Generation of All Prime Implicants by Using BDD (Binary Decision Diagrams)

Kimio GOTO*, Naoya KANEKAWA**, Takashi ITO***, Masaki MASUMIZU***,
Hisayuki TATSUMI* and Xiao-ping LING*

Abstract

This paper describes a reduction of computer operating time for generating prime implicants by adding BDD (Binary Decision Diagrams) method to the usual CONSENSUS method. It was confirmed by running C language programs on workstation AS4015.

1. Introduction

Generating prime implicants of logic function by using the CONSENSUS method after BDD (Binary Decision Diagrams) Expansion has finished (this method is called BDD EXP method hereafter) was studied and compared with the single CONSENSUS method. Computer operating times for the two C-language programs for both methods were compared by being run on the SUN workstation AS4015.

2. Algorithm of BDD EXP[1~3]

The flow chart of BDD EXP method is shown in Fig. 1. The important features of this method cover Step 5 through Step 10, that is, mainly Prediction of the Order of Expansion Variables, BDD Expansion and its Reduction, Generation of Expansion Literal Sequences, etc. The details are described as follows:

2.1 Generation of Truth Table

When an original logic function f is expressed by the sum-of-products form, such combination of variable values that changes any minterm included in some product term of this original function into a logic value of 1 can be converted easily into the decimal number corresponding to the minterm number.

In the same way, when an original logic function f is expressed by the product-of-sums form, such combination of variable values as changes any maxterm included in some sum term of this original function into a logic value of 0 can be converted easlly into the decimal number corresponding to the maxterm number.

Received, 1994. 9. 20.

* 情報工学科
** 情報工学科卒研生
*** 情報工学専攻修士学生

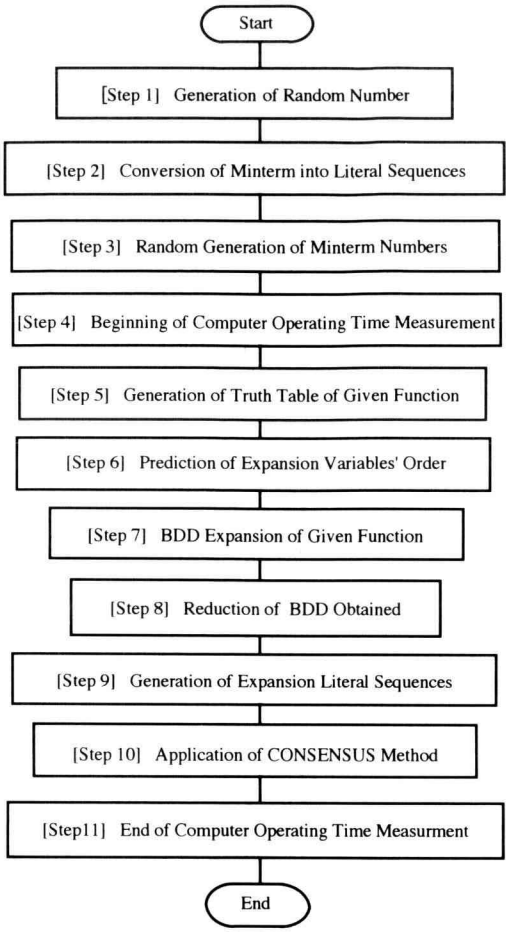


Fig. 1. Flow Chart for Realizing BDD EXP Method

Therefore, the truth table of an original logic function f is easily generated.

2.2 Prediction of the Order of Expansion Variables[4]

In the BDD Expansion, it is essential to predict in advance what order of expansion variables gives the fewest nodes, because it makes the expansion simpler and quicker. For this prediction we used the quasi-truth table method described afterwards.

After the expansion for the i'th level was performed by using the expansion variable xi (where i has a value of 1 to n, and n is the numbr of variables), there can be some nodes containing the expansion variable xi and some terminals such as 1-terminals or 0-terminals in the i'th expansion level. How these combinations of nodes and terminals at the i'th level are created is dependent on which variable is chosen as the expansion variable xi, which is necessary to move the expansion from the i'th level to the immediately successive (i+1)'th level. And the total number of nodes and terminals at the i'th level is $2^{(i-1)}$. Therefore, if

we count either of the number of nodes or the number of terminals, we can then obtain the other. In truth, the count of terminals is much easier than the count of nodes. Therefore, we try to find 0-terminals and 1-terminals, by using the truth table and quasi-truth table. Here, the new truth table generated under the following conditions is called the i'th quasi-truth table: (1) the expansion variable $x\boldsymbol{i}$ to move the expansion from the i'th level to the $(i+1)$'th level has been excluded from input variables in this table. And (2) for some combination of input variables having no $x\boldsymbol{i}$, its output function $f\boldsymbol{X}, \boldsymbol{i}$ can contain $x\boldsymbol{i}$, single 0, or single 1 in this table. Such i'th quasi-truth table is generated from the $(i+1)$'th quasi-truth table as follows:

Two values of function $f\boldsymbol{X}, \boldsymbol{i+1}$ are compared with each other for two input sequence values adjacent to each other with $x\boldsymbol{i}$ in the $(i+1)$'th quasi-truth table. Concretely, a value of function $f\boldsymbol{X}, \boldsymbol{i+1}$ for an input sequence value $AO=(a\mathbf{1}, a\mathbf{2}, \ldots, x\boldsymbol{i}=0, \ldots, a\boldsymbol{n})$, is compared with one for $Al=(a\mathbf{1}, a\mathbf{2}, \ldots, x\boldsymbol{i}=1, \ldots, \mathrm{an})$ adjacent to $A0$ and decimal number $2^{(i-1)}$ away from A0 (where $a\boldsymbol{j}$ shows either $x\boldsymbol{j}=0$ or $x\boldsymbol{j}=1$ and j is not i). If the function $f\boldsymbol{x}, \boldsymbol{i}+1$ has the same value $\boldsymbol{d}$ for these two input sequence values, the function $f\boldsymbol{x}, \boldsymbol{i}$ becomes $\boldsymbol{d}$ (where $\boldsymbol{d}$ is either 0 or 1) for an input sequence value having no $a\boldsymbol{i}$, namely $(a\mathbf{1}, a\mathbf{2}, \ldots, a\boldsymbol{i}-\mathbf{1}, a\boldsymbol{i}+\mathbf{1}, \ldots, a\boldsymbol{n})$. This $\boldsymbol{d}$ is put into the i'th quasi-truth table. Then, by counting the number of $\boldsymbol{d}$'s in the i'th quasi-truth table, we can obtain the sum of the number of 0-terminals and the number of 1-terminals in the i'th level. Such generation of quasi-truth table is begun with the n'th quasi-truth table, that is, the truth table of original logic function f. Fig. 2 shows an example of quasi-truth table.

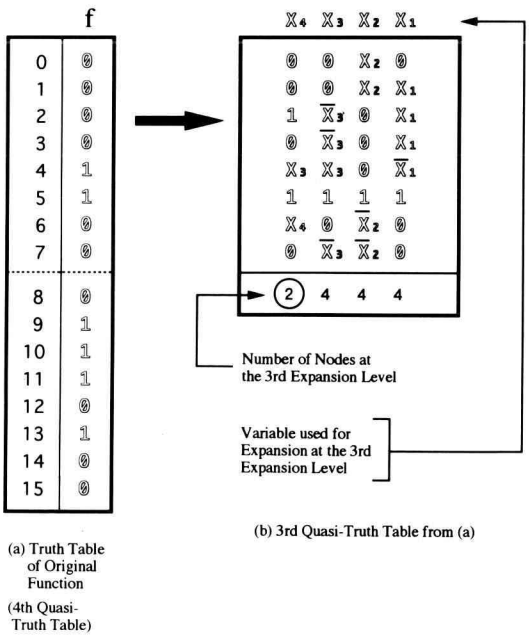


Fig. 2. Prediction of the Order of Expansion Variables

After the same countings are tried for all n variables at the same i'th level, a variable generating the fewest nodes is chosen as the most suitable expansion variable xi to move expansion from the i'th level to the $(i+1)$'th level. This procedure is repeated from the last n'th level to the first level. Such quasi-truth table continues to decrease the size by one half at each level but to complicated itself much more. This is avoided by managing the quasi-truth table generation techniques on the program.

2.3 BDD Expansion

The BDD expansion is accomplished by using Shannon's expansion according to the predicted order of expansion variables as follows:

First, a set of product terms of original function f is assigned to the root.

Second, if 0 (or 1) is substituted for the expansion variable xi, all elements (that is, all product terms) containing xi in the set for the parent node are extinguished (or lose only their literal xi's) and all elements containing $\sim xi$(*) lose only their literal $\sim xi$'s (or are extinguished). As a result, the set of updated elements is put into the left node (or the right node). Especially, when all elements are 0's (or at least one or more elements are 1's) at some node, the node becomes a 0-terminal (or a 1-terminal). This procedure is repeated according to the predicted order of expansion variables until no nodes appear.

In addition, at this BDD expansion, the following informations connecting with each node must be stored in some memory area:

(1) the individual node number information,

(2) the expansion information of its node, and

(3) the information representing the function contents of its node, that is, all product terms of the function at its node.

Here, the expansion information includes the node number of parent node, the expansion variable used at the parent node, and a logic value, 0 or 1, substituted for this expansion variable.

2.4 Reduction of BDD

After BDD expansion is completed, this BDD is reduced. If two child nodes (or two terminals) generated by some parent node are identical, only the left child node is connected directly to the grandparent node, and the parent node is deleted along with the right child node.

2.5 Generation of Expansion Literal Sequences

The expansion literal sequences are generated by concatenating each literal on each expansion path while going back along the path to the root after starting with the 1-terminals. When the logic value 1 (or 0) has been given to the expansion variable xi, the literal becomes the affirmative (or the negation) of xi. This concatenation is performed

(Comment*: $\sim xi$ means the negation of xi.)

easily by referring to the above-mentioned informations connecting with each node already memorized.

2.6 Application of the CONSENSUS Method

The prime implicants are generated by applying CONSENSUS method to these new literal sequences.

3. Results and Discussions

For the BDD EXP method and the CONSENSUS method, two corresponding C language programs are prepared. By running these two programs on the workstation AS4015 (made by Toshiba as OEM product of SUN-Microsystems), both the number of generated prime implicants and the computer operating times were measured and then compared. These results are shown in Fig. 3 and Fig. 4, respectively. In these measurements, the number of variables n given in the original logic function was intended for four variables through the ten or more variables. These two methods are applicable to any sum-of-products type of logic functions, and the decimal minterm numbers generated in the form of random numbers were applied to computer inputs. The abscissas of figures 3 and 4 show the minterm density which means the ratio of the number of minterms to the number of all possible minterms 2^n in Karnaugh map. After these measurements were repeated ten times for an input consisting

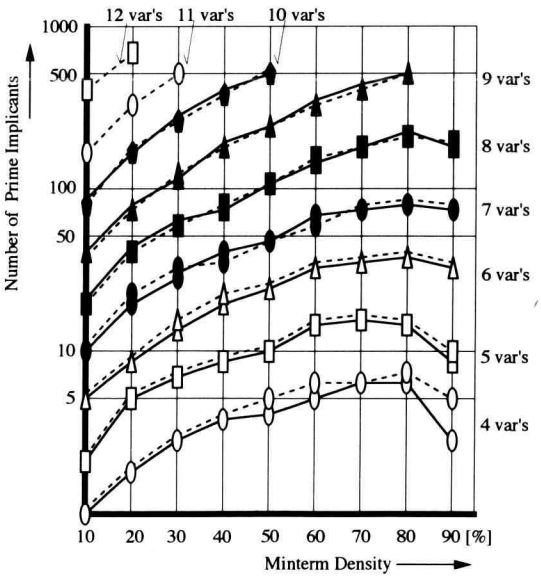
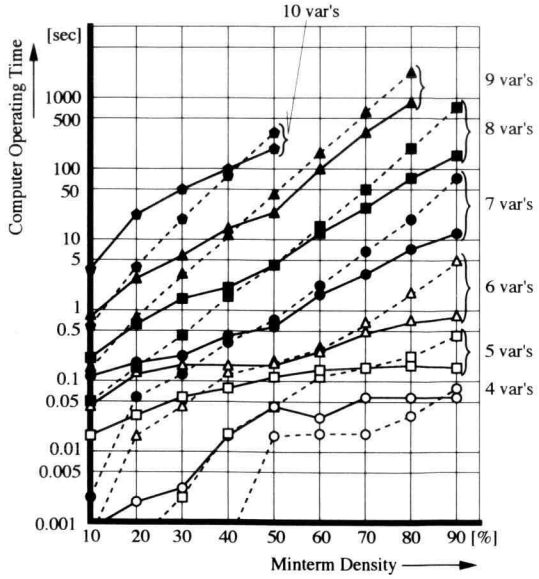


Fig. 3. Relations between Minterm Densities and the Number of Prime Implicants



——BDD EXP method
····CONSENSUS method
var's; Abbreviation of "variables"

Fig. 4. Relations between Minterm Densities and Computer Operating Times

of minterm numbers corresponding to each minterm density and each variable, the results obtained were averaged.

From Fig. 3, it is proved that there is little difference between the two methods concerning the number of prime implicants generated.

From Fig. 4 the following results are obtained for the computer operating time:

(1) At higher minterm densities than the branching point of 40-45% for more than six variables, the computer operating time for the BDD EXP method was smaller than the computer operating time for the CONSENSUS method, but at lower minterm densities than this branching point for more than six variables, operating time became greater.

(2) For the minterm densities beyond this branching point, the larger the minterm densities, the better the improvements of computer operating time of the BDD EXP method to the CONSENSUS method, for the same variables.

(3) For greater than six variables, the improvements of computer operating times were almost equal, and they were almost 1/5.5-1/6.

4. Conclusion

According to this study, the following things were affirmed:

(1) It was proved that the prediction of the order of expansion variables is effectively realized by using the quasi-truth table.

(2) It was confirmed the BDD EXP method is more effective to generate prime implicants than the CONSENSUS method at the larger densities and higher variables.

References

1) Sheldon B. Akers, "Binary Decision Diagrams," IEEE Trans. Comput., vol. C-27, pp. 509-516, 1978.
2) Randal E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," IEEE Trans. Comput. vol. C-35, pp. 677-691, 1978.
3) R.K. Brayton, G.D. Hachtel, and A.L. Sangiovanni-Vincentelli, "Multilevel Logic Synthesis," Proc. IEEE, vol. 78, pp. 264-300, 1990.
4) Steven J. Friedman and Kenneth J. Supowit, "Finding the Optimal Variable Ordering for Binary Decision Diagrams," IEEE Trans. Comput., vol. C-39, pp. 710-713, 1990.