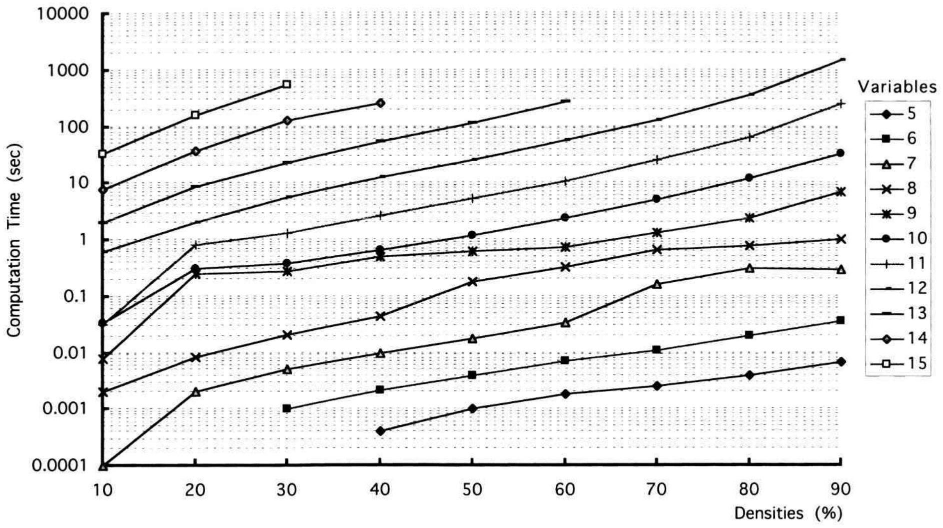

Fig. 3. Computation time taken in generating all prime implicants by the method of this paper. Number of variables, $n=5\sim15$.

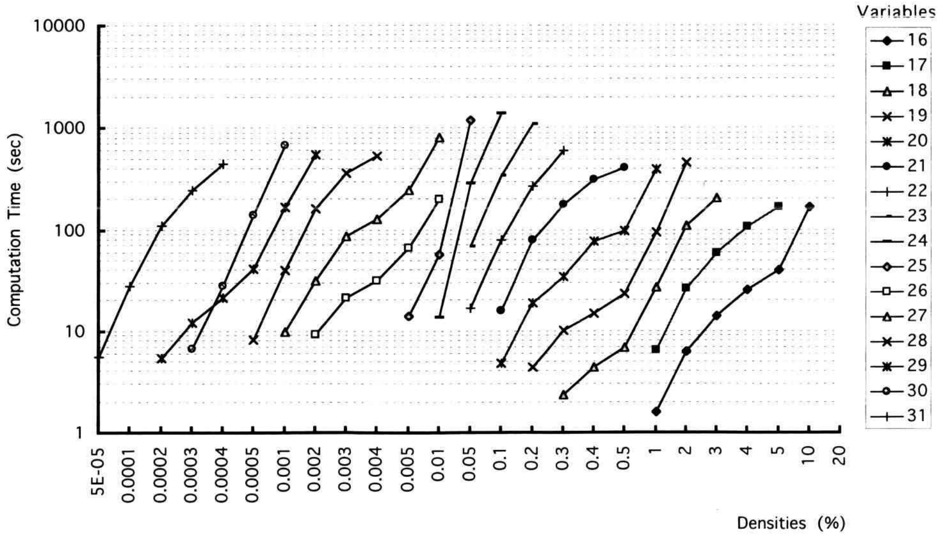

Fig. 4. Computation time taken in generating all prime implicants by the method of this paper. Number of variables, $n=16\sim31$.

minterm densities beyond fifty percent and in terms of measureble variable range, the method of using matrix operation is a little superior to the old IONEX method. Then, by this method, the measurement was possible even beyond thirty variables. This was proven by the result of Fig. 4.

4. Conclusions

The Three-Branch-Tree Expansion method using Matrix Operation in this paper is not inferior to the original three-branch-tree expansion method. And it is certain the method in this paper has a lot of resistance to the function with the higher variables.

References

[1] GOTO, K. : "Five Methods for Simplification of Logic Function and Comparison of their Characteristics", Proceedings of 1990 IEEE International Symposium on Circuits and Systems, Vol. 2 of 4, p. 1122-1125, May 1990.

[2] GOTO, K., MENJO, M., TATSUMI, H., LING X.P. and TAKAHASHI S. : "Improvement of Prime Implicants Generation of Logic Function by Multi-Level-Synthesis Method", Proceedings of the IEEE TENCON '94, Vol. 2, pp. 938-942, Aug. 22-26, 1994.