PAPER

Reduction of Quantum Cost by Making Temporary Changes to the Function

Nurul AIN BINTI ADNAN[†a)], ***Nonmember***, **Shigeru YAMASHITA**[†b)], ***Senior Member***, ***and*** **Alan MISHCHENKO**[††c)], ***Nonmember***

SUMMARY This paper presents a technique to reduce the *quantum cost* by making temporary changes to the functionality of a given Boolean function. This technique is one of the very few known methods based on manipulating *Exclusive-or Sum-Of-Products (ESOP)* expressions to reduce the quantum cost of the corresponding circuit. The idea involves adding *Mixed Polarity Multiple-Control Toffoli (MPMCT)* gates to temporarily change the functionality of the given function, so that the modified function has a smaller quantum cost. To compensate for the temporary change, additional gates are inserted into the circuit. The proposed method finds a small ESOP expression for the given function, and then finds a good pair of product terms in the ESOP expression so that the quantum cost can be reduced by applying the transformation. The proposed approach is likely to produce a better quantum cost reduction than the existing methods, and indeed experimental results confirm this expectation.

key words: *quantum cost, changing functionality, Mixed Polarity Multiple Control Toffoli gates, ESOP*

1. Introduction

Quantum computers are expected to outperform conventional computers due to the use of some known quantum algorithms. These quantum algorithms are designed to solve important problems such as database search and factorization. In order to demonstrate the ability of quantum computing in the near future, quantum algorithms should be efficiently implemented. In general, many of the established quantum algorithms include a part to calculate Boolean functions corresponding to a problem instance; quantum algorithms usually consist of *common parts* and *unique parts* [1]. *Common parts* do not differ for each problem instance. On the other hand unique parts differ for each problem instance. For example, one of the famous quantum algorithms, Grover search algorithm [2], consists of the so-called *oracle* part and the other part. The oracle part is used to calculate a Boolean function depending on a problem.

Therefore, an efficient design technique for realization of a Boolean function is crucial even for quantum circuits, as pointed out in the literature (e.g., [1])

Manuscript received September 16, 2016.
Manuscript revised February 2, 2017.
Manuscript publicized March 23, 2017.

[†]The authors are with the Graduate School of Information Science and Engineering, Ritsumeikan University, Kusatsu-shi, 525–8577 Japan.

[†]The author is with the Electrical and Computer Sciences University of California, Berkeley, USA.

a) E-mail: nu_ain@ngc.is.ritsumei.ac.jp
b) E-mail: ger@cs.ritsumei.ac.jp
c) E-mail: alanmi@berkeley.edu

DOI: 10.1587/transinf.2016EDP7397

To design a reversible circuit, we need to use primitive reversible gates. Among existing reversible gates, Mixed Polarity Multiple-Control Toffoli (MPMCT) gates are often used to implement a reversible circuit for Boolean functions. This is because the functionality of MPMCT gates naturally corresponds to AND-EXOR expressions as mentioned later. Indeed most existing researches generate an initial circuit consisting of MPMCT gates [3], [4] based on a small Exclusive-or Sum-Of-Product (ESOP) expression [5]. After that we decompose a large gate (i.e., with the large number of inputs) into *elementary (i.e., physically realizable) gates*. Once an initial circuit is obtained, further post-optimization techniques such as library-based, transformation-based and template-based optimization can be applied [7].

In this paper, we propose a totally new technique for generating initial quantum circuits for Boolean functions with reduced quantum cost. Intuitively, our technique works as follows. We attempt to make temporary changes to the original function by adding some MPMCT gates, so that the changed function leads to a smaller quantum circuit. We evaluate "good" temporary changes and pick the one that reduces the total quantum cost, including the additional MPMCT gates, as much as possible.

Further, based on the idea, we propose an iterative heuristic method to reduce the quantum cost for practical (i.e., large) circuits. Our technique is of particular interest since it can be considered as a good *pre-optimization method* to make temporary changes to the given function, unlike the existing methods that keep the function unchanged. Over the years, many ESOP minimization-based quantum circuit design methods have been proposed, but to our knowledge, ours is the only work (i.e., [6]) that formally discusses how ESOP expressions can be manipulated to reduce the quantum cost of the corresponding circuit *by making temporary changes to the original function*.

This paper is organized as follows. The next section explains the basic concepts used in this work. Section 3 proposes our idea to reduce the quantum cost by manipulating the ESOP expressions. Section 4 explains our proposed heuristic approach based on the idea with motivational examples. Section 5 shows our experimental results, and Sect. 6 concludes the paper and discusses future work.

Fig. 1 A NOT gate

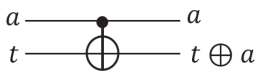

Fig. 2 A CNOT gate

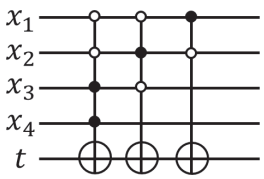

Fig. 3 MPMCT gates

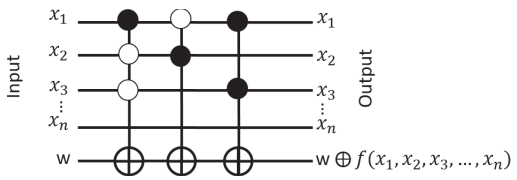

Fig. 4 Reversible circuit model

2. Preliminaries

2.1 Qubit

In classical computers, a bit is used to store the information; a bit can be in one of the two states, 0 or 1. In contrast, in quantum computers, quantum bits (usually called qubits) are used. A qubit can be found in superposition states, that is, in multiple states (i.e., in both states, 0 or 1) at the same time. The implementation of superposition states in qubits is very important as it allows for representing multiple outcomes of a particular computation. This advantage explains the power of quantum computers.

2.2 Quantum Gates

A quantum computer can be constructed from a universal set of logic gates. These logic gates are inherently reversible, that is, they map each possible input pattern into a unique output pattern. Here we introduce some reversible gates used in this paper.

(a) NOT Gate

NOT gate is the simplest example of a one-qubit reversible gate. A reversible NOT gate has one input and one output, as illustrated in Fig. 1. It has no control and inverts any value of bit (either 0 or 1) that passes through the gate.

(b) CNOT Gate

A controlled-NOT (CNOT) gate has a single control bit, and a single target bit, as illustrated in Fig. 2. A CNOT gate inverts the value of the target bit if the control bit is set to 1. All values passing through the control bit remain unchanged.

(c) Mixed Polarity Multiple Control Toffoli (MPMCT) Gate

A reversible gate with all positive controls is called a Multiple-Control Toffoli (MCT) gate, while a reversible gate having both positive and negative controls is called a Mixed Polarity Multiple Control Toffoli gate (MPMCT). The size of an MPMCT gate is the number of controls plus one.

An MPMCT gate with target bit x_t, positive controls $\{x_{i_1}, x_{i_2}, \cdots, x_{i_k}\}$ and negative controls $\{x_{i_{k+1}}, \cdots, x_{i_m}\}$ maps x_t to $(x_{i_1} x_{i_2} \cdots x_{i_k}\ \overline{x_{i_{k+1}}} \cdots \overline{x_{i_m}}) \oplus x_t$. The exclusive OR operation is denoted by $\oplus$. The value of the target bit of an MPMCT gate is inverted if all the positive and negative controls are set to 1 and 0, respectively. On the other hand, the values of both control bits and unconnected bits pass through the gate unchanged.

Figure 3 contains an example of 3 MPMCT gates. The target bit is indicated by $\oplus$ symbol, while a black dot symbol is used to denote the positive control and a white dot is used to denote the negative control. The function for gates in Fig. 3 can be calculated as f(x) = $\overline{x_1} \cdot \overline{x_2} \cdot x_3 \cdot x_4 \oplus \overline{x_1} \cdot x_2 \cdot \overline{x_3} \oplus x_1 \cdot \overline{x_2}$. A negative control is identified by an overline in the logic expression.

2.3 Quantum Cost

In the quantum computation research community, it is usually assumed that physically realizable elementary gates in the quantum computation cannot have more than two qubit interactions. Therefore, in most of the research publications on quantum circuit design, the cost of a quantum gate with many inputs is defined as the number of basic (i.e., less than three inputs) gates to realize the gate. This cost is often called a *quantum cost*.

In this paper, we use the quantum cost defined in the previous work [8], as shown in Table 1. The table shows the cost of an MPMCT gate with m control bits when $m - 3$ auxiliary bits are available. Even if we have more efficient ways to realize an MPMCT gate in the future, our framework does not need to change; we simply change the cost values accordingly.

2.4 Realizing Boolean Function with MPMCT Gates

Like most of the existing design methods, we consider the model of reversible circuits for calculating Boolean functions as shown in Fig. 4. More formally, our circuit model is as follows:

- We have n input variables, $x_1, \cdots, x_n$, whose states should not be changed.
- We have one target bit, w, used to calculate a target Boolean function $f(x_1, \cdots, x_n)$ as $w \oplus f(x_1, \cdots, x_n)$.

We impose the above restrictions because they are necessary for a reversible circuit to be used in existing quantum algorithms which utilize the so-called *quantum interference*.

A **minterm** of a Boolean function is the combination of

Table 1 Quantum cost of MPMCT gates with m control bits ($0 \le m \le 15$)

Number of Control Bits m	Number of Negative Controls															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1															
1	1	2														
2	5	5	6													
3	14	14	16	18												
4	20	20	20	22	24											
5	32	32	32	34	36	38										
6	44	44	44	44	46	48	50									
7	56	56	56	56	58	60	62	64								
8	68	68	68	68	68	70	72	74	76							
9	80	80	80	80	80	82	84	86	88	90						
10	92	92	92	92	92	92	94	96	98	100	102					
11	104	104	104	104	104	104	106	108	110	112	114	116				
12	116	116	116	116	116	116	116	118	120	122	124	126	128			
13	128	128	128	128	128	128	128	130	132	134	136	138	140	142		
14	140	140	140	140	140	140	140	140	142	144	146	148	150	152	154	
15	152	152	152	152	152	152	152	152	154	156	158	160	162	164	166	168

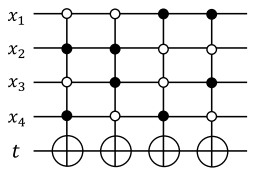

Fig. 5 The quantum circuit for Table 2

values (negative or positive) of all of the input variables such that the Boolean function evaluates to 1 on this combination. In the following, an $MPMCT_n$ gate means an MPMCT gate that has n control bits.

To realize an n-input Boolean function with k minterms by a reversible circuit, it is possible to use k $MPMCT_n$ gates such that;

(1) Each $MPMCT_n$ gate corresponds to each minterm of the function.
(2) The polarity of each control bit for an MPMCT gate corresponds to each variable's polarity in the corresponding minterm.

In other words, if x_i or $\overline{x_i}$ appears in a minterm, the corresponding control bit is positive or negative, respectively. In this construction, the target bit of all the $MPMCT_n$ gates is the same as the qubit, for which we want to realize the function.

For instance, Table 2 shows a 4-input Boolean function with 4 minterms, and the circuit in Fig. 5 realizes the function: $x_2 \cdot x_4 \cdot \overline{x_1} \cdot \overline{x_3} \oplus x_2 \cdot x_3 \cdot \overline{x_1} \cdot \overline{x_4} \oplus x_1 \cdot x_4 \cdot \overline{x_2} \cdot \overline{x_3} \oplus x_1 \cdot x_3 \cdot \overline{x_2} \cdot \overline{x_4}$. For example, the leftmost gate in Fig. 5 corresponds to $x_2 \cdot x_4 \cdot \overline{x_1} \cdot \overline{x_3}$; the control bits for x_2 and x_4 are in the positive polarities denoted by black circles, and x_1 and x_3 are in the negative polarities denoted by white circles.

2.5 Initial Circuit Generation by Minimizing ESOP Expression

To understand the main contribution of this paper, it is beneficial to understand how we can design a reversible circuit by minimizing ESOP expressions. For that purpose, in the following, we review a design method proposed by Arabzadeh *et. al.* [3]. Their method can find a circuit with a low quantum cost by minimizing ESOP expressions using the rules applied to a Karnaugh map (a Kmap, hereafter).

A Kmap for a Boolean function is a two-dimensional arrangement of cells where each cell has the corresponding output value ("1" or "0") of the Boolean function. In a Kmap, we often omit "0" for simplicity. For example, the leftmost diagram in Fig. 6 shows a Kmap for the function: $\overline{x_1}x_2\overline{x_3}x_4 \oplus \overline{x_1}x_2x_3x_4 \oplus x_1x_2\overline{x_3}x_4 \oplus x_1x_2x_3x_4$. The cell at the second row from the top and the second col-

Table 2 A truth table for a 4-input Boolean function with 4 minterms

x_1	x_2	x_3	x_4	$f(x)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

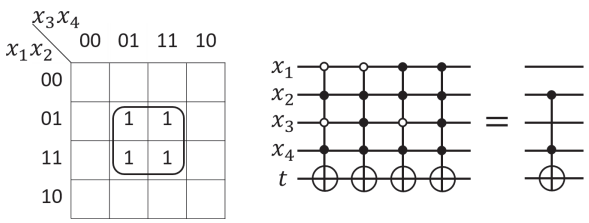

Fig. 6 Optimization for a group with 2^2 cells

umn from the left corresponds to the function value when $(x_1, x_2, x_3, x_4) = (0, 1, 0, 1)$. Each cell having 1 corresponds to one minterm in the function. A Kmap is very useful for simplifying a Boolean expression because we can determine which minterms should be combined from the positional relationship of corresponding cells in the Kmap. For example, the rectangular shape of adjacent four cells in the leftmost diagram in Fig. 6 means that we can get the simple expression: x_2x_4 for the function.

In the following, we refer to a blank cell or a cell having the 0 value as **0-value cell** in a Kmap. Also, a cell having the 1 value is called **1-value cell**. For a minterm with n input variables, it can be expressed as one $MPMCT_n$ gate. Also one 1-value cell corresponds to one minterm in a Boolean function. Therefore, in a Kmap for an n-input function, one 1-value cell corresponds to one $MPMCT_n$ gate.

In the classical logic design, we can optimize a two-level AND/OR/NOT expression of a function by manipulating the Kmap for the function with some rules. Arabzadeh *et. al.* proposed a similar optimization method for quantum circuits based on Kmaps [3]; we refer to the method as the Arabzadeh method in this paper. In the Arabzadeh method, they apply the so-called CTRs (Common-Target Rules) to quantum circuits; the rules are stated as follows.

(a) All 1-value cells should at least belong to one *group* of cells.
(b) Each group has a size of 2^p $(p \geq 0)$.
(c) Each group should have a maximum size.
(d) Each 1-value cell should belong to an odd number of groups.
(e) Each 0-value cell should belong to an even number of groups.
(f) The number of groups should be minimum (as much as possible).

A *group* is a collection of cells that are next to each other, and denoted by a rectangle on a Kmap. The above rules indicate that we can also cover 0-value cells when we consider the optimization of a quantum circuit; this is in contrast to the case of a two-level AND/OR/NOT expression where we focus only on 1-value cells.

A group in a Kmap is used to optimize a part of an expression of a certain Boolean function. For instance, 2^p $(p \geq 0)$ cells correspond to 2^p $MPMCT_n$; if they can be in one group of 2^p $(p \geq 0)$ cells, the 2^p $MPMCT_n$ can be optimized to become a single $MPMCT_{n-p}$ gate. Thus, we can reduce the quantum cost drastically.

Figure 6 shows how this rule is applied to optimize a circuit. The four leftmost cells in a Kmap in Fig. 6 belong to a group that can be described as $\overline{x_1}x_2\overline{x_3}x_4 \oplus \overline{x_1}x_2x_3x_4 \oplus x_1x_2\overline{x_3}x_4 \oplus x_1x_2x_3x_4$. This corresponds to the circuit that has four $MPMCT_4$ gates in the middle of the same figure. After the optimization by using the above rules, we can optimize the function to be x_2x_4, and thus we can get the optimized circuit on the right of the same figure. By looking up the quantum cost listed in the literature [8], the quantum cost is reduced from 80 to only 5 in this example.

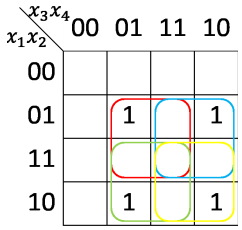

Fig. 7 Group patterns for Kmap for Table 2

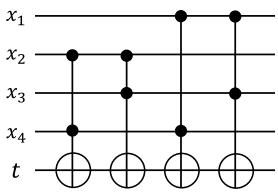

Fig. 8 The quantum circuit for Fig. 7

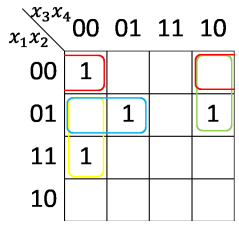

Fig. 9 A bad case for Arabzadeh method

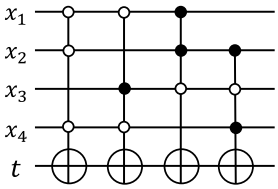

Fig. 10 The quantum circuit for Fig. 9

In short, the quantum cost can be reduced by eliminating more control bits of MPMCT gates in a circuit. When the size of a group (of adjacent cells) is 2^p, we can see that the ratio of the number of gates after optimization to the number of gates before optimization becomes $\frac{1}{2^p}$. Also the ratio of the number of control bits after optimization to the number of control bits before optimization becomes $\frac{n-p}{n}$. Thus when the size of a group (of adjacent cells) is large, we can reduce the quantum cost by using the above optimization strategy.

For the Boolean function shown in Table 2, if we make groups using a Kmap as shown in Fig. 7, we are able to realize the resulting quantum circuit as shown in Fig. 8. In the above example, the quantum cost is reduced from 80 to 20. Thus, in most cases, we can reduce the quantum cost by using the Arabzadeh method. In the next section, we propose another way to reduce the quantum cost; our method can reduce the cost of the same example to only 9.

3. A Pre-Optimization Technique To Reduce Quantum Costs

3.1 General Idea and a Motivational Example

When we utilize the Arabzadeh method, the location of a 1-value cell in the Kmap always stays at the initial point as a given Boolean function. It seems to be natural to do so, but the initial locations of 1-value cells may cause bad results in some cases.

For example, for a Kmap shown in Fig. 9, the method can only generate the circuit as shown in Fig. 10; we can find a much better circuit by using our pre-optimization technique as follows.

Our proposal in this paper involves moving 1-value cells so that the resulting Kmap can be implemented by a

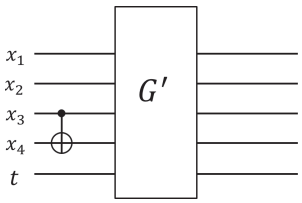

Fig. 11 Insertion of a CNOT gate before G'

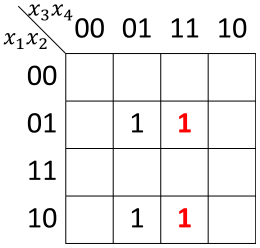

Fig. 12 The Kmap after the insertion of the first CNOT gate

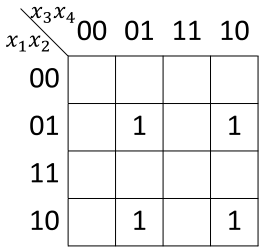

Fig. 13 The Kmap before the insertion of the first gate

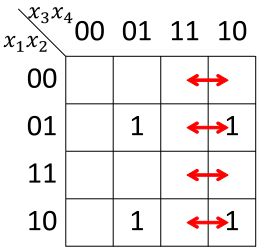

Fig. 14 The movement of cells by the insertion of the first CNOT gate

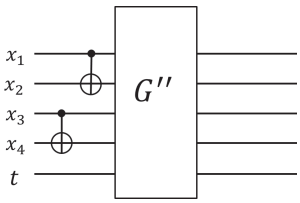

Fig. 15 The insertion of the second CNOT gate

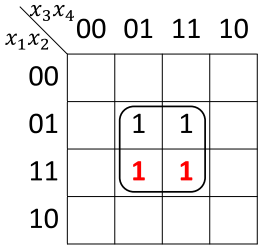

Fig. 16 The Kmap after the insertion of the two CNOT gates

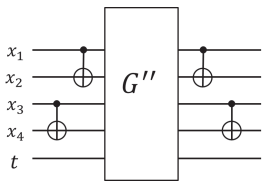

Fig. 17 The insertion of two CNOT gates to restore the original function

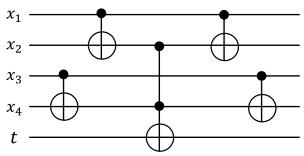

Fig. 18 The final quantum circuit by our method: Cost 9

reversible circuit with a much lower quantum cost. In order to do so, we add MPMCT gates to the initial quantum circuit. Note that inserting MPMCT gates changes the functionality of the quantum circuit. Therefore we also add the corresponding MPMCT gates at the end in order to cancel the effect so that we can have the desired functionality, as shown below.

Let us explain what we mean by "inserting an MPMCT gate" using the following example. Let G be a quantum circuit realizing the Boolean function shown in Table 2. In our proposed method, we insert an MPMCT gate whose positive control bit is x_3 and the target bit is x_4 (i.e., CNOT gate) before G' as shown in Fig. 11. Our idea is as follows: if we implement a circuit G' whose Kmap is shown in Fig. 12, the entire circuit (i.e., G' with the inserted CNOT) realizes the original function of (G) whose Kmap is shown in Fig. 13.

The reason is as follows: the inserted CNOT (the control bit is x_3 and the target bit is x_4) inverts the value of x_4 when $x_3 = 1$. This means that the gate changes the input state $(x_1, x_2, x_3, x_4) = (0110)$ to (0111), for example. Also, the gate moves (1010) to (1011). More precisely, the gate swaps four pairs of cells in a Kmap, as shown in Fig. 14. Therefore, we can conclude that by inserting the gate, it is enough for G' to realize the function as shown in Fig. 12. In this case, the desired function will be realized as shown in Fig. 13.

Similarly, if we insert the second CNOT gate as shown in Fig. 15, the sub-circuit G'' in Fig. 15 should realize the function whose Kmap is as shown in Fig. 16. This is because the second gate moves the minterm (1001) to (1101), and (1011) to (1111). Thus G'' should implement the function in Fig. 16, which can be done using one Toffoli gate.

Note that the resulting states of qubits after the circuit in Fig. 15 are not exactly the same as the ones in the desired circuit shown in Fig. 8, because we changed the functionality of x_2 and x_4 by inserting the two MPMCT gates. Therefore, we then insert the matching MPMCT gates after G'' at the end of the circuit, as shown in Fig. 17.

In this example, we can get the circuit shown in Fig. 18 whose quantum cost is 9. It should be noted that we get a circuit with a quantum cost of 20 for this example if we simply use the Arabzadeh method only.

3.2 How to Insert MPMCT Gates for a Good Pre-Optimization

The above-mentioned example shows that there is a case where inserting some MPMCT gates improves the quantum cost drastically. As we explained, inserting an MPMCT gate corresponds to the movement of 1-value cells in a Kmap, i.e., changing the specification of a given function. Thus, such a change of the specification can be considered as *pre-optimization* before generating a quantum circuit.

For simplicity, we consider an n-input function having k $(= 2^p)$ 1-value cells here. If k is small, we can try all the possible movements of 1-value cells as follows.

- We select one cell from all the k 1-value cells, and move all the other 1-value cells so that all k 1-value cells form a group of adjacent cells. As a result, the function is changed so that it can be realized using only one $MPMCT_{(n-p)}$ gate.
- The number of the above possible movements is $k! \times {}_nC_p$ by the following reason. First we have k possibilities to select one 1-value cell which we do not move. For example, let us consider a cell denoted by ① in Fig. 19 for the selected 1-value cell. Then, for the selected 1-value cell, we consider ${}_nC_p$ groups of k $(= 2^p)$ adjacent cells corresponding to product terms with $n-p$

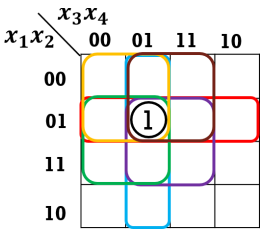

Fig. 19 Various grouping of adjacent cells for a product term with two literals

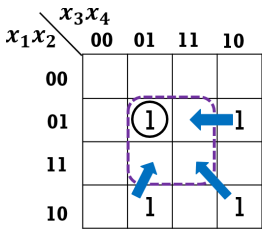

Fig. 20 An example of moving 1-value cells

literals of a n-variable function. As Fig. 19 shows, there are 6 ($= {}_4C_2$) such groups in this example where $n = 4$, $p = 2, k = 4$. We select one group from the ${}_nC_p$ groups, and then we move other $k - 1$ cells into the selected group of cells. We have $(k - 1)!$ of choices to move which 1-value cell to which cells in the group. For example, in Fig. 20, ① is the selected 1-value cell and the dotted area is the selected group of cells; we move $k-1$ ($= 3$) cells as the figure shows, and we have $(k-1)!$ possibilities of such movements. Thus, in total, all the possible movements is $k\times{}_nC_p\times(k-1)!$, which is equals to $(k)! \times {}_nC_p$.

Thus, if this number is not large, we can try all possible movements to find the best movement corresponding to the best insertion of MPMCT gates with the lowest quantum cost.

Our experimental results demonstrate that the above idea is potentially useful by considering a small function with $n = 4$ and $k = 4$ in Sect. 5.1. However, obviously, the above exhaustive search is not applicable to practical (i.e., large) functions; we have various heuristics for large functions.

4. A Heuristic Method Based on Our Idea

It would be a unique idea to make temporary changes to the functionality of a given Boolean function such that the modified function has a small ESOP expression, which results in a quantum circuit with reduced quantum cost. However, in order to utilize the idea for large practical functions, we have to come up with an excellent heuristic approach to avoid the above-mentioned exhaustive search. In this section, we present a heuristic approach to solve this problem.

First, let us explain how we can reduce the quantum cost by merging a pair of MPMCT gates. We will use the following "Merge Rules".

Merge Rule 1

If two MPMCT gates A and B have the same control bits and the same target bit but the polarity of only one control bit is different (i.e., negative and positive), then the two gates can be merged into one MPMCT gate.

For example, Fig. 21 shows two gates whose polarity of one control bit, x_i, is different. Then, the two MPMCT gates (Fig. 21) can be merged into one single MPMCT gate,

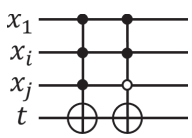

Fig. 21 Two MPMCT gates

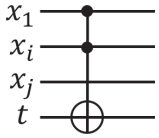

Fig. 22 After applying Merge Rule 1

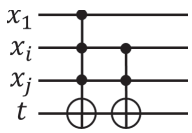

Fig. 23 Two MPMCT gates

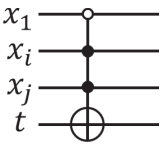

Fig. 24 After applying Merge Rule 2

as shown in Fig. 22. The control bit of different polarities can be omitted in the merged gate.

Merge Rule 2

If the control bits of a MPMCT gate, A, plus one additional bit, x_i, are the control bits of another MPMCT gate, B (in other words, if B has all the control bits of A as its control bits with one more control bit, x_i), then the two gates (A and B) can be replaced by one single MPMCT gate with the same control bits as B except for changing the polarity of x_i.

For example, Fig. 23 shows two gates where one gate has fewer control bits by one bit, compared to the other gate. These two gates can be merged into one gate of the same size as the larger gate, as shown in Fig. 24.

We should note that if the target bit of one gate is not the control of the other gate, the two MPMCT gates can be interchanged. By utilizing this property, we can change the order of gates so that we can use the above Merge Rules.

Based on the idea presented in Sect. 3, our strategy is to insert an MPMCT gate in order to make temporary changes to the given functionality so that we can apply the above Merge Rules to the changed function. The quantum cost of an MPMCT gate whose number of inputs is more than two, is relatively large and may substantially increase the cost when it is used. This is why we only consider using CNOT gates for making temporary changes to the functionality. We would also like to note that adding a CNOT gate changes the functionality so that some original MPMCT gates should be changed to the ones with a higher quantum cost. Thus, we first classify MPMCT gates based on how they should be changed by adding a CNOT gate to avoid the above-mentioned unwanted increase of the quantum cost.

Classification of MPMCT Gates.

In the following, we classify gates in a circuit according to their characteristics when we consider inserting a CNOT gate, g, whose control and target bits are x_c and x_t, respectively.

Type A: Type A gates have a control bit at x_t, and we should change the polarity of the control bit on x_t if we insert the CNOT gate g. In other words, Type A gates have a control bit on x_t and the same polarity control

bit on x_c as the inserted CNOT gate.

Type B: We do not need to change the polarity of any control bit of Type B gates even if we insert the CNOT gate g. There are two kinds of such gates: one has no control bit at x_t, and the other has control bit at x_c with the different polarity as g.

Type C: We consider a gate as a Type C gate if the gate has no control bit at x_c but has a control bit on x_t. When we insert g, the function is changed so that any Type C gate, g_c, should be changed to two MPMCT gates having one more control bit than g_c. Thus, the quantum cost for Type C gates should be increased drastically if we change the functionality by inserting g.

The following heuristic procedure tries to insert the best CNOT gate to reduce the quantum cost. The procedure applied the CNOT gate insertion to only a sub-circuit that does not contain Type C gates because Type C gates are not good for inserting a CNOT gate, as discussed above.

Procedure Inserting_Best_CNOT(G)

Here we want to reduce the quantum cost of a MPMCT-based circuit G to calculate a Boolean function of n-variable, consisting of m gates: $g_1, g_2, g_2, \cdots, g_m$. If a circuit has more than one output, we apply our procedure independently to each output, and combine them in the end.

Inserting_Best_CNOT(G) tries all the combination of i and j ($1 \leq, i, j \leq n$) as well as the polarities of the control bit (negative or positive) for the following circuit transformation, and chooses the best transformed circuit with the smallest quantum cost.

- For (i, j), insert a CNOT gate whose control and target bits are x_i and x_j at the beginning of the circuit G.
- Group the gates in G into Type A, B, and C for the inserted CNOT gate. Then, transform G to G' consisting of the following parts in this order from the beginning of the circuit.

  CNOT gate: The inserted CNOT gate whose control and target bits are x_i and x_j at the beginning.

  Sub-circuit A': A sub-circuit containing all Type A gates. We need to change the polarity of each control bit in x_j because of the property of Type A gates. Thus, we call this part Sub-circuit A' (not Sub-circuit A).

  Sub-circuit B: A sub-circuit containing all the Type B gates. These gates are not changed even if the CNOT gate is inserted before this part.

  CNOT gate: The inserted CNOT gate whose control and target bits are x_i and x_j to restore the change of the functionality.

  Sub-circuit C: A sub-circuit containing all the Type C gates. These gates are not changed because they are put outside of the pair of inserted CNOT gates.

- Because the polarity of the control bit (i.e., x_j) of each gate in Sub-circuit A' is changed, there is a chance that we can apply either Merge Rule 1 or Merge Rule 2 to some pairs of gates in the sub-circuit consisting of Sub-circuit A' and Sub-circuit B. We apply the rule repeatedly in an greedy manner such that we can reduce the cost as much as possible until there is no pair, to which the two merge rules can be applied.

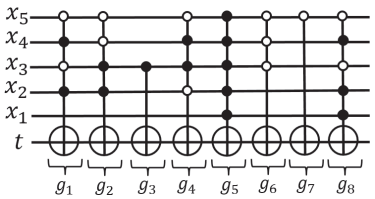

Fig. 25 An example of reversible circuit

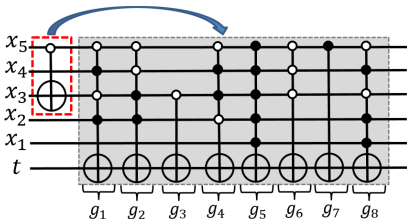

Fig. 26 Inserting a CNOT gate

Inserting_Best_CNOT(G) transform G into G' consisting of two sub-circuits (and additional two CNOT gates), i.e., one part of the G' consists of Sub-circuit A' and Sub-circuit B, and the other part is Sub-circuit C. We recursively apply the procedure to the two sub-circuits until we cannot get any good circuit transformation. (More precisely, we call Inserting_Best_CNOT(Sub-circuit A' + Sub-circuit B) and Inserting_Best_CNOT(Sub-circuit C) recursively.) The following example helps understand the proposed procedure.

Example 1: Let us consider an example of a reversible circuit shown in Fig. 25. Let $G = \{g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8\}$ be a set of MPMCT gates. If we insert an MPMCT gate whose negative control bit is x_5 and target bit is x_3 (i.e., CNOT gate) before G, as shown in Fig. 26, the inserted CNOT gate (the control bit is x_5 and target bit is x_3) inverts the value of x_3 when $x_5 = 0$.

This means that the CNOT gate inverts the input state of the control bit of g_1, g_2, g_4, g_6 and g_8, as shown in Fig. 27. For instance, g_2 works when the primary input values are $(-1100) = x_2\ x_3\ \overline{x_4}\ \overline{x_5}$ in the original function. But after we make temporary changes to the function by inserting the CNOT gate, the above input values are changed to $(-1000) = x_2\ \overline{x_3}\ \overline{x_4}\ \overline{x_5}$. Note that "-" refers to don't care which can be treated as either 1 or 0. Thus we change the polarity of the control bit of each Type A gate, as shown in Fig. 28.

In contrast, for g_5 and g_7, inserting the CNOT gate does not change the primary values when these gates are used. Thus, these gates are Type B gates, and they are not changed after inserting the CNOT gates, as shown in Fig. 29.

For this example, g_3 is a Type C gate (See Fig. 30). So,

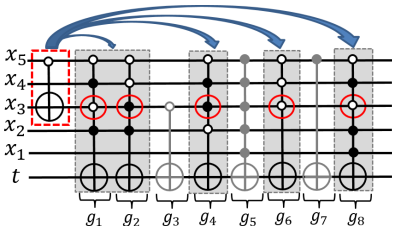

Fig. 27 Type A gates (shaded)

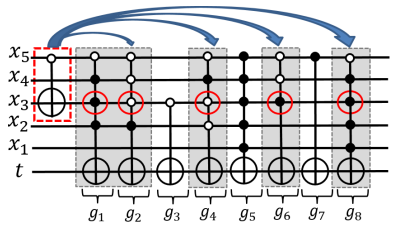

Fig. 28 Modified Type A gates after an MPMCT gate is inserted

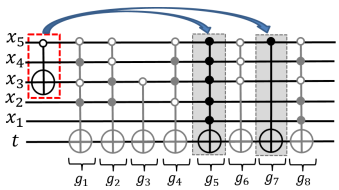

Fig. 29 Type B gates (shaded)

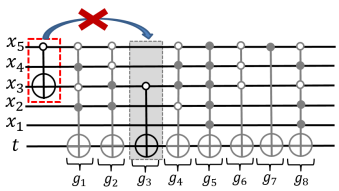

Fig. 30 Type C gate (shaded)

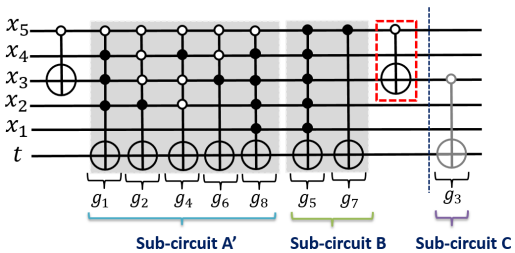

Fig. 31 Type C gate is put after the second CNOT gate to restore the original function

we do not want to change the input values for this gate by inserting the CNOT gate. Therefore, we put g_3 outside the effect of the functional change by the CNOT gate as shown in Fig. 31. Note that the second CNOT gate before g_3 in Fig. 31 is to restore the original function.

After changing the polarities of the control bit of Type A gates by considering the effect of inserted CNOT gate, as mentioned above, we try to find some pairs of gates of Type A and Type B, to which the merge rules can be applied. If there is a gate pair that satisfies the conditions of a merge rule, we take the best pair of gates (in terms of the cost reduction) and update the circuit according to the merge rule. For example, gates g_8 and g_5 satisfy the condition of Merge Rule 1 as shown in Fig. 32. Thus, we replace the pair of gates with a single gate g'_5, as shown in Fig. 33.

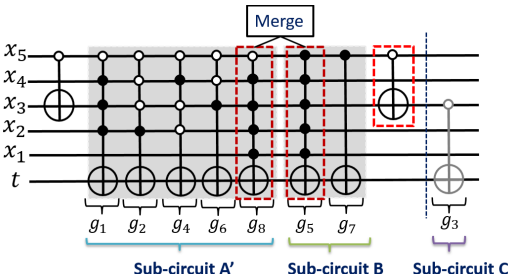

Fig. 32 A Pair of gates that satisfy Merge Rule 1

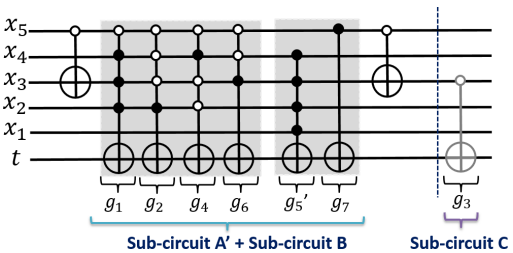

Fig. 33 After merging g_5 and g_8

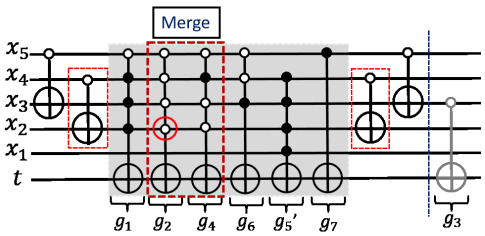

Fig. 34 Inserting the second pair of CNOT gates

Next, we apply our procedure Inserting_Best_CNOT again to the shaded sub-circuit, as shown in Fig. 33, and we insert another pair of CNOT gates (dashed boxes in the figure), as shown in Fig. 34 where we change the polarity of the control bit x_2 in gate g_2. Then, we try to find any pair of gates, which satisfy the merge rule conditions, and apply Merge Rule 1 to the modified gates g_2 and g_4. As a result, the pair of gates is replaced with a new single gate g'_2, as shown in Fig. 35.

Our procedure searches all possible gate pairs satisfying the merge rule conditions, and updates sub-circuits. The procedure continues until there is no reduction is possible. This way, we can get a circuit with a lower quantum cost.

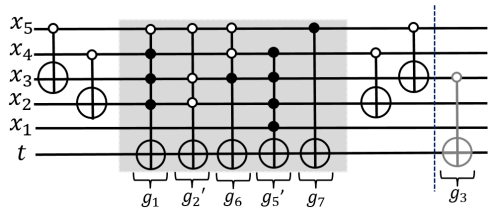

Fig. 35 After merging g_2 and g_4

Table 3 Preliminary Experimental Results

Quantum Cost	EXORCISM4 [10]	Proposed Pre-Optimization
0 – 20	292	1096
21 – 40	968	724
41 – 60	560	0
61 – 80	0	0
Average Cost	31.6	20.2

Table 4 Heuristic Experimental Results

Function	EXORCISM4 [10]	Proposed Heuristic	CPU time (s)	% Cost Reduction
z4ml	666	513	0.01	22.97
9symml	3,429	1,563	0.08	54.42
alu2	13,807	10,504	0.04	23.92
alu4	557,684	383,312	0.92	31.27
b1	32	15	0.00	53.13
vda	297,959	287,785	0.19	3.41
ttt2	57,207	56,447	0.02	1.33
apex6	3,248,463	3,068,226	4.30	5.55
cm82a	151	122	0.00	19.21
x2	822	757	0.00	7.91
4mod7_26	179	113	0.00	36.87
4_49_7	216	203	0.00	6.02
hwb4_12	266	182	0.00	31.58
rd32_19	30	21	0.00	30.00
rd53_68	318	198	0.00	37.74
rd73_69	1391	998	0.05	28.25
rd84_70	3558	2263	0.17	36.40
sym6_63	651	347	0.00	46.70
sym9_71	7439	4727	0.05	36.47
urf5_76	64,914	34,914	0.30	46.22
urf6_77	57338621	23905443	104.39	58.31
seq_201	159,851,461	97,757,257	0.88	38.84
cordic	55,152,422	20,286,205	2.86	63.22
too large	402,341,259,373	352,671,037,810	56.11	12.35

5. Experimental Results

5.1 Preliminary Experimental Results

To evaluate the potential effectiveness of our pre-optimization technique, we applied the method explained in Sect. 3 to all the 4-input functions with four minterms, i.e., we tried to find the best insertion resulting in the lowest quantum cost for each of ${}_{16}C_4 = 1820$ functions.

For comparison, we also tried to find a good quantum circuit for each function by finding a small ESOP expression. Note that this method should be essentially similar to designing reversible circuits by minimizing ESOP expressions, e.g., using the Arabzadeh method. To get small ESOP expressions, we used a state-of-the-art ESOP minimizer EXORCISM4 [10].

Table 3 shows the comparison between our method and the method based on the ESOP minimization by EXORCISM4 [10]. The table shows the number of functions (among all the 4-input functions with four minterms) generated by the two methods, divided into four groups in terms of the quantum costs. As can be seen from this table, our proposed method succeeds in designing most of the given quantum circuits with lower quantum costs. We confirmed that our proposed method achieves lower quantum cost for about 97.6% (1,776 Boolean functions) of all the functions compared to the EXORCISM4 program. In average, our method can reduce the cost by 36%, compared to the EXORCISM4 program.

5.2 Heuristic Experimental Results

In order to demonstrate the usefulness of our idea for large practical functions, we have proposed a heuristic method explained in Sect. 4.

We applied our algorithm to various benchmark circuits and compared our results with the quantum circuit obtained by using the ESOP minimizer, EXORCISM4 [10]. The result is shown in Table 4 where the second column refers to the quantum cost by the ESOP minimizer, and the third column refers to the quantum cost obtained by our proposed method, respectively. The fourth column reports the CPU time of our method. The outcome of the comparison clearly shows that the proposed method can significantly reduce the quantum cost in all cases.

6. Conclusion and Future Work

This paper proposes a new way to reduce the quantum cost by manipulating the ESOP expressions. The approach involves inserting MPMCT gates to make temporary changes to the functionality of Boolean functions followed by the application of conditional merge rules. In order to evaluate the proposed method, we had applied it to various benchmark circuits. The experimental results in Sect. 5 clearly show that the proposed method can significantly reduce the cost in almost all cases. Thus, the experiments demonstrated that our method not only produces a smaller ESOP expression but also leads to a lower quantum cost for quantum circuits.

We have only focused on demonstrating the usefulness of making temporary changes to the original functionality by inserting MPMCT gates. Currently, we use only MPMCT gates to design a quantum circuit. Thus, one possible direction of future work is to investigate how useful the proposed idea would be when applied to other quantum gates (i.e., controlled-V and controlled $V\dagger$, where V and $V\dagger$ are referred to as the square root of NOT gates; applying V twice and applying $V\dagger$ twice are both equivalent to a NOT operation).

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 24106009 and 15H01677.

References

[1] S. Yamashita, S. Minato, and D.M. Miller, "DDMF: An efficient

decision diagram structure for design verification of quantum circuits under a practical restriction," In IEICE Trans. Fundamentals, vol.E91-A, no.12, pp.3793–3802, Dec. 2008.
[2] L.K. Grover, "A fast quantum mechanical algorithm for database search," In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp.212–219, 1996.
[3] M. Arabzadeh, M. Saeedi, and M.S. Zamani, "Rule-based optimization of reversible circuits," In Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific, pp.849–854, IEEE 2010.
[4] K. Datta, G. Rathi, I. Sengupta, and H. Rahaman, "An improved reversible circuit synthesis approach using clustering of ESOP cubes," J. Emerg. Technol. Comput. Syst., vol.11, no.2, pp.1–16, Nov. 2014.
[5] K. Fazel, M.A. Thornton, and J.E. Rice, "ESOP-based Toffoli gate cascade generation," In IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp.206–209, citeseer, 2007.
[6] N.A.B. Adnan, K. Kushida, and S. Yamashita, "A pre-optimization technique to generate initial reversible circuits with low quantum cost," In IEEE International Symposium on Circuits and Systems (ISCAS), pp.2298–2301, May 2016.
[7] M. Saeedi and I.L. Markov, "Synthesis and Optimization of Reversible Circuits—a survey," ACM Comput. Surv., vol.45, no.2, March 2013.
[8] Z. Sasanian and D.M. Miller, "Reversible and quantum circuit optimization: A functional approach," In Reversible Computation 4th International Workshop, RC 2012, Copenhagen, Denmark, July 2-3, 2012, Revised Papers, vol.7581, pp.112–124, 2013.
[9] X. Cheng, Z. Guan, W. Wang, and L. Zhu, "A simplification algorithm for reversible logic network of positive/negative control gates," In Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference, pp.2442–2446, May 2012.
[10] Two-level exclusive sum-of-product minimization, http://web.cecs.pdx.edu/˜alanmi/research/min/minESOP.htm.

Nurul Ain Binti Adnan is currently a third year Ph.D student pursuing her Ph.D. degree at Graduate School of Information Science and Engineering, Ritsumeikan University, Japan. She received her B.E., degree in Electrical and Electronic Engineering in 2012 and M.E., degree in Electronic Engineering in 2014 from Graduate School of Engineering, Okayama University of Science, Japan. Her research interests include quantum circuit design, quantum cost and synthesis of topological quantum computation.

Shigeru Yamashita is a professor of Department of Computer Science, College of Information Science and Engineering, Ritsumeikan University. He received his B. E., M. E. and Ph.D. degrees in information science from Kyoto University, Kyoto, Japan, in 1993, 1995 and 2001, respectively. His research interests include new types of computation and logic synthesis for them. He received the 2000 IEEE Circuits and Systems Society Transactions on Computer-Aided Design of Integrated Circuits and Systems Best Paper Award, SASIMI 2010 Best Paper Award, 2010 IPSJ Yamashita SIG Research Award, and 2010 Marubun Academic Achievement Award of the Marubun Research Promotion Foundation. He is a senior member of IEEE, and a member of IPSJ.

Alan Mishchenko received the M.S. degree from the Moscow Institute of Physics and Technology, Moscow, Russia, in 1993, and the Ph.D. degree from the Glushkov Institute of Cybernetics, Kiev, Ukraine, in 1997. From 1998 to 2002, he was a Visiting Scientist at Portland State University, Portland, OR, USA. Since 2002, he has been with the EECS Department at University of California, Berkeley, CA, USA, where he is currently a Full Research Engineer. His research interests include developing computationally efficient methods for synthesis and verification. He was a recipient of the D.O. Pederson TCAD Best Paper Award in 2008, and the SRC Technical Excellence Award in 2011, for his work on ABC.