Reversible Circuit Synthesis Using a New Modified Tabulation Method

Chin-Yung Lu[1]
Shiou-An Wang[2]
[1]Instructor, Department of Electronic Engineering, Delin Institute of Technology
[2]Assistant Professor, Department of Computer Science and Information Engineering, Delin Institute of Technology

Abstract

An important property of reversible circuits is that they can reduce the energy consumption which is now a big problem for the advanced circuit design. If a circuit is reversible, it can reduce the energy consumption caused by information loss. The tabulation method is more efficient than other simplification methods for combination logic circuits. But the classical tabulation method is not directly applicable to reversible circuits because the basic logic gates, except the NOT gate, are not reversible gates. In this paper, we propose a method to solve the problem so that the tabulation method can be applied to the reversible circuit synthesis. Our algorithm provides a systematic method to simplify the reversible circuit. This can produce the resulting expression in exclusive-sum form and transform it into a final reversible circuit with lower quantum cost. Moreover, we can realize permutations to be reversible circuits with lower quantum cost and without unnecessary garbage bits. We can also convert irreversible circuits by adding qubits to make the circuits reversible. The experimental results show that the average saving in quantum cost is 15.82% compared with previous approaches.

Key word –Reversible circuit, Circuit optimization, Logic synthesis, Quantum computing, Tabulation method

使用新修改的列表方法執行可逆電路合成

盧勤庸[1]

王秀安[2]

[1]德霖技術學院電子工程系專任講師
[2]德霖技術學院資訊工程系助理教授

中文摘要

可逆電路的一個重要特性是它們可以減少能源的消耗，對於先進電路設計而言，能源消耗問題現在是一個大問題，如果電路是可逆的，它可以降低因為訊息損失所造成的能源消耗。在組合邏輯電路中，比較其他的電路化簡方法，列表方法是更有效的方法，但是傳統的列表方法不能直接適用於可逆電路，這是因為除了 NOT 邏輯閘之外，基本的邏輯閘都不是可逆邏輯閘，在這篇論文中，我們提出了一個方法來解決這個問題，可以使得列表方法應用在可逆電路的合成工作上，我們的演算法提供了一種系統性的方式，可以化簡可逆電路，可以產生以互斥和形式表示的結果關係式，並且將其轉換成為具有較低量子成本的可逆電路，此外，我們可以把排列電路轉換成為可逆電路，而且這個電路具有較低的量子成本和沒有不必要的無用量子位元，我們也可以轉換不可逆電路，只需要增加量子位元，就可以使得電路變成可逆。根據實驗結果表明，和以前的方法比較，平均可以節省 15.82%的量子成本。

關鍵字 －可逆電路，電路最佳化，邏輯合成，量子計算，列表方法

I. Introduction

Over the last few years, energy consumption has been the subject of studies for the advanced circuit design. As the CMOS process technology progresses, circuit speed becomes much faster while energy consumption problem is becoming more and more serious. This problem for the portable devices is even more serious than others. Landauer [1] proved that traditional binary irreversible gates result in power dissipation in spite of implementation. Bennett [2] showed that for a binary circuit built from reversible gates no power is consumed due to information loss. It is tempting to consider that reversible circuits may be useful for the power dissipation problem. Moreover, quantum computing [3] is one of the important applications of reversible circuits. Several quantum algorithms which improve some traditional problems have been proposed [4], [5]. Thus, quantum computing becomes one of the most rapidly expanding research fields. In fact, quantum circuits must be reversible, so reversible circuits can be a special case of quantum circuits. A considerable number of studies [6-11, 19, 21] have been made on developing algorithm for synthesis of reversible circuits.

Since the number of inputs and the number of outputs in a reversible circuit must be the same [6], it means there are *n* outputs to synthesize at the same time, where *n* is the number of inputs. Let us consider the synthesis of a reversible circuit from two approaches as shown as follows. First, we need to find as many common gates of most outputs as possible, and then generate individually the result of every output one by one as shown in Fig. 1(a). It is difficult to find the set of common gates by using the tabulation method since the tabulation method is for a single output synthesis. Moreover, if the number of gates to synthesize an output is unlimited, the number of equivalent circuits for this output is also unlimited. Thus, it is very hard to directly find the proper set of common gates which is suitable for most outputs because the possible common gates are too many to find. Additionally, timing is another serious problem.

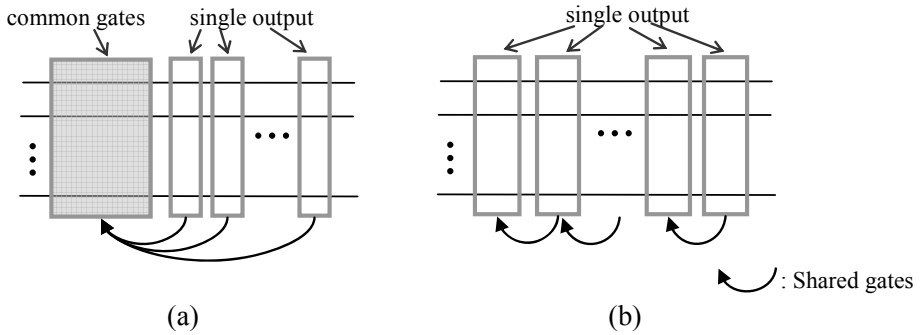

Fig. 1. Different synthesis approaches. (a) common gates shared among most outputs. (b) common gates shared between two outputs.

Second, we can find the more simplified circuit of a single output, and then generate the simplified circuit of the next output according to the previous output functions, until the simplified circuits of all the outputs are generated as shown in Fig. 1(b). In order to obtain a more simplified circuit, the circuit of the output in the rear should reuse the gates in the front. These gates become the common gates between two outputs to reduce the gate count.

In fact, it is very difficult to find the common gates for most outputs, and the first approach is not applicable by using the tabulation method, so we employ the second approach to synthesize. But we do not synthesize each output of the circuit individually and connect them together to obtain the whole circuit. This way leads to too many gates and unnecessary quantum bits (qubits) in the circuit since the front gates of the circuit can not be reused. Our algorithm can reuse the front gates by rebuilding the truth table to reduce the gate count.

II. Notations and Preliminaries

Definition 1. A generalized Toffoli gate with n+1 inputs and n+1 outputs is a reversible gate performing the operation

$$(x_1, x_2, ..., x_n, x_{n+1}) \rightarrow (x_1, x_2, ..., x_n, (x_1 \cdot x_2 \cdot ... \cdot x_n) \oplus x_{n+1}),$$

where x_1, x_2,…, and x_n are controls, x_{n+1} is a target and the notation $\oplus$ denotes the exclusive-OR operation.

According to Definition 1, a NOT gate which performs NOT operation is a generalized Toffoli gate with 1 input and 1 output. Its operation is $(x) \rightarrow (\bar{x})$. Similarly, a CN gate which performs exclusive-OR operation is a generalized Toffoli gate with 2 inputs and 2 outputs. Its operation is $(x_1, x_2) \rightarrow (x_1, x_1 \oplus x_2)$. A Toffoli gate with 2 controls is a generalized Toffoli gate with 3 inputs and 3 outputs. Its operation is $(x_1, x_2, x_3) \rightarrow (x_1, x_2, (x_1 \cdot x_2) \oplus x_3)$. It performs AND operation if input x_3 is equal to 0. These gates are shown in Fig. 2. In Fig. 2(b), a 2-input and 2-output reversible circuit has a CN gate. The vertical line represents a gate, where the notation $\oplus$ denotes a target and ● a control. The horizontal lines represent wires or qubits. Our universal gate set contains three basic reversible gates, NOT, CN, and Toffoli gates. A reversible circuit can be realized by the universal gate set after adding the needed garbage bits. This set is the smallest complete set of gates. These gates were defined by Toffoli [12] and are all reversible gates.

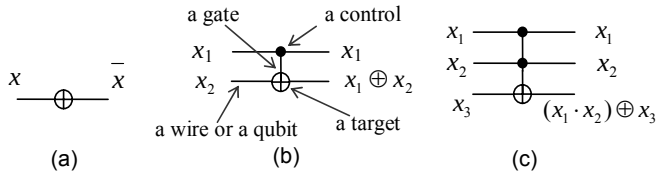

Fig. 2. Basic reversible gates. (a) NOT gate. (b) CN gate. (c) Toffoli gate.

Definition 2. An *exclusive-sum* form of a reversible circuit is the exclusive-OR sum of product terms, in which a product term is composed of literals or subterms, where a subterm can be transformed into CN gates or a Toffoli gate.

For example, $f = e \oplus (a \oplus b)(a \oplus c) \oplus ad$ is the expression in exclusive-sum form. Actually, exclusive-sum form is a variation of exclusive-sum-of-products form. It is a 3-level expression.

A *permutation* which performs permutations of input vectors is a reversible circuit because it has the same number of input and output variables and each mapping to an output pattern from an input pattern is unique.

In order to estimate the performance of algorithms, we apply two main criteria to determine which is a better result. One is the number of garbage bits. The synthesis method in [19] reduces the quantum cost by introducing a few additional garbage bits. The quantum costs of their simplified circuits are lower than others, but increasing the number of garbage bits lead to another problem. Since reversible gates are all quantum gates and the number of qubits is limited and very expensive to implement by today's technology, the number of garbage bits must be as few as possible. Thus it is clear that the number of garbage bits is an important criterion for the synthesis methods. Another criterion is the quantum cost. The quantum cost is based on that proposed in [16-18]. The quantum cost of a reversible gate is the number of one- or two-qubit controlled-V operations which implement the gate.

Moreover, the classical tabulation method called the Quine-McCluskey method [13], [14] is an especially useful method for simplifying switching functions of a large number of variables. Therefore, the tabulation method is an efficient method for minimizing the conventional logic circuit. However, there are some problems to apply the tabulation method for reversible circuits as described as follows.

First, the output is not at the default qubit in the reversible circuit. Because the tabulation method only generates the result of a single output, we can not locate the output at the default qubit for reversible circuits. Assuming that output *f* should be placed at the qubit *b*, the result of synthesis is $f = \bar{a} \oplus c(a \oplus b)$ by using the tabulation method. But output *f* can not be placed at the qubit *b* and its gate count is 3. Of course, we can add a switch gate to change the position of output *f*, but a switch gate is equal to 3 CN gates. So this is not a good approach.

Second, the quantum cost of a NOT gate is equal to a CN gate. The classical tabulation method does not care about the number of NOT gates since the cost of a NOT gate is much lower than those of other gates in the classical gate library. So the synthesis of a classical circuit does not minimize the number of NOT gates. But the cost of a NOT gate is equal to a CN gate in a reversible circuit and therefore, the number of NOT gates should be minimized. Third, every output is generated independently without considering the relationship between the

outputs. In order to find a simpler result, the individual results of all the outputs should share as many common gates as possible. However, the relationship between the outputs is not considered in the process of the synthesis by the classical tabulation method. So it is hard to find a simpler reversible circuit. Finally, a reversible circuit is built by adding unnecessary qubits. Some of the expressions generated by the classical tabulation method can not be transformed into a reversible circuit unless some qubits are added. However, these qubits may be redundant.

III. Modified Tabulation Method

The tabulation method [13] has been used to minimize a classical Boolean circuit. To reduce the number of comparisons between two terms, the minterms are sorted into groups according to the number of 1's in each term. For example, $f(a,b,c,d)=\sum m(0,1,2,3,6,7,12,13)$ is represented by the list of minterms as shown in column 0 of Fig. 3. In the list, the term in group 0 has zero 1's, the terms in group 1 have one 1, those in group 2 have two 1's, and those in group 3 have three 1's. Two terms can be combined if they differ in only one variable. Thus, only terms in adjacent groups must be compared and can be combined by $XY \oplus X\overline{Y} = X$. Comparison of terms in two non-adjacent groups is unnecessary since the two terms will differ in at least two variables. At the same way, comparison of terms within a group is also unnecessary since two terms with the same number of 1's must differ in at least two variables. We compare terms in adjacent groups and form new groups of terms and new columns until no further terms could be combined. To find an essential term can be divided into two steps as described below:

	Column 0
group 0	(1_0) 0000
group 1	(1_1) 0001
	(1_2) 0010
group 2	(1_3) 0011
	(1_6) 0110
	(1_{12}) 1100
group 3	(1_7) 0111
	(1_{13}) 1101

	Column 1	
group 0	(1_0,1_1)	000-
	(1_0,1_2)	00-0
group 1	(1_1,1_3)	00-1
	(1_2,1_3)	001-
group 2	(1_3,1_7)	0-11
	(1_6,1_7)	011-
	(1_{12},1_{13})	110-

	Column 2	
group 0	(1_0,1_1,1_2,1_3)	00--
	~~(1_0,1_2,1_1,1_3)~~	~~00--~~
group 1	(1_2,1_3,1_6,1_7)	0-1-

Fig. 3. The list of minterms from the truth table.

1. Finding a term with 1's and no 0: First we will compare the term in group 0 of Column 0 with all of the terms in group 1 in the same column as shown in Fig. 3. Terms 0000 and 0001 can be combined to eliminate the fourth variable, which generates 000-, where the dash indicates a missing variable. Similarly, 0000 and 0010 are combined to form 00-0 whose expression is $\overline{a}\overline{b}\overline{d}$. The result of other groups compared with their adjacent groups is listed in Column 1 of Fig. 3. Likewise, terms in each group of Column 1 need only be compared with terms in the adjacent groups which have dashes in the same places. The result of comparing terms is listed in Column 2 of Fig. 3. After deleting the

duplicate terms, the process terminates since no further combination is possible.

2. Making a term bigger by adding needed 0's and more 1's: We combine more 1's and needed 0's to become the biggest term which has the maximum number of 1's. This means its corresponding expression is more simplified. We choose the term 00-- to combine. The minterms 1_0, 1_1, 1_2, and 1_3 in the chosen term do not need to be combined since they are in the term already. The minterms which are not in the chosen term are listed as shown in Fig. 4. In the list, the minterms in group 1 have one 1 and the minterms in group 2 have two 1's, excluding the 1's whose corresponding places are dashes in the chosen term. The chosen term 00-- can be combined with group 1 to get a bigger term 0--- since they differ in one variable excluding the third and fourth variables. In order to form the term 0---, we need to add two 0's 0_4 and 0_5. Similarly, another term 0-1- in Column 2 of Fig. 3 also can be combined to get the same term 0---. Thus, we just choose the term with the maximum number of 1's from them since it can be transformed into a more simplified expression.

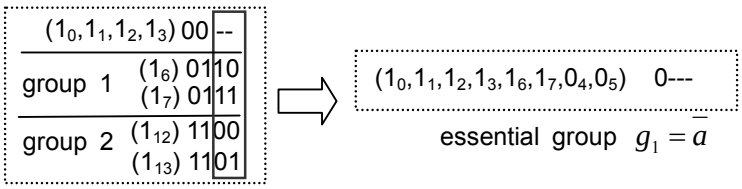

Fig. 4. Finding an essential terms $\bar{a}$.

The term is called an essential term if it has the maximum number of 1's. An essential term can be a variable or a product term. In Fig. 4, the term 0--- can be transformed into the corresponding expression $\bar{a}$ and it is an essential term since it has the maximum number of 1's. In order to find the next essential term, 1's are changed into 0's and 0's into 1's in the term since the OR operation is replaced with the exclusive-OR operation. Then we can find another essential term $b\bar{c}$. 1's are exchanged with 0's in the term. We need to find the remaining essential terms until there is no 1's left. Of course, a single 1 can be an essential term if it can not be combined with any other 1's. Our method can also include don't care minterms like the classical tabulation method.

IV. Realizing Reversible Circuit

In this section, we will realize a reversible circuit according to the expression generated by the modified tabulation method. For a reversible circuit, it must be a multi-output function because the number of outputs is equal to the number of inputs and is always larger than 1. The modified tabulation method can generate the expression for one output at a time, so finding the expression of a multi-output circuit needs to apply the method several times. Realizing a reversible circuit can be divided into four steps as described in the following subsections A-D, respectively.

A. Generating the Expression for a Single Output

The goal of this subsection is to find a set of essential terms for a single output. Each essential term can produce a corresponding product term at every pass until no 1's are left. Then we can obtain a resulting expression by connecting these product terms with exclusive-OR operations. Now the resulting expression is in exclusive-sum-of-products form.

Assuming that the truth table has *n* variables, there are 2^{n-1} 1's in the output *f*, and output *f* should be located at the qubit *x*. Now we will present the procedure to generate the expression of a single output *f* as follows:

1. Exchange 1's with 0's, where *x*=1 or *x*=0, in the output *f* of the truth table to locate output *f* at the qubit *x*.
2. If there are 1's in the truth table, group the list of the minterms for output *f* from the new truth table.
3. Find an essential term.
4. Replace 0's with 1's and 1's with 0's in the term of the truth table.
5. Repeat Steps 2-4 until there are no 1's left in the truth table.
6. Find a minimum set of essential terms. If there is more than one such set, choose a set with the lowest quantum cost.
7. Transform the set of essential terms into a corresponding expression and build a reversible circuit from the expression.

After using the tabulation method, we can get the function $f = x \oplus g_1 \oplus g_2 \oplus ... \oplus g_m$, where g_i is an essential term and *x*, which is the default qubit for the output, is a variable. This function can limit the position of the output at the qubit *x* in the circuit. It is suitable for two circuits. One is a permutation and another is an irreversible circuit which should avoid locating at the synthesized outputs for non-synthesized outputs. Otherwise, the function $f = g_1 \oplus g_2 \oplus ... \oplus g_m$ can not limit the position of the output. This function is suitable for some outputs of an irreversible circuit which needs not limit the output position.

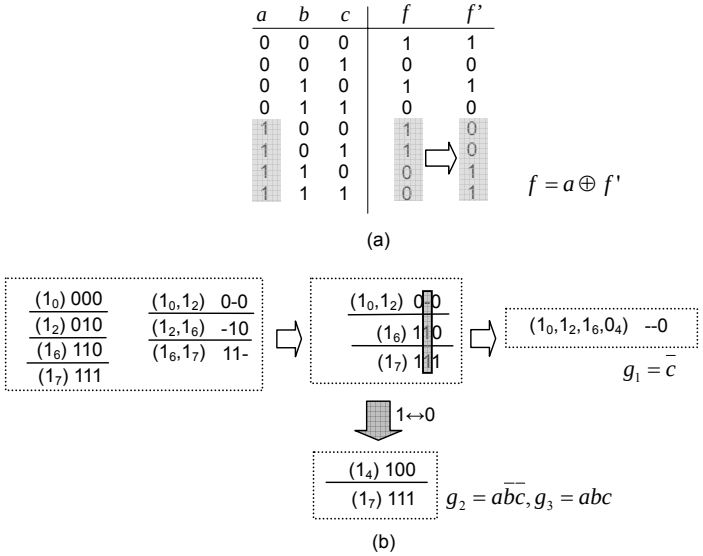

Fig. 5. Generate the expression of a single output. (a) Locate output *f* at the default qubit *a*. (b) Find a set of essential terms.

For example, let $f(a,b,c)=\sum m(0,2,4,5)$. First we exchange 1's with 0's, where a=1, in the output f of the truth table and get the expression $f = a \oplus f'$ since output f should be located at the qubit a as shown in Fig. 5(a). There are still four 1's in the truth table and group the list of the minterms. Combine minterms into an essential term --0 and get a partial expression $\bar{c}$ as shown in Fig. 5(b). Then replace 0's with 1's and 1's with 0's in the term of the truth table. Now, there are only two 1's in the truth table. We can directly get two essential terms $a\bar{b}\bar{c}$ and abc since these two 1's differ in two variables. The resulting expression is $f = a \oplus f' = a \oplus g_1 \oplus g_2 \oplus g_3 = a \oplus \bar{c} \oplus a\bar{b}\bar{c} \oplus abc$. $a\bar{b}\bar{c}$ and abc can be further transformed into $a(b \oplus \bar{c})$ and $\overline{X \oplus Y} = X \oplus \bar{Y}$. This transformation is described in subsection B. For completing the circuit, add the proper parentheses into the expression $f = (a \oplus (\bar{c} \oplus a(b \oplus \bar{c})))$ as described in subsection C. Finally, we can build the reversible circuit from the expression as shown in Fig. 6.

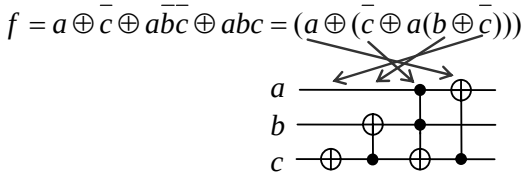

Fig. 6. The circuit generated from the resulting expression.

B. Expression Transformation

The expression as described in subsection A is still too complicated and can be further simplified. Actually, the expression generated by the modified tabulation method is in exclusive-sum-of-products form. After performing the process of expression transformation, the resulting expression really becomes an exclusive-sum form. There are two stages to do the process of expression transformation as shown below:

1. Combine two product terms of the expression to generate a more simplified term. We can find that the quantum cost of the simplified expression becomes lower.
2. Reduce the number of NOT gates since the cost of a NOT gate is equal to a CN gate. Thus, in order to reduce the gate count and quantum cost, the number of NOT gates should be further reduced by the following two equalities

$$\begin{cases} (\overline{X \oplus Y}) = \bar{X} \oplus Y = X \oplus \bar{Y} \\ \bar{X}\bar{Y} = \bar{X} \oplus \bar{X}Y = \bar{Y} \oplus X\bar{Y} \end{cases},$$

where X and Y are variables or terms, and a term here is the exclusive-OR sum of variables. This reveals that the NOT gates can be shared or eliminated to reduce the number of NOT gates. For example, the function $f = a \oplus \bar{c} \oplus a(\overline{b \oplus c})$ can move the position of a NOT operation by $\overline{X \oplus Y} = X \oplus \bar{Y}$. The new function $a \oplus \bar{c} \oplus a(b \oplus \bar{c})$ has lower quantum cost since the NOT operation $\bar{c}$ is shared.

C. Build a Reversible Circuit from the Resulting Expression

Although an expression can now be transformed into a more simplified one by the expression transformation, the synthesis work for a single output is not complete yet. We still need to build a reversible circuit from the expression. The algorithm to build a reversible circuit from the expression is as shown below:

1. Add parentheses in the expression: According to the function of a gate, add appropriate parentheses into the expression. Each set of parentheses represents a CN gate or a Toffoli gate excluding a NOT gate. The direction of adding parentheses can be classified into three modes. If the resulting expression is $f = x_1 \oplus x_2 \oplus ... \oplus x_n$, where x_i is a variable or a term. The modes are described as follows:
   a. From left to right: $f = (((x_1 \oplus x_2) \oplus x_3)... \oplus x_n)$.
   b. From right to left: $f = (x_1 \oplus (x_2 \oplus ...(x_{n-1} \oplus x_n)))$.
   c. From two sides to the center: $f = (((x_1 \oplus x_2) \oplus x_3) \oplus ... \oplus (x_{n-2} \oplus (x_{n-1} \oplus x_n)))$.

If parentheses can not be added to the expression, the sequence of the terms in the expression should be changed excluding the first term which can locate the output at the default qubit. For example, $f = (((c \oplus a) \oplus b) \oplus ab)$. There are 3 set of parentheses in the expression *f*. This means the reversible circuit has 3 gates which are 2 CN gates and 1 Toffoli gate.

Transform each set of parentheses into a corresponding gate one by one. If parentheses are contained within other parentheses, start with the innermost group and work outward to transform into corresponding gates. If parentheses are at the same level, the sequence of the transformation may need to be adjusted to build the circuit with a lower gate count. For example, $f = ((c \oplus ab) \oplus (b \oplus a))$. Since $(c \oplus ab)$ and $(b \oplus a)$ are at the same level, $(c \oplus ab)$ needs to be transformed first, then transform $(b \oplus a)$. This way we can generate a simplified circuit with only 3 gates. On the contrary, the circuit can not be generated without adding a qubit if $(b \oplus a)$ is transformed first.

If the circuit can not be realized, it is necessary to add qubits into the circuit. The right time to add a qubit is described as follows:

1. If there is only a product term $(y_1 y_2 ... y_m)$ in a set of parentheses, where y_i is a variable or a term, add an extra qubit whose input is 0. This means that the product term is transformed into an AND gate.
2. If a term in a set of parentheses is $(xz \oplus xy_1 y_2 ... y_m)$, where *x* is a variable or a term, and *z* is a 1 or a variable, then add an extra qubit whose input is 0. This means that we can copy the needed function to implement the circuit.
3. A circuit must be added an extra qubit, or it can not be realized.

D. Generating the Complete Expression of a Reversible Circuit

Because there are multiple outputs in a reversible circuit and the tabulation method can only simplify a single output at a time, we should choose a proper output first to realize the

circuit. Its outputs become the intermediate inputs of other outputs. And then rebuild the truth table to find the circuit of other outputs. This way can lead to lower quantum cost and does not add unnecessary qubits because of reuse by rebuilding the truth table. Note that the expressions of the outputs which have been synthesized are not changed. There are two rules of choosing a proper output to simplify as follows:

a) First choose the output whose expression is simpler than others because it is easier to realize.
b) If the complexities of the expressions are similar, choose the output whose expression has fewer NOT operations.

For finding the possible better result, we can perform the tabulation method for all the possible combination of outputs. Besides, there are two special gates, Peres gate and inverse Peres gate which are composed of a Toffoli gate and a CN gate, each with a quantum cost 4. It is possible to make the quantum cost of a reversible circuit even lower by using Peres gates or inverse Peres gates. The transformation of a Peres gate is to add a CN gate after a 2-control Toffoli gate which is at the last place of the circuit, where the target of the CN gate should be at a garbage bit. Now, we will present the procedure to generate the complete expression of a reversible circuit as follows:

1. Estimate the approximate gate count of each output which has not been chosen.
2. Choose an output which has not been chosen and has the lower gate count than others.
3. Group the list of the minterms for this output from the truth table and generate the expression as described in subsection A.
4. The expression is transformed into a more simplified one as described in subsection B.
5. Realize the circuit of this output from the expression as described in subsection C.
6. Treat the intermediate outputs as the new inputs and then rebuild the truth table.
7. If there is any output which has not been chosen, then go to Step 1.
8. Transform 2-control Toffoli gates into Peres gates or inverse Peres gates.
9. In order to find a much simpler result, add proper gates at the front of the circuit, rebuild the truth table, and then repeat Steps 1-8.

TABLE I. TRUTH TABLE FOR REALIZING A REVERSIBLE CIRCUIT.

Input (output *e*)			Input (output *d*)			Input (output *f*)			Output		
a	*b*	*c*	*a*	*e*	*z*	*d*	*e*	*z*	*d*	*e*	*f*
0	0	0	0	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	1	0	0	1
0	1	1	0	1	0	0	1	0	0	1	1
1	0	0	1	0	0	1	0	0	1	0	0
1	0	1	1	1	1	0	1	1	0	1	0
1	1	0	1	1	0	1	1	0	1	1	0
1	1	1	1	0	1	1	0	1	1	0	1

We will give an example to explain our algorithm in detail. From the Output column of the truth table in Table I, we find the expression of output *e* is simpler than others. Then generate the expression $e=(b\oplus(a\oplus\bar{c}))$ by using our algorithm. Output *e* which is at the correct wire is at the qubit *b*, so realize the circuit of output *e* in part (i) of Fig. 7. The intermediate outputs *a*, *e* and *z* become the new inputs of the truth table, where $z=a\oplus\bar{c}$, and then rebuild the truth table.

There are still two outputs *d* and *f* which have not been chosen yet. Choose output *d* to simplify and obtain $d=(a\oplus ez)$. Realize the circuit of output *d* (part (ii) of Fig. 7). Finally, choose output *f* to simplify and obtain $f=((z\oplus e)\oplus de)$. Then realize the circuit of output *f* as shown in part (iii) of Fig. 7. We find the number of garbage bits is 0, the gate count is 6, and the quantum cost is 12, where the center two gates become a Peres gate.

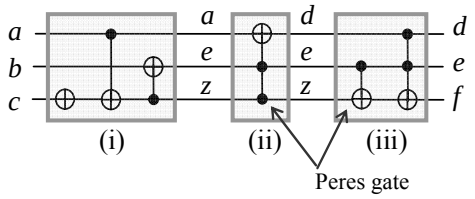

Fig. 7. Final circuit of the example in Table I.

V. Experimental Results

Table II presents the results of all the 40,320 reversible functions of 3 variables by using various algorithms [20], [22] to compare with ours. Synthesis takes one hour 12 minutes on a 1.73 GHz Intel Pentium M processor with 256MB RAM for all the functions using our algorithm. Table II shows our results are lower than Agrawal [22] and almost equal to the optimal results in the average gate count. The results of Maslov [20] are equal to ours in the average gate count, but they take much more time consumption than ours.

To compare our results with other synthesis methods, we performed a set of simulations to evaluate the performance of our algorithm. The benchmarks of our simulations are available at [17], [22]. Table III shows the specifications of benchmarks and detail simulation results by using our algorithm. The results of other synthesis methods are based on [17], [22]. The Name and Size columns in Table III contain the name and the number of qubits of benchmarks, respectively. The Others column in the Garbage columns of the table shows the minimum number of garbage bits generated by other methods described at [17], and the Ours column in the Garbage columns contains the minimum number of garbage bits by using our algorithm. The Others and Ours columns in the Quantum Cost column are the minimum quantum cost by using other methods and our algorithm, respectively. Then we list the percentage saving in quantum cost to compare our results with other methods.

We find that our results in the number of garbage bits are equal to or lower than those for all other synthesis methods. This means that the number of garbage bits is not increased by our algorithm. We also find that our results in quantum cost are equal to or much lower than most other synthesis methods except three cases. We find that one of these three cases make quantum cost lower by increasing the number of garbage bits. As discussed Section II, the number of garbage bits should be kept minimum because they are very expensive. The average quantum cost saving in the table is 15.82%. In Table III, the circuits in the last eight rows are permutations and our results are better than or equal to those using other methods in the number of garbage bits and quantum cost.

TABLE II. EXPERIMENTAL RESULTS FOR ALL REVERSIBLE FUNCTION OF THREE VARIABLES.

#gates	Agrawal [22]	Maslov [20]	Ours	Optimal
10	0	0	0	0
9	30	2	0	0
8	3297	659	686	577
7	12488	10367	10391	10253
6	13620	16953	16846	17049
5	7503	8819	8885	8921
4	2642	2780	2772	2780
3	625	625	625	625
2	102	102	102	102
1	12	12	12	12
0	1	1	1	1
Average gate count	6.10	5.88	5.88	5.87
Time	few min.	96 hours	1 hour 12 min.	N/A

TABLE III. EXPERIMENTAL RESULTS FOR REALIZING REVERSIBLE CIRCUITS.

Benchmark		Garbage		Quantum Cost		Cost
Name	Size	Others[17]	Ours	Others[17]	Ours	Saving
mod5	5	4	4	13	8	38.46%
2of5(#1)	6	5	5	107	38	64.49%
2of5(#2)	6	6	5	32	38	-18.75%
rd32	4	2	2	8	8	0%
5mod5(#3)	6	5	5	84	83	1.19%
5mod5(#4)	6	5	5	76	83	-9.21%
xor5	5	4	4	4	4	0%
rd53(#5)	7	4	4	75	73	2.67%
rd53(#6)	7	4	4	65	73	-12.31%
ham3(#1)	3	0	0	7	7	0%
ham3(#2)	3	0	0	10	7	30%
3_17	3	0	0	12	12	0%
example1[22]	3	0	0	16	7	56.25%
example2[22]	3	0	0	5	5	0%
example3[22]	3	0	0	15	7	53.33%
example4[22]	4	0	0	51	19	62.75%
example5[22]	4	0	0	20	20	0%

VI. Conclusions

The tabulation method is efficient to minimize a Boolean function. It is an important tool which is often used in the conventional logic design. But it is not directly applicable to reversible circuit design because the basic logic gates except the NOT gate are not reversible gates. In this paper, we propose a new minimization algorithm based on the classical tabulation method for minimizing a reversible circuit to generate the expression in exclusive-sum form so that its quantum cost is lower than or equal to that in exclusive-sum-of-products form. We can realize permutations to be reversible circuits and also convert irreversible circuits into reversible circuits after adding some qubits. With our algorithm, we can reduce not only the quantum cost but also the number of garbage bits. The average quantum cost saving is 15.82%. According to the simulation results, our algorithm is very efficient and useful for reversible circuits just like the classical tabulation method for the conventional Boolean circuits.

References

[1] R. Landauer, “Irreversibility and heat generation in the computing process, “IBM J. Research and Development, pp. 5:183-191, 1961.

[2] C. H. Bennett, “Logical reversibility of computation, “IBM J. Research and Development, pp. 17:525-532, November, 1973.

[3] M. Nielsen and I. Chuang, “Quantum Computation and Quantum Information,” Cambridge, U.K.: Cambridge Univ. Press, 2000.

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

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

[6] A. Mishchenko and M. Perkowski, “Logic synthesis of reversible wave cascades, “ in International Workshop on Logic and Synthesis, pp. 197-202, June, 2002.

[7] D. Voudouris, S. Stergiou, and G. Papakonstantinou, “Minimization of Reversible Wave Cascades, “ IEICE Trans. Fundamentals, Vol. E88-A, No. 4, pp. 1015-1023, April, 2005.

[8] V. V. Shende, A. K. Prasad, K. N. Patel, I. L. Maslov, and J. P. Hayes, “Scalable simplification of reversible logic circuits, “ in IWLS, May, 2003.

[9] D. M. Miller and G. W. Dueck, “Spectral techniques for reversible logic synthesis, “ in 6th International Symposium on Representations and Methodology of Future Computing Technologies, pp. 56-62, March, 2003.

[10] P. Kerntopf, “A new heuristic algorithm for reversible logic synthesis, “ in Design Automation Conference, pp. 834-837, June, 2004.

[11] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, “Synthesis of reversible logic circuits,” IEEE Trans. On CAD of Integrated Circuits and Systems, Vol. 22, No. 6, pp. 710-722, June, 2003.

[12] T. Toffoli, “Reversible computing, “ Tech. memo MIT/LCS/TM-151, MIT Lab for Comp. Sci., 1980.

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

[14] Richard S. Sandige, “Modern Digital Design,” New York: McGraw-Hill, 1990.

[15] D. Maslov and G. W. Dueck, “Reversible cascades with minimal garbage, “ IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 23, No. 11, pp. 1497-1509, November, 2004.

[16] D. Maslov and G. W. Dueck, “Improved quantum cost for n-bit Toffoli gates, “ Electronics Letters, Vol. 39, No. 25, pp. 1790-1791, December, 2003.

[17] D. Maslov, G. W. Dueck, and N. Scott, “Reversible logic synthesis benchmarks page, “ [Online] Available: www.cs.uvic.ca/~dmaslov/, February, 2005.

[18] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, “Elementary gates for quantum computation,” Physical Review A, 50: pp. 3457-3467, Nov. 1995.

[19] D. Maslov, “Dynamic Programming algorithms as reversible circuits: symmetric function realization,” TR03-161, UNB, August, 2003.

[20] D. Maslov, D. M. Miller, and G. W. Dueck, “Techniques for the synthesis of reversible Toffoli networks, “ in quant-ph/0607166, July, 2006.

[21] D. M. Miller, D. Maslov, and G. W. Dueck, “A transformation based algorithm for reversible logic synthesis, “ in Design Automation Conference, pp. 318-323, June, 2003.

[22] A. Agrawal and N. K. Jha, “Synthesis of reversible logic, “ in Design, Automation and Test in Europe, pp.1384-1385, February, 2004.