Quantum Boolean Circuits Construction Using Tabulation Method

Chin-Yung Lu, Shiou-An Wang, and Sy-Yen Kuo

Department of Electrical Engineering and Graduate Institute of Electronic Engineering
National Taiwan University, Taipei, sykuo@cc.ee.ntu.edu.tw, Taiwan

Abstract — **The Tabulation method can be used with a computer to simplify Boolean logic functions with up to 6 or more variables, especially with a large number of variables. For the various applications, their circuits usually are complex and we must simplify the circuit design to the best of our ability. In this paper, we present an algorithm that can efficiently simplify a quantum Boolean circuit with an arbitrary number of input variables by using the tabulation method. In terms of the space consumption, we use only one auxiliary qubit as the output qubit, and keep all the input qubits unchanged.**

Index Terms — **Logic minimization, quantum Boolean circuit, tabulation method.**

I. Introduction

Quantum computing is one of the most rapidly expanding research fields recently. Over the last few years, several quantum algorithms, such as Shor's quantum factoring [1] and Grover's fast database search algorithm [2] have emerged. They are much faster than their best classical counterparts. To implement a quantum computer, we need to construct quantum Boolean circuits [3] which consist of quantum gates. Unlike conventional AND-OR-NOT-based circuits, quantum Boolean circuits are based on NOT, CN, and CCN gates. Although with different building blocks, they can still be synthesized by the classical AND, XOR, and NOT functions.

Tabulation method [4] can be used to reduce Boolean functions, especially for a large number of variables. The concept of tabulation method, however, is very important for the following two reasons:

1) The method can be programmed to be used as a tool for quantum Boolean functions minimization.
2) The method is a promising technique for reducing quantum Boolean functions

II. Tabulation Method

The classic tabulation method consists of two separate steps, determining prime implicants and finding a minimum set of prime implicants, but our algorithm doesn't need to apply the second step. We just find the term with the maximum number of minterms at every cycle partially like the first step of the classical method.

Definition 1: Let a minterm m with n variables be $x_1x_2...x_n$, where x_i represents the i-th variable and x_i=0 or 1.

In the quantum Boolean function, using a tabulation method to simplify quantum logic circuits is not easy because the XOR functions are substituted for the OR functions. It makes a little bit difference between the classical and the quantum logic circuits by using the tabulation method to simplify logic functions.

Tabulation method usually starts with a listing of the specified minterms and groups the minterms according to the number of 1's. The simplified term which contains minterms with the relation of logic minimization can be obtained to combine minterms or terms by using the theorems $xy \oplus x\bar{y} = x$ or $x\bar{y}z \oplus xyz = x(y \oplus z)$, where x represents a product of literals, y and z are a single variable, and $\oplus$ notation indicates XOR function.

Definition 2: Let m_1 and m_2 be two minterms with n variables. If they differ in exactly one variable, the i-th variable, then they can be combined to yield a new term $x_1x_2...x_{i-1}*x_{i+1}...x_n$, where the '*' notation means that the i-th variable is eliminated.

Definition 3: Let m_1 and m_2 be two minterms with n variables. If they differ in two variables, the i-th and j-th variables, then they can be combined to yield a new term $x_1x_2...x_{i-1}\text{-}x_{i+1}...x_{j-1}\text{-}x_{j+1}...x_n$, where the two '-' notations mean that the two corresponding variables are combined to form an XOR function.

For example, two terms $x_1x_2\bar{x}_3x_4$ and $\bar{x}_1x_2x_3x_4$ which differ in two variables, x_1 and x_3, are combined to form a new term $x_2x_4(x_1 \oplus x_3)$.

III. Combination Rules

Now we introduce a set of combination rules to simplify a circuit. By applying these rules, we can get a simplified quantum Boolean circuit.

Rule 1: If two terms including minterms differ in the i-th variable and are equal in other variables, then they can be combined by eliminating the i-th variable to form a term.

In Fig. 1, because two terms differ in only one variable, they can be combined by eliminating the variable x_1. This simplified term have four minterms 0, 4, 8, and 12. Note

0-7803-8536-5/04/$20.00 ©2004 IEEE

that (0,4) 0*00 indicates a term with two minterms 0 and 4, where (0,4) is the decimal representation of combined format of two minterms given in parenthesis and 0*00 is the binary representation of the term.

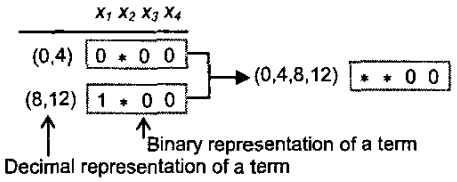

Fig. 1. An example for Combination rule 1.

Rule 2: If two terms differ in two variables, the *i*-th and *j*-th variables, and one of these two variables of a term is * and another term also has only one * in the different one of the two variables, then they can be combined by eliminating the *i*-th and *j*-th variables.

In Fig. 2, the term (0,4,6,8,12,14) ***0 has six minterms given in parenthesis and two non-minterms, 2 and 10.

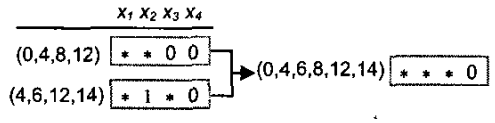

Fig. 2. An example for Combination rule 2.

Rule 3: If two terms differ in *k* variables and these two terms have only one * notation in every different variable, then they can be combined by eliminating these *k* variables.

The term (0,4,5,8,12) **0* has five minterms given in parenthesis and three non-minterms, 1, 9, and 13, in Fig. 3.

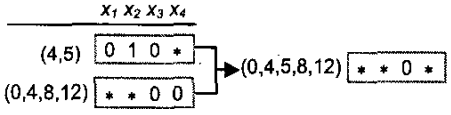

Fig. 3. An example for Combination rule 3.

Rule 4: If two terms differ in only one variable, the *i*-th variable, and have – notations in the same variables, then they can be combined by eliminating the *i*-th variable.

Its algebraic equivalent is $\bar{x}_1\bar{x}_3(\overline{x_2 \oplus x_4}) \oplus \bar{x}_1 x_3(x_2 \oplus x_4) = \bar{x}_1(\overline{x_3 \oplus x_2 \oplus x_4})$ as shown in Fig. 4.

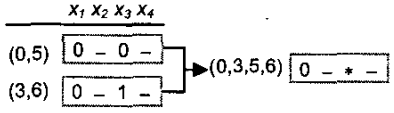

Fig. 4. An example for Combination rule 4.

Rule 5: If two terms differ in only one variable, the *i*-th variable, excluding the variables which are – notations, then they can be combined by eliminating the *i*-th variable.

The term (8,13,14) 1-*- has three minterms given in parenthesis and one non-minterm, 11, as shown in Fig. 5.

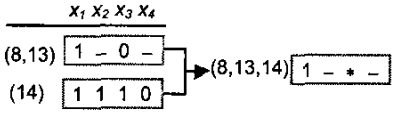

Fig. 5. An example for Combination rule 5.

IV. Algorithm

A. Grouping the minterms:

In order to find the most simplified term with the maximum number of minterms, all possible pairs of minterms should be compared and combined. To reduce the required number of comparisons, the minterms in binary are sorted into groups according to the number of 1's. Therefore,

$$F(x_1, x_2, x_3, x_4) = \sum(0,4,5,6,8,12,14)$$

is grouped as shown in Column 0 of Fig. 6.

In Column 0, the term in group 0 has zero 1's, the terms in group 1 have one 1, and so on.

B. AND function generation:

First we compare every term in group 0 with all of the terms in group 1 in Column 0 of Fig. 6. Two terms which differ in only one variable can be combined to eliminate the different variable by using Rule 1, which yields a new simplified term. The resulting terms are listed in Column 1.

Because the comparison of group 0 with groups 2 and 3 is unnecessary, we proceed to compare terms in groups 1 and 2. Notation v in Fig. 6 indicates this term has been selected to combine with others.

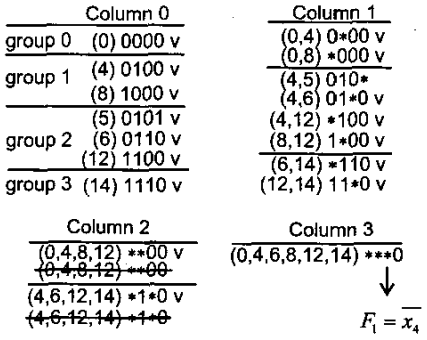

Fig. 6. AND function generation at the first cycle.

Then terms in the first group in Column 1 need only be combined with terms in the second group which have * notation in the corresponding place by using Rule 1. The resulting terms are listed in Column 2.

Finally, we find term (0,4,8,12) can be combined with term (4,6,12,14) by using Rule 2 as shown in Column 3. Then we need to calculate the number of minterms in this term.

C. XOR function generation:

In the quantum Boolean circuit, the XOR function is as important as the AND function. Thus, we need to simultaneously think of the AND and XOR functions in the deduction of the expression, then this kind of the method can really simplify quantum Boolean circuits.

Firstly, the minterms must be partitioned into two parts which have even and odd number of minterms respectively as shown in Column 0 of Fig. 7. Then we can find all possible pairs of XOR functions in these two parts respectively. Because two terms with a relation of XOR function differ in two variables, they can be combined into a new term, for example term (0,5) 0-0-. The resulting terms are listed in Column 1.

In Column 1 of Fig. 7, we collect the terms with two – notations in the corresponding places together into Column 2 by using Rule 4. Then term (0,6,8,14) can be combined with term (5) to yield a new term (0,5,6,8,14). Now we can calculate the number of minterms in this term.

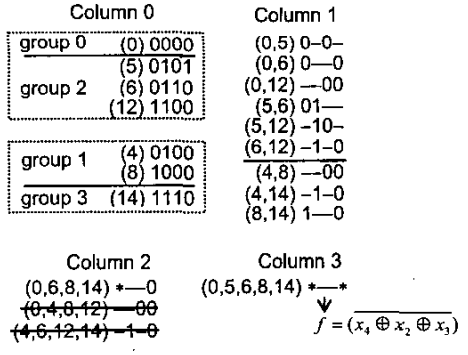

Fig. 7. XOR function generation at the first cycle.

D. Algorithm:

Suppose that the original truth table have n variables. The algorithm is described as follows:

Step 1. Calculate the number of minterms. If the number is over 2^{n-1}, then change minterms into non-minterms and non-minterms into minterms.

Step 2. Group the minterms.

Step 3. Apply the AND function generation.

Step 4. Apply the XOR function generation.

Step 5. Select the most simplified term from Steps 3 and 4. According to the select term, we can get the corresponding Boolean function and change the minterms in the term into non-minterms and the non-minterms into minterms.

Step 6. If there exist any minterms, then go to Step 2.

Step 7. Finally, combine all Boolean functions.

After running Steps 3 and 4 as shown in Fig. 6 and 7, we find the term (0,4,6,8,12,14) has the maximum number of minterms, so the first part of the expression is $F_1 = \overline{x_4}$. At the second cycle, we can get another function $F_2 = \overline{x_2}x_3\overline{x_4}$. The remaining minterm is term (5) after the second cycle, so we can directly obtain the function $F_3 = \overline{x_1}x_2\overline{x_3}x_4$. Finally, we combine all the Boolean functions producing at every cycle and the final circuit is shown in Fig. 8.

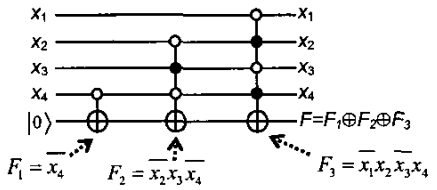

Fig. 8. The final circuit.

V. CONCLUSION

For the quantum Boolean functions, XOR function and AND function are equally important to form the simplified expressions. Thus, our algorithm is developed in accordance with the properties of these two functions. In this paper, we have proposed an algorithm that transforms an original truth table into a quantum Boolean circuit. With our algorithm, we can reduce not only the number of quantum gates but also the basic operations of the circuit. No other auxiliary qubit or intermediate storage is needed. Therefore, it is efficient in terms of both space and time.

REFERENCES

[1] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring" *Proc. of the 35th Annual IEEE Symposium on the Foundations of Computer Science*, pp. 124-134, 1994.

[2] L. Grover, "A fast quantum mechanical algorithm for database search" *Proc. of the 28th Annual ACM symposium on the Theory of Computing*, pp. 212-219, 1996.

[3] I-Ming Tsai and Sy-Yen Kuo, "Quantum Boolean Circuit Construction and Layout under Locality Constraint" *Proc. of the 1st IEEE Conference on Nanotechnology*, pp. 111-116, 2001.

[4] Charles H. Roth, Jr, *Fundamentals of Logic Design* 3rd ed. St. Paul: West Pub. Co. 1985.