IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

VOL. E101-A NO. 12
DECEMBER 2018

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER *Special Section on VLSI Design and CAD Algorithms*

Function Design for Minimum Multiple-Control Toffoli Circuits of Reversible Adder/Subtractor Blocks and Arithmetic Logic Units

Md Belayet ALI[†a)], ***Nonmember***, **Takashi HIRAYAMA**[†], **Katsuhisa YAMANAKA**[†], ***and* Yasuaki NISHITANI**[†], ***Members***

SUMMARY In this paper, we propose a design of reversible adder/subtractor blocks and arithmetic logic units (ALUs). The main concept of our approach is different from that of the existing related studies; we emphasize the function design. Our approach of investigating the reversible functions includes (a) the embedding of irreversible functions into incompletely-specified reversible functions, (b) the operation assignment, and (c) the permutation of function outputs. We give some extensions of these techniques for further improvements in the design of reversible functions. The resulting reversible circuits are smaller than that of the existing design in terms of the number of multiple-control Toffoli gates. To evaluate the quantum cost of the obtained circuits, we convert the circuits to reduced quantum circuits for experiments. The results also show the superiority of our realization of adder/subtractor blocks and ALUs in quantum cost.

key words: *reversible function, reversible adder/subtractor, reversible arithmetic logic unit, incompletely-specified function, operation assignment, quantum cost, multiple-control Toffoli*

1. Introduction

The arithmetic logic unit is a key element in programmable computing devices. Several designs of ALUs were investigated based on reversible logic [1]–[5]. Some researchers also proposed new reversible gates to design ALU blocks [2]. In the design of ALUs, an adder/subtractor block is another important key elements. A faster adder/subtractor block will increase the efficiency of the ALU performance as well as that of the whole system. Recently, several designs have been proposed to construct reversible adder/subtractor blocks [6]–[16]. The objective of the researchers was to design ALUs or adder/subtractor blocks that depict higher performance and lower-power consumption than that of the existing system. Reversible computing is generally considered to be an unconventional form of computing, which has drawn considerable attention from researchers in order to design low-power computing devices [17], [18]. It is, therefore, important to have fast reversible ALUs and adder/subtractor blocks. Their performance could affect the efficiency of the whole system.

Recently, several studies have investigated the reversible circuit synthesis [19]–[22]. Figure 1 is an overview of the design flow of reversible synthesis. The primary goal for designing cost-efficient reversible circuits is to minimize the cost of the circuits in terms of a few metrics such as the number of constant inputs, garbage outputs [23], [24], reversible gates [25], and the quantum cost (QC) [26]–[29]. Constant inputs are lines that exist on the input side with a certain fixed logical value, either 0 or 1, whereas garbage outputs exist on the output side and do not perform any useful operation to facilitate further computations. Both the constant inputs and garbage outputs are simply added to the circuit to maintain reversibility at the design stage of reversible functions. From the published studies, we observe that multiple-control Toffoli (MCT) gates can be extensively used to synthesize a reversible circuit [30]–[32]. Therefore, the number of MCT gates is generally used as the cost metric during the stage of reversible circuit synthesis. A reversible circuit is further decomposed into cascades of elementary quantum gates [33]–[35], which is called a quantum circuit. At the stage of the quantum circuit synthesis, the QC is the most common cost metric, which is measured by counting the number of elementary quantum gates required to implement the quantum circuit.

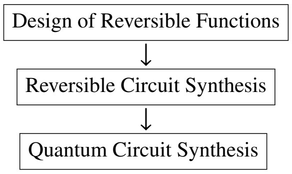

Design of Reversible Functions
Reversible Circuit Synthesis
Quantum Circuit Synthesis

Fig. 1 Overview of the design flow of reversible synthesis.

In this paper, we adopt the MCT gate library to design reversible adder/subtractor and simple ALU circuits. According to our knowledge, no optimization of reversible full adder/subtractor circuits using only the MCT gate library with the lowest possible number of working lines, constant inputs, and garbage outputs has been proposed even though the MCT gate library is the most fundamental and widely-used gate library for the synthesis of reversible circuits. Rangaraju et al. [12] have proposed three designs for reversible half and full adder/subtractor circuits using a different gate library including Feynman [36], Fredkin, TR, and Peres gates. In some of the cases, their proposed designs required five or more working lines, a large number of reversible gates, constant inputs, and garbage outputs with a high QC. Moghimi et al. [16] proposed a new 4x4 universal reversible gate as a cost-efficient full adder/subtractor in terms of the reversible and quantum metrics. They showed

Manuscript received March 13, 2018.
Manuscript revised June 20, 2018.

[†]The authors are with the Department of Electrical Engineering and Computer Science, Iwate University, Morioka-shi, 020-8551 Japan.
a) E-mail: belayet@kono.cis.iwate-u.ac.jp

DOI: 10.1587/transfun.E101.A.2231

some improvement over the existing full adder/subtractor design by comparing their design in terms of the number of reversible gates, garbage outputs, constant inputs and the QC. Sultan et al. [5] have proposed a full adder/subtractor circuit using Feynman and Peres gates, which showed the improvement over the previous work [12] in terms of the reversible gate counts, the garbage outputs, the constant inputs and the QC. Gupta et al. [37] proposed a reversible LU with eight operations using five inputs. The Mini-ALU reported in Revlib [38] can perform four operations: OR, AND, ADD, Identity. Both ALUs were implemented with the MCT gate library. As we have seen in Fig. 1, the design flow of reversible synthesis consists of three stages. We found that related works for the reversible adder/subtractors and ALUs lacked the analysis on the level of the reversible function design. Our approach is to investigate the function design thoroughly to improve the reversible circuits. The minimization problem for MCT circuits is also extended accordingly. As a result, we obtain the minimum MCT circuits of the adder/subtractors and the ALUs. We also give the quantum circuits of our reversible adder/subtractors. There were very few works that proposed the quantum circuit implementation for the adder/subtractors while obtaining the quantum circuit is the goal of the reversible synthesis.

This paper is organized as follows. The next section outlines some preliminaries of reversible and quantum circuits. In Sect. 3, we propose design techniques specific to the reversible functions of adder/subtractor blocks and ALUs. Those minimum MCT circuits and reduced quantum circuits are given in Sects. 4 and 5, respectively. We conclude in Sect. 6.

2. Preliminaries

This section introduces necessary terminology of the reversible and quantum circuits.

2.1 Reversible Functions

The function $f(x_1, x_2, ..., x_n)$ of n Boolean variables is called reversible if the number of outputs is equal to that of inputs and if any input pattern can be mapped to a unique output pattern [19], [23]. The reversibility is valued since no information is lost in the logic operations. Among the conventional logic operations such as AND, OR, and NOT, the NOT operation is reversible; it has a one-to-one correspondence between the input and output. The AND and OR operations are not reversible; they give a single output for two inputs. To convert the irreversible function to a reversible one, garbage outputs are used to distinguish equal output patterns, and constant inputs are added to equalize the number of inputs and outputs of the function.

2.2 Reversible Gates

Several reversible gates have been proposed during the last few decades, from which the Toffoli, Peres, Feynman, and Fredkin are conventionally used to synthesize reversible circuits. Recently, researchers have proposed different gate libraries such as MCT; multiple-control Fredkin (MCF); Peres (P); NOT, CNOT, and Toffoli (NCT); multiple-control Toffoli and Peres (MCT+P); multiple-control Toffoli and multiple-control Fredkin (MCT+MCF). Among them, the MCT gate library is most widely used for reversible logic synthesis. For this reason, we use the MCT gate library to design reversible circuit.

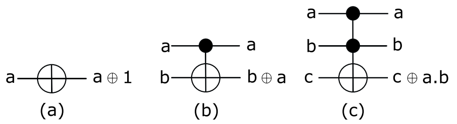


Fig. 2 MCT gate library (a) NOT, (b) CNOT, (c) Toffoli.

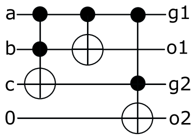


Fig. 3 Reversible circuit.

Definition 1: A multiple-control Toffoli (MCT) gate has $(k-1)$ control lines $\{x_1, x_2, \ldots, x_{k-1}\}$ and one target line x_k, whose function is a mapping from $(x_1, x_2, \ldots, x_{k-1}, x_k)$ to $(x_1, x_2, \ldots, x_{k-1}, (x_1 x_2 \cdots x_{k-1}) \oplus x_k)$. In the case $k = 1$, the gate maps x_1 to $1 \oplus x_1$.

In this paper, an MCT gate with $(k-1)$ controls is denoted by Toffoli-k. Toffoli-1 and Toffoli-2 are also called NOT and Controlled-NOT (CNOT), respectively. Toffoli-3 is the original Toffoli gate. The schematic of MCT gates are depicted in Fig. 2. The control lines are denoted by black dots (●), whereas the target line is denoted by ⊕.

2.3 Reversible Circuits

A reversible function can be realized by cascading the reversible gates where fan-out and feedback are not directly allowed in the realization of a reversible circuit. In reversible circuits, each variable of the function is represented using a circuit line. A Boolean function that is not reversible can be transformed into a reversible function by adding extra-working lines to ensure reversibility. The extra inputs in a reversible function can be pre-set to a constant value, which can be either 0 or 1. The extra outputs are referred to as garbage outputs. Figure 3 shows a reversible circuit with three reversible gates, one constant input (0), two garbage outputs (g_1, g_2), and two outputs (o_1, o_2). The minimum number of reversible gates, constant inputs, and garbage outputs are the properties of good quality of reversible circuits. In this paper, we focus on the minimization of the MCT gate count in reversible circuits.

2.4 Quantum Logic

The logic representation in quantum computation is quite different from the logic representation in the classical computation. The basic unit of information in quantum computation is a qubit represented by a state vector. The states $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ or $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$ are known as the computational basis states. The state of an arbitrary qubit $\alpha\,|0\rangle+\beta\,|1\rangle$ is described by the vector $\begin{pmatrix}\alpha\\\beta\end{pmatrix}$, where α and β are complex numbers that satisfy the constraint $|\alpha|^2 + |\beta|^2 = 1$. The measurement of a qubit results in either 0 with probability $|\alpha|^2$ or 1 with probability $|\beta|^2$. Similarly a generalized two qubit state can be described [39] as

$$|\Psi\rangle = \lambda_1\,|00\rangle + \lambda_2\,|01\rangle + \lambda_3\,|10\rangle + \lambda_4\,|11\rangle = \begin{pmatrix}\lambda_1\\\lambda_2\\\lambda_3\\\lambda_4\end{pmatrix},$$

where $\lambda_1\lambda_4 = \lambda_2\lambda_3$ for non-entanglement.

On the contrary, a classical bit has a state either 0 or 1, which is analogous to the measurement of a qubit state either $|0\rangle$ or $|1\rangle$ respectively. The main difference between bits and qubits is that a bit can be in either state 0 or 1 whereas a qubit can be in a superposition of $|0\rangle$ and $|1\rangle$.

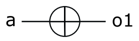
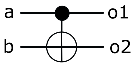
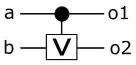
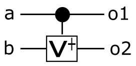
2.5 Quantum Gates

Quantum gates are the building blocks of quantum circuits, like classical logic gates such as AND, OR, and NOT are for conventional digital circuits. Many quantum gates have been defined and studied, but we concentrate on the elementary quantum gates NOT, CNOT, Controlled-V, and Controlled-V†, also known as the quantum primitives. This set of gates is known as the NCV gate library. These gates have been widely used to synthesize binary reversible functions. The elementary gates are represented by their unitary matrices. A gate which acts on k qubits is represented by a $2^k \times 2^k$ unitary matrix [40], which may include complex elements, as shown in Table 1.

The 2-line controlled-V gate changes the target line using the transformation defined by the matrix $V = \frac{1+i}{2}\begin{pmatrix}1 & -i\\-i & 1\end{pmatrix}$ if the single control line has the value 1. The 2-line controlled-V† gate changes the target line using the transformation defined by the matrix $V^\dagger = V^{-1} = \frac{1-i}{2}\begin{pmatrix}1 & i\\i & 1\end{pmatrix}$ if the single control line has the value 1. Gates V and V† are referred to as square-root-of-NOT gates since $V^2 = (V^\dagger)^2 = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$.

Controlled-V and Controlled-V† are the inverses of each other, whereas NOT and CNOT (generally MCT gates) are the self-inverse gates. Two adjacent Controlled-V (or Controlled-V†) gates with the same target and control can be replaced by a CNOT. Any primitive among CNOT, Controlled-V, and Controlled-V† can be represented by the pair of other primitives, which is shown in Fig. 4. It is referred to as the splitting rule. The inverse of the splitting rule is referred to as the merging rule.

Table 1 Quantum gate symbols and unitary matrices [39].

Gate Name	Gate Symbols	Matrix
NOT		$\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$
CNOT		$\begin{pmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{pmatrix}$
Controlled-V		$\begin{pmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & \frac{(1+i)}{2} & \frac{(1-i)}{2}\\0 & 0 & \frac{(1-i)}{2} & \frac{(1+i)}{2}\end{pmatrix}$
Controlled-$V^\dagger$		$\begin{pmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & \frac{(1-i)}{2} & \frac{(1+i)}{2}\\0 & 0 & \frac{(1+i)}{2} & \frac{(1-i)}{2}\end{pmatrix}$

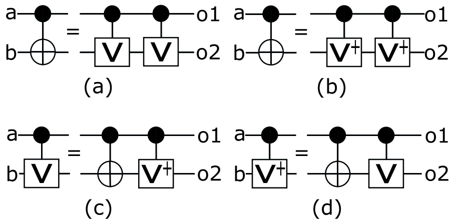


Fig. 4 Splitting and merging rules.

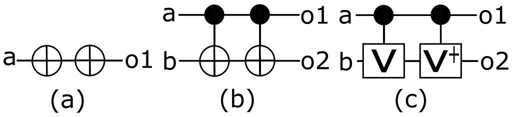


Fig. 5 Deletion rule.

If two adjacent gates form the identity function, then they can be deleted, which is known as the deletion rule in quantum primitives. Therefore, two NOT gates, two CNOT gates, and an adjacent (Controlled-V, Controlled-V†) pair (any order) with the same target and control can be eliminated, which is depicted in Fig. 5.

The mobility of gates is determined using the following property [25], which is called the moving rule.

Property 1 (Moving rule): Two adjacent gates g_1 and g_2 with controls c_1 and c_2 and targets t_1 and t_2 can be interchanged if $t_2 \notin c_1$ and $t_1 \notin c_2$.

Example 1: An elementary quantum gate has zero or one control and one target. Only a NOT gate has no control. In Fig. 7(a), gates g_1 and g_2 with controls $\{a\}$ and $\{b\}$ and targets c and c, respectively, can be interchanged because $c \notin \{a\}$ and $c \notin \{b\}$. Subsequently, gates g_1 and g_3 with controls $\{a\}$ and $\{a\}$ and targets c and b, respectively, can be interchanged because it satisfies the same condition. Thus, the gate g_1 can be moved to any other location in the circuit.

The deletion and merging rules directly reduce the quantum gates in a circuit. The moving rule is used to enhance the applicability of the deletion and merging rules.

Hung et al. [41] showed that when both CNOT and

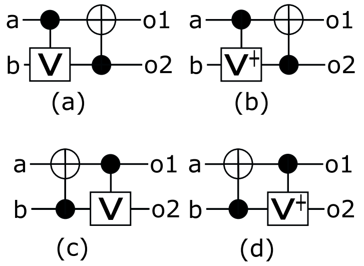


Fig. 6 Merged 2-qubit gate.

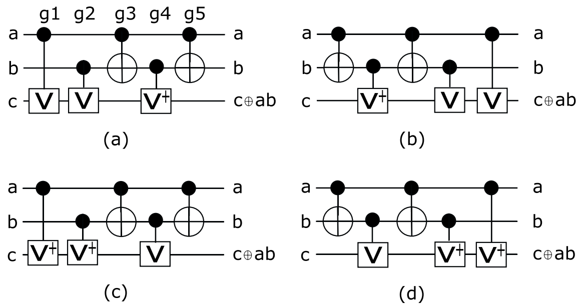


Fig. 7 A quantum circuit.

Controlled-V (or Controlled-V†) are operating on the same two qubits in a symmetric pattern as shown in Fig. 6, their total quantum cost is considered as one as well. We call it the 2-qubit gate rule in the quantum cost metrics.

2.6 Quantum Circuits

A quantum circuit is a model for quantum computation and can be realized by cascading the quantum gates. Quantum operations are all reversible, and every classical reversible circuit may be implemented in quantum technology. The number of elementary quantum gates required for implementing a reversible circuit is referred to as the QC of a reversible circuit. The QC is an important parameter to determine the quality of the quantum circuits as well as that of the reversible circuits. There may be multiple quantum circuit realizations for a reversible circuit. For example, Fig. 7 shows four realizations of the Toffoli-3 gate. Fig. 7(b) is the reverse representation of Fig. 7(a). Fig. 7(c) and (d) are obtained by interchanging the Controlled-V and Controlled-V† in Fig. 7(a) and (b). The QC of every circuit in Fig. 7 is five as it consists of five quantum gates. However, during the transformation of MCT circuits to quantum circuits, the QC of the MCT circuits may differ according to the selection of the quantum gate realization of Toffoli gates. For example, consider the MCT circuit shown in Fig. 3. The circuit has two Toffoli-3 gates. If the two Toffoli gates are replaced by the realization shown in Fig. 7(a), the QC of the circuit will be 9. But if the first Toffoli gate is replaced by the realization shown in Fig. 7(a), and if the second gate is replaced by Fig. 7(b), the QC will be 8. The selection of the appropriate realization of the Toffoli gate will not affect the functionality of the circuits, however it may provide a better quality of the circuits in terms of the quantum gate count. This property is used in the reduction of the QC in Sect. 5.1.

3. Design of Reversible Functions

The unique and interesting point in this paper is our idea of improvement of reversible circuits. Different from the related works, we try improvements on the level of the function design. Our approach of investigating the reversible functions includes the following:

- Embedding irreversible functions into incompletely-specified reversible functions,
- Operation assignment,
- Permutation of the function outputs.

Moreover, we provide a few extensions of these techniques to deal with a set of incompletely-specified reversible functions.

3.1 Embedding Irreversible Functions into Incompletely-Specified Reversible Functions

The functions of the adder/subtractor and ALU are irreversible. To obtain a reversible circuit of an irreversible function, first, we embed the irreversible function into an incompletely-specified function to make it reversible. In this paper, we set a restriction in which only the minimum necessary ancilla lines are added for embedding. Consider the functional specification of ALU in Table 2, where '-' denotes the don't-care value. We added two garbage outputs to the irreversible function to embed it into a reversible function. The garbage outputs are regarded as the columns of the don't-care values in the truth table. Therefore, this function is called an incompletely-specified reversible function. The required garbage outputs depends on the maximum number of repetition of an output pattern in the truth table. If the number of repetition is M, $\lceil \log_2 M \rceil$ garbage outputs are necessary [24]. In Table 2, each of the output pattern (0) and (1) is repeated four times for the input $\{0100, 1000, 1001, 1010\}$ and $\{0101, 0110, 0111, 1011\}$, respectively, which is the maximum number of repetitions. Using $\lceil \log_2 4 \rceil = 2$, two garbage outputs are added. However, no additional lines are required in this reversible circuit since the number of outputs is equal to that of inputs.

Generally, an irreversible function can be embedded in a reversible function by adding the necessary garbage outputs, and the resulting reversible function is incompletely specified. To discuss about the incompletely-specified functions, some definitions are given. Note that 'a reversible function' indicates a fully-specified reversible function hereafter, unless otherwise noted.

Definition 2: An incompletely-specified reversible function is abbreviated to an *ISRF*. Let F be a (fully-specified) reversible function. We say that *F matches an ISRF F'* if $F(\boldsymbol{X}) = F'(\boldsymbol{X})$ for all inputs $\boldsymbol{X} \in \{0, 1\}^n$ except the case $F'(\boldsymbol{X}) =$ '-', where '-' is the don't-care value. $F \in: F'$ denotes that F matches F'. As an extension, for a set of ISRFs

Table 2 An incompletely-specified function ALU.

S_1	S_2	A	B	g_1	g_2	O_1	O_2
0	0	0	0	-	-	0	0
0	0	0	1	-	-	0	1
0	0	1	0	-	-	0	1
0	0	1	1	-	-	1	0
0	1	0	0	-	-	-	0
0	1	0	1	-	-	-	1
0	1	1	0	-	-	-	1
0	1	1	1	-	-	-	1
1	0	0	0	-	-	-	0
1	0	0	1	-	-	-	0
1	0	1	0	-	-	-	0
1	0	1	1	-	-	-	1
1	1	0	0	-	-	0	0
1	1	0	1	-	-	1	1
1	1	1	0	-	-	0	1
1	1	1	1	-	-	0	0

Table 3 Four operations of ALU.

	A	B	g_1	g_2	O_1	O_2
ADD: $F_0(A,B) =$	0	0	-	-	0	0
	0	1	-	-	0	1
	1	0	-	-	0	1
	1	1	-	-	1	0

	A	B	g_1	g_2	O_1	O_2
OR: $F_1(A,B) =$	0	0	-	-	-	0
	0	1	-	-	-	1
	1	0	-	-	-	1
	1	1	-	-	-	1

	A	B	g_1	g_2	O_1	O_2
AND: $F_2(A,B) =$	0	0	-	-	-	0
	0	1	-	-	-	0
	1	0	-	-	-	0
	1	1	-	-	-	1

	A	B	g_1	g_2	O_1	O_2
SUB: $F_3(A,B) =$	0	0	-	-	0	0
	0	1	-	-	1	1
	1	0	-	-	0	1
	1	1	-	-	0	0

$\mathcal{F}$, $F \in: \mathcal{F}$ denotes that $F \in: F'$ holds for some ISRF F' in $\mathcal{F}$. We also say that F *matches* $\mathcal{F}$ if $F \in: \mathcal{F}$.

The minimality, in this paper, is defined by the gate count of MCT circuits. Below, an extension of the minimality to a set of ISRFs is given.

Definition 3: Let F be a reversible function of n variables. Among all n-bit MCT circuits that realize F, those with the exact minimum number of MCT gates are called *the minimum MCT circuits of F*. *The size of* F is defined by the gate count of the minimum MCT circuit of F, which is denoted by $\gamma(F)$. We extend γ to that for a set of ISRFs $\mathcal{F}$; $\gamma(\mathcal{F})$ is defined by the minimum $\gamma(F)$ for all $F \in: \mathcal{F}$, namely, $\gamma(\mathcal{F}) = \min\{\gamma(F) \mid F \in: \mathcal{F}\}$. $OPT(\mathcal{F})$ is the set of reversible functions $F \in: \mathcal{F}$ with the minimum size $\gamma(\mathcal{F})$; $OPT(\mathcal{F}) = \{F \mid F \in: \mathcal{F}, \gamma(F) = \gamma(\mathcal{F})\}$.

3.2 Operation Assignment

In this work, we introduce a new approach called synthesis with operation assignment. It is a permutation of groups of rows in the truth table. This approach is applicable to those circuits which have more than one operations such as adder/subtractor and arithmetic logic unit. This type of the circuits has the selector bits to choose the desired operation. Even if the set of operations is the same, the functions consisting of those operations vary according to the assignment of the selector bits to the operations. Our approach is to try all permutations of operations and find the minimum circuit realization. If a function has m selector bits, $2^m!$ variants are considered. To utilize the concept of the permutation of operations, we give a formal definition of an equivalence relation on the operation assignment.

Definition 4: Suppose that a multiple-output function F has m selector bits $\{S_1, S_2, \ldots, S_m\}$ and is represented by $F(S_1, S_2, \ldots, S_m, \boldsymbol{X})$. F may be an ISRF. For a non-negative integer i $(0 \le i \le 2^m - 1)$, $F_i(\boldsymbol{X})$ denotes the cofactor $F(i_1, i_2, \ldots, i_m, \boldsymbol{X})$, where $(i_1, i_2, \ldots, i_m)$ is an m-bit binary representation of i. The cofactors $F_i(\boldsymbol{X})$ are called operations of $F(S_1, S_2, \ldots, S_m, \boldsymbol{X})$. Literals $\bar{S}$ and S are denoted by S^0 and S^1, respectively. By using these notations, the Shannon expansion of F with respect to variables $S_1, S_2, \ldots, S_m$ is represented by $F(S_1, S_2, \ldots, S_m, \boldsymbol{X}) = \bigvee_{0 \le i \le 2^m-1} S_1^{i_1} S_2^{i_2} \cdots S_m^{i_m} \cdot F_i(\boldsymbol{X})$. We say that a function defined by $\bigvee_{0 \le i \le 2^m-1} S_1^{i_1} S_2^{i_2} \cdots S_m^{i_m} \cdot F_{\pi(i)}(\boldsymbol{X})$ *is OA-equivalent to* $F(S_1, S_2, \ldots, S_m, \boldsymbol{X})$ if π is a permutation of $\{0, 1, \ldots, 2^m - 1\}$. The set of all functions OA-equivalent to F is called *the OA-equivalence class of F* and denoted by $OAEC(F)$, in which the selector bits $\{S_1, S_2, \ldots, S_m\}$ of F is assumed to be specified in the definition of F.

m selector bits can identify up to 2^m operations. Therefore, for a function with m selector bits, there exist $2^m!$ OA-equivalent functions at most by the permutation of 2^m operations.

Property 2: For a function F with m selector bits, $|OAEC(F)| \le 2^m!$.

For a function F and a circuit C, it is commonly said that C realizes F if the function of C is equal to F. We also extend the concept of circuit realization from a reversible function to a set of ISRFs.

Definition 5: Let C be a reversible circuit and $\mathcal{F}$ a set of ISRFs. We say that *C realizes $\mathcal{F}$* if the function of C matches $\mathcal{F}$.

Example 2: Consider the function of ALU shown in Table 2, in which S_1 and S_2 are the selector bits. Table 3 shows the four operations $F_0, \ldots, F_3$ of ALU. With these operations, ALU is represented by $\bar{S}_1\bar{S}_2F_0(\boldsymbol{X}) \vee \bar{S}_1S_2F_1(\boldsymbol{X}) \vee S_1\bar{S}_2F_2(\boldsymbol{X}) \vee S_1S_2F_3(\boldsymbol{X})$. For example, the function ALU' = $\bar{S}_1\bar{S}_2F_1(\boldsymbol{X}) \vee \bar{S}_1S_2F_0(\boldsymbol{X}) \vee S_1\bar{S}_2F_3(\boldsymbol{X}) \vee S_1S_2F_2(\boldsymbol{X})$ made by a permutation of operations is OA-equivalent to ALU. The difference between ALU and ALU' is the assignment of

Table 4 Operation assignment for ALU.

S_1	S_2	ALU	ALU'
0	0	ADD	OR
0	1	OR	ADD
1	0	AND	SUB
1	1	SUB	AND

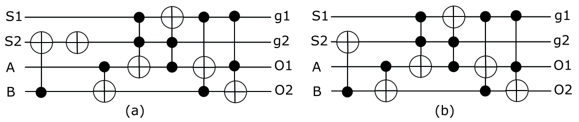


Fig. 8 (a) Circuit realizing ALU, (b) Circuit realizing $OAEC$(ALU).

operations, which is summarized in Table 4. A function OA-equivalent to ALU provides the same set of operations as ALU. However, the cost of their circuit realizations are different in general. A minimum MCT circuit of ALU with the original assignment in Table 4 is shown in Fig. 8(a) using seven reversible gates (one NOT gate, two CNOT, and four Toffoli-3 gates) with a quantum cost of 23. If we use a different assignment of operations as ALU', a different circuit realization using six gates (two CNOT and four Toffoli-3 gates) with a quantum cost of 22 is obtained as shown in Fig. 8(b). This example shows that permutation of operations has direct effect on the cost of the reversible circuit realization.

3.3 Permutation of Function Outputs

The idea of permutation of outputs of a reversible function has been introduced in the book by Wille et al. [42], which is called Synthesis with Output Permutation (SWOP). The following definition is the ISRF version of SWOP and its extension to a set of ISRFs.

Definition 6: For a given ISRF F, the set of variants made by the permutation of outputs in F is called *the P-equivalence class of* F, which is denoted by $PEC(F)$. We extend PEC so that it accepts a set of ISRFs $\mathcal{F}$: $PEC(\mathcal{F}) = \bigcup_{F\in\mathcal{F}} PEC(F)$.

The P-equivalence is a permutation of columns in the table whereas the OA-equivalence in the previous section is a permutation of groups of rows in the truth table.

$|PEC(F)|$ for an n-output ISRF F is $n!$ in maximum since $PEC(F)$ is a permutation of the outputs of F.

Property 3: For an n-output ISRF F, $|PEC(F)| \leq n!$.

Note that the P-equivalence class of a reversible function F is different from the conjugacy class [43] of F (also called the line reordering). In the conjugacy class, the inputs and outputs in F are relabeled simultaneously. In the P-equivalence class, the outputs in F are relabeled, but the inputs are not altered. It is known that every function in the conjugacy class of F can be realized in a circuit with the same gate count. However, the gate count of the circuit of a function in the P-equivalence class of F is not the same as that of F in general like Example 3. This means that there may be a smaller function in the gate count than F among functions matching $PEC(F)$.

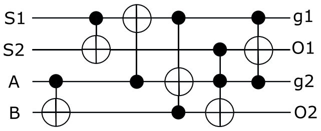


Fig. 9 Circuit realizing PEC(ALU).

Example 3: Consider that ALU shown in Table 2 maps the input (S_1, S_2, A, B) to the output (g_1, g_2, O_1, O_2). The minimum MCT circuit shown in Fig. 8(a) consists of seven gates. Figure 9 shows an MCT circuit realizing PEC(ALU), in which the four outputs of the function have been reordered to another position. More precisely, the MCT circuit shown in Fig. 9 maps the input (S_1, S_2, A, B) to the output (g_1, O_1, g_2, O_2). This reduces the overall number of gates from seven to six.

4. Minimum MCT Circuits

The two techniques $OAEC$ and PEC given in the previous section can be combined for a further reduction of gates in a circuit. In this section, we discuss the minimization of MCT circuits realizing $PEC(OAEC(F))$ for a given F. By Definition 6, PEC has been extended to accepting a set of ISRFs like $OAEC(F)$.

4.1 Minimization Algorithm

The adder/subtractor and ALUs that will be proposed in the later sections are 4-bit or 5-bit functions. The minimum MCT circuits for all the 4-bit reversible functions have been obtained by Golubitsky et al. [43]. Their algorithm, however, does not support ISRFs. A SAT-based algorithm [30] can be applied to ISRFs. The algorithm basically produces one minimum circuit for a given ISRF; however, it does not provide a list of minimum circuits for other reversible functions that match the ISRF. Instead of these sophisticated algorithms, we use the hash tables of minimum MCT circuits with up to seven gates for 4-bit functions and five gates for 5-bit functions. The tables are simply constructed using an exhaustive enumeration of gate combinations. The procedure MakeTable in Fig. 10 is used to construct such tables. Although the applicability of this simple strategy is limited to small circuits due to the massive consumption of memory, it is sufficient to achieve our purpose. In the algorithm, the number of input/output lines, or bits, are denoted as n, and minimum MCT circuits with i gates are stored in ht_i. The function FindOPT obtains $OPT(\mathcal{F})$ and its minimum MCT circuits in pairs by searching for $ht_1, ht_2, \ldots$ successively. A memory saving technique with symmetry [43] is used, however, the description of this technique is omitted from Fig. 10. We combine the hash tables of these minimum circuits and the techniques proposed in Sect. 3, to obtain the

```
var $ht_0, ht_1, ht_2, \ldots$ : Hash Table;
procedure MakeTable($m$)
                                                        ▹ Input: $m$ is an integer.
            ▹ Side Effect: minimum MCT circuits with up to $m$ gates are
    stored in the hash tables $ht_0, ht_1, ht_2, \ldots$.
    if $m = 0$ then $ht_0[identity] \leftarrow$ the empty circuit;
    else
        MakeTable($m - 1$);
        for all entry $C \in ht_{m-1}$ do
            for all $G \in \mathcal{G}_n$ do
                if $ht_m[func(C \mid G)] = \emptyset$ then
                    $ht_m[func(C \mid G)] \leftarrow C \mid G$;
                end if
            end for
        end for
    end if
end procedure

function FindOPT($\mathcal{F}$)
                                                  ▹ Input: $\mathcal{F}$ is a set of ISRFs.
            ▹ Output: $OPT(\mathcal{F})$ and its minimum MCT circuits in pairs.
    var $i \leftarrow 1$ : Integer;
    var $S \leftarrow \emptyset$ : Set;
    while ($S = \emptyset$) or ($ht_i$ is not empty) do
        for all key $F \in ht_i$ do
            if $F \in: \mathcal{F}$ then $S \leftarrow \{(F, ht_i[F])\} \cup S$;
            end if
        end for
        $i \leftarrow i + 1$;
    end while
    return $S$;
end function
```

$\mathcal{G}_n$: the set of all MCT gates with up to n bits.
$C \mid G$: the concatenation of circuit C and gate G.
$func(C \mid G)$: the reversible function of the circuit $C \mid G$.

Fig. 10 Simple minimization algorithm with up to m gates.

minimum MCT realization of the adder/subtractor and ALU circuits.

Both the SAT-based algorithm and FindOPT can obtain minimum MCT circuits, however, we have used FindOPT in this paper. We actually tried the SAT-based algorithm and compared the results with ours. Thus, we have confirmed that the list of minimum MCT circuits that were obtained by FindOPT includes a better circuit in terms of QC rather than the one that was obtained using the SAT-based algorithm.

4.2 Reversible Adder/Subtractor

In this section, we obtain the minimum MCT circuits of half and full adder/subtractors by using our proposed techniques. An adder/subtractor has the two operations of 'adder' and 'subtractor', and the 1-bit selector S decides whether either of these operations should be performed. The possible assignment of operations is shown in Table 5. Along with half and full adders, there are half and full adder/subtractors.

4.2.1 Half Adder/Subtractor

Table 5 Operation assignment for adder/subtractor.

S	Assignment 1	Assignment 2
0	Adder	Subtractor
1	Subtractor	Adder

Table 6 An incompletely specified half adder/subtractor function hAS_1.

c	S	A	B	g_1	g_2	C/B	S/D
0	0	0	0	-	-	0	0
0	0	0	1	-	-	0	1
0	0	1	0	-	-	0	1
0	0	1	1	-	-	1	0
0	1	0	0	-	-	0	0
0	1	0	1	-	-	1	1
0	1	1	0	-	-	0	1
0	1	1	1	-	-	0	0
1	0	0	0	-	-	-	-
1	0	0	1	-	-	-	-
1	0	1	0	-	-	-	-
1	0	1	1	-	-	-	-
1	1	0	0	-	-	-	-
1	1	0	1	-	-	-	-
1	1	1	0	-	-	-	-
1	1	1	1	-	-	-	-

Table 7 $OPT(PEC(\mathsf{hAS}_1))$, and its minimum MCT circuits.

Reversible Function	Minimum MCT Circuit
(0 3 6 13 4 15 2 1 8 11 14 5 12 7 10 9)	(2 1) (3 2) (1 3 0)
(0 3 2 13 4 15 6 1 8 11 10 5 12 7 14 9)	(2 3 1) (3 2) (1 3 0)
(0 3 2 9 4 15 6 5 8 11 10 1 12 7 14 13)	(2 3 0) (3 2) (1 3 0)
(0 1 7 14 4 13 3 2 8 9 15 6 12 5 11 10)	(2 1) (1 3 0) (2 3)
(0 1 3 14 4 13 7 2 8 9 11 6 12 5 15 10)	(2 3 1) (1 3 0) (2 3)
(0 1 3 10 4 13 7 6 8 9 11 2 12 5 15 14)	(2 3 0) (1 3 0) (2 3)

According to the operation assignment that is presented in Table 5, two OA-equivalent functions of the half adder/subtractor are observed to exist for Assignments 1 and 2. Equations (1) and (2) provide the logical expressions for these functions. 'C/B' and 'S/D' stand for Carry/Borrow and Sum/Difference, respectively.

$$\begin{cases} C/B = \bar{S}AB \vee S(AB \oplus B) \\ S/D = A \oplus B \end{cases} \tag{1}$$

$$\begin{cases} C/B = \bar{S}(AB \oplus B) \vee SAB \\ S/D = A \oplus B \end{cases} \tag{2}$$

Equation (1) is embedded in the truth table as in Table 6, which is denoted by hAS_1. While embedding, two garbage outputs, g_1 and g_2, are added because they are required by the maximum number of repetitions of an output pattern in the truth table. To balance the numbers of inputs and outputs, a constant input $c = 0$ is added. Similarly, the embedded Eq. (2) is denoted by hAS_2.

Table 7 shows the minimum MCT realization of the P-equivalence class of hAS_1, or $OPT(PEC(\mathsf{hAS}_1))$, which was obtained by our FindOPT program. The six reversible functions have minimum MCT circuits with three gates. On the right side of Table 7, a circuit is represented by a sequence of reversible gates, in which $(x\ y)$ represents a CNOT gate and $(x\ y\ z)$ represents a Toffoli gate. The last parameter is the target line, and the remaining parameters are the control lines. For example, "(2 1) (3 2) (1 3 0)" in the first row represents the MCT circuit of Fig. 11(a), where the numbering of the lines is $(c, S, A, B) = (0, 1, 2, 3)$. The QC of the MCT circuit is seven (Fig. 11(b)).

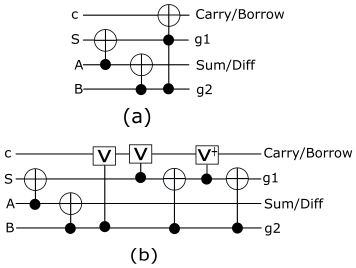


Fig. 11 The proposed half adder/subtractor: (a) the minimum MCT circuit, and (b) the reduced quantum circuit.

Table 8 $OPT(PEC(\mathsf{hAS}_2))$, and its minimum MCT circuits.

Reversible Function	Minimum MCT Circuit
(0 15 6 1 4 3 2 13 8 7 14 9 12 11 10 5)	(3 2) (2 1) (1 3 0)
(0 11 2 1 4 7 6 13 8 3 10 9 12 15 14 5)	(1 3 0) (3 2) (2 3 0)
(0 15 2 1 4 3 6 13 8 7 10 9 12 11 14 5)	(3 2) (2 3 1) (1 3 0)

Similar experiments were conducted on hAS_2. The results, however, were not better than those of hAS_1. The $OPT(PEC(\mathsf{hAS}_2))$ are shown in Table 8. Three functions have minimum MCT circuits with three gates.

Since the possible operation assignments for the half adder/subtractor are hAS_1 and hAS_2, we have $OAEC(\mathsf{hAS}_1) = \{\mathsf{hAS}_1, \mathsf{hAS}_2\}$. Based on the experiments that were performed using hAS_1 and hAS_2, we have confirmed that their minimum size is $\gamma(PEC(OAEC(\mathsf{hAS}_1))) = \gamma(PEC(\mathsf{hAS}_1)) = \gamma(PEC(\mathsf{hAS}_2)) = 3$; additionally, hAS_1 and hAS_2 can be realized by MCT circuits using two CNOT gates and one Toffoli gate.

4.2.2 Full Adder/Subtractor

A full adder/subtractor acts as a full adder or a full subtractor depending on the value of the selector bit s. As we have seen in Table 5, two full adder/subtractor functions are possible, whose logic expressions can be represented using Eqs. (3) and (4), respectively.

$$\begin{cases} C/B = \bar{S}(AB \vee AC \vee BC) \vee S(\bar{A}B \vee BC \vee \bar{A}C) \\ S/D = A \oplus B \oplus C \end{cases} \tag{3}$$

$$\begin{cases} C/B = \bar{S}(\bar{A}B \vee BC \vee \bar{A}C) \vee S(AB \vee AC \vee BC) \\ S/D = A \oplus B \oplus C \end{cases} \tag{4}$$

In a similar mannar to that of the half adder/subtractor, Eqs. (3) and (4) are embedded to ISRFs, and are denoted by fAS_1 and fAS_2, respectively. To balance the number of inputs and outputs, two garbage outputs g_1 and g_2 are added, however, no lines are added. fAS_1 and fAS_2 can be realized in 4-line MCT circuits.

In Table 9, the list of $OPT(PEC(\mathsf{fAS}_1))$ is presented; this list is obtained using our FindOPT program. It can be observed that 30 functions have minimum MCT circuits with five gates, i.e., $\gamma(PEC(\mathsf{fAS}_1)) = 5$. As an example of the minimum MCT circuit of the full adder/subtractor, "(1 0) (3 2) (2 1) (3 0) (0 2 3)" is shown in Fig. 12, where the numbering of the lines is $(S, A, B, C) = (0, 1, 2, 3)$. The circuit consists of four CNOT gates and one Toffoli gate. The reduced quantum circuit realization of the minimum MCT circuit of the proposed full adder/subtractor is shown in Fig. 12(b)

Although we have obtained $OPT(PEC(\mathsf{fAS}_2))$, the list has been omitted in this paper. This is because no results were observed to be better than that of $OPT(PEC(\mathsf{fAS}_1))$. In summary, $OPT(PEC(\mathsf{fAS}_2))$ consists of 26 functions, and $\gamma(PEC(\mathsf{fAS}_2)) = 5$. Thus, we have $\gamma(PEC(OAEC(\mathsf{fAS}_1))) = \gamma(PEC(\mathsf{fAS}_1)) = \gamma(PEC(\mathsf{fAS}_2)) = 5$.

4.3 Simple Reversible Arithmetic Logic Units

The operation assignment works more efficiently for the design of ALUs than adder/subtractors since an ALU has more operations assigned by the selector bits. This section reports the minimum MCT circuits of some benchmarks of simple reversible ALUs consisting of only four or eight 1-bit operations. It should be noted that, this is not the limitation of the operation assignment technique but the limitation of the minimizer. Instead of the exact minimizer, it is possible to use a fast heuristic synthesizer supporting ISRFs to obtain MCT circuits of larger ALUs without guaranteeing the minimality. In this paper, however, we focus on obtaining minimum MCT circuits. The minimum results are valuable as scientific and theoretical data in the related fields. To apply a heuristic method to the equivalence classes for larger ALUs [2] is our next task.

4.3.1 1-bit ALU Benchmarks

We applied our function design technique to 1-bit ALU benchmarks: Mini-ALU [38] and Gupta's-LU [37]. The embedding in the truth table of the reversible ALU is similar to that of the reversible adder/subtractor. In this paper, we skip the truth table representation of those reversible ALUs.

FindOPT program confirmed that $\gamma(PEC(OAEC(\text{Mini-ALU})) = 5$. The list of $OPT(PEC(OAEC(\text{Mini-ALU})))$ is omitted, but the number of it is $|OPT(PEC(OAEC(\text{Mini-ALU})))| = 266$. One of the minimum MCT circuits among them is shown in Fig. 13. The MCT circuit reported in Revlib [38] consists of six gates. We reduced the reversible gates from six to five.

For Gupta's-LU, $\gamma(PEC(OAEC(\text{Gupta's-LU}))) = 3$ and $|OPT(PEC(OAEC(\text{Gupta's-LU})))| = 48$. One of the minimum MCT circuits is shown in Fig. 14. Originally Gupta's-LU [37] was constructed by 18 reversible gates whereas our technique obtained circuits with only three reversible gates.

4.3.2 Revised ALUs

As seen in Sect. 4.2, addition and subtraction are essential operations in computation. It is preferable that an ALU has those two operations. Since Mini-ALU does not have subtraction, we propose a revised ALU, denoted by ALU,

Table 9 $OPT(PEC(\mathsf{fAS}_1))$ and its minimum MCT circuits.

Reversible Function	Minimum MCT circuit
(0 14 6 9 12 3 11 5 8 7 15 1 4 10 2 13)	(1 0) (3 2) (2 1) (3 0) (0 2 3)
(0 5 13 10 12 11 3 6 8 15 7 2 4 1 9 14)	(1 0) (2 3) (3 1) (2 0) (0 3 2)
(0 14 6 1 12 3 11 13 8 7 15 9 4 10 2 5)	(3 2) (1 0) (2 1) (2 3 0) (0 2 3)
(0 5 13 2 12 11 3 14 8 15 7 10 4 1 9 6)	(2 3) (1 0) (3 1) (2 3 0) (0 3 2)
(0 10 2 9 14 5 13 7 8 3 11 1 6 12 4 15)	(3 0) (1 0) (3 2) (0 2 3) (1 2)
(0 1 9 10 13 14 6 7 8 11 3 2 5 4 12 15)	(2 0) (1 0) (2 3) (0 3 2) (1 3)
(0 14 6 9 4 3 11 13 8 7 15 1 12 10 2 5)	(3 2) (1 2 0) (2 1) (3 0) (0 2 3)
(0 5 13 10 4 11 3 14 8 15 7 2 12 1 9 6)	(2 3) (1 3 0) (3 1) (2 0) (0 3 2)
(0 10 2 1 14 5 13 15 8 3 11 9 6 12 4 7)	(3 2) (1 0) (2 3 0) (0 2 3) (1 2)
(0 1 9 2 13 14 6 15 8 11 3 10 5 4 12 7)	(2 3) (1 0) (2 3 0) (0 3 2) (1 3)
(0 14 6 9 4 11 3 13 8 7 15 1 12 2 10 5)	(3 0) (3 2) (1 2 3) (2 1) (0 2 3)
(0 5 13 10 4 3 11 14 8 15 7 2 12 9 1 6)	(2 0) (2 3) (1 3 2) (3 1) (0 3 2)
(0 1 9 10 5 6 14 15 8 11 3 2 13 12 4 7)	(2 0) (2 3) (1 3 2) (0 3 2) (1 3)
(0 10 2 9 6 13 5 15 8 3 11 1 14 4 12 7)	(3 0) (3 2) (1 2 3) (0 2 3) (1 2)
(0 10 2 9 6 5 13 15 8 3 11 1 14 12 4 7)	(3 2) (3 0) (1 2 0) (0 2 3) (1 2)
(0 1 9 10 5 14 6 15 8 11 3 2 13 4 12 7)	(2 3) (2 0) (1 3 0) (0 3 2) (1 3)
(0 6 14 9 4 3 11 13 8 15 7 1 12 10 2 5)	(2 0) (3 2) (2 1) (1 2 3) (0 2 3)
(0 13 5 10 4 11 3 14 8 7 15 2 12 1 9 6)	(3 0) (2 3) (3 1) (1 3 2) (0 3 2)
(0 5 13 10 12 11 2 7 8 15 6 3 4 1 9 14)	(2 1) (1 0) (3 1) (0 2 3) (0 3 2)
(0 14 6 9 12 1 11 7 8 5 15 3 4 10 2 13)	(3 1) (1 0) (2 1) (0 3 2) (0 2 3)
(0 5 13 2 4 11 3 6 8 15 7 10 12 1 9 14)	(1 3 0) (2 3) (3 1) (1 2 0) (0 3 2)
(0 14 6 1 4 3 11 5 8 7 15 9 12 10 2 13)	(1 2 0) (3 2) (2 1) (1 3 0) (0 2 3)
(0 14 6 1 4 3 11 13 8 7 15 9 12 10 2 5)	(1 2 0) (3 2) (2 1) (1 2 3 0) (0 2 3)
(0 5 13 2 4 11 3 14 8 15 7 10 12 1 9 6)	(1 3 0) (2 3) (3 1) (1 2 3 0) (0 3 2)
(0 5 13 2 4 3 11 6 8 15 7 10 12 9 1 14)	(2 3) (2 3 0) (1 3 2) (3 1) (0 3 2)
(0 14 6 1 4 11 3 5 8 7 15 9 12 2 10 13)	(3 2) (2 3 0) (1 2 3) (2 1) (0 2 3)
(0 10 2 1 6 5 13 7 8 3 11 9 14 12 4 15)	(3 2) (2 3 0) (1 2 0) (0 2 3) (1 2)
(0 1 9 2 5 14 6 7 8 11 3 10 13 4 12 15)	(2 3) (2 3 0) (1 3 0) (0 3 2) (1 3)
(0 1 9 2 5 6 14 7 8 11 3 10 13 12 4 15)	(2 3) (2 3 0) (1 3 2) (0 3 2) (1 3)
(0 10 2 1 6 13 5 7 8 3 11 9 14 4 12 15)	(3 2) (2 3 0) (1 2 3) (0 2 3) (1 2)

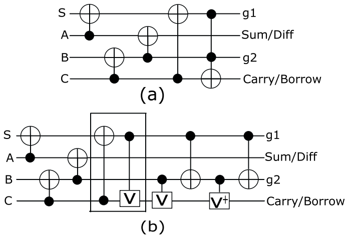
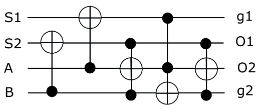


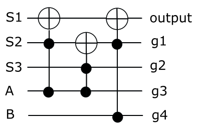
Fig. 12 Proposed full adder/subtractor (a) minimum MCT circuit, and (b) reduced quantum circuit.



S_1S_2	Operation
0 0	OR
0 1	ADD
1 0	AND
1 1	ID

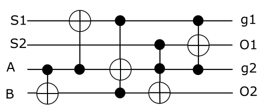
Fig. 13 Mini-ALU: Minimum MCT circuit and its operation assignment.

in which subtraction is adopted instead of ID in Mini-ALU. The truth table has been given in Tables 2 and 3 as the examples of the operation assignment and the output permutation. We confirmed that $\gamma(\mathsf{ALU}) = 7$, $\gamma(OAEC(\mathsf{ALU})) = 6$, and $\gamma(PEC(\mathsf{ALU})) = 6$ as in Examples 2 and 3. By combining $OAEC$ and PEC, a further reduction has been made: $\gamma(PEC(OAEC((\mathsf{ALU})))) = 5$. The list of $OPT(PEC(OAEC(\mathsf{ALU})))$ is omitted, but the number of it is $|OPT(PEC(OAEC(\mathsf{ALU})))| = 19$. One of the minimum MCT circuits among them is shown in Fig. 15.



S_1S_2	S_3 0	S_3 1
0 0	Constant 0	Constant 1
0 1	AND	NAND
1 0	XOR	XNOR
1 1	OR	NOR

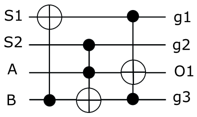
Fig. 14 Gupta's-LU: Minimum MCT circuit and its operation assignment.



S_1S_2	Operation
0 0	ADD
0 1	OR
1 0	SUB
1 1	AND

Fig. 15 ALU: Minimum MCT circuit and its operation assignment.

In contrast, we consider a simpler version of Gupta's-LU. Gupta's-LU implements eight operations with five inputs. By reducing the operations into AND, OR, and XOR, a compact version with four inputs can be defined, which is denoted by LU. We select AND, OR, and XOR because they are the most fundamental and common logic operations. The NOT operation is included in XOR. NOT can be executed by assigning the value 1 to one of the XOR operands: $\bar{x} = x \oplus 1$. The compact LU is useful when this set of operations are sufficient for the use of logic units. FIND-OPT confirmed that $\gamma(PEC(OAEC(\mathsf{LU}))) = 3$ and $|OPT(PEC(OAEC(\mathsf{LU})))| = 28$. One of the minimum MCT cir-



S_1S_2	Operation
0 0	XOR
0 1	OR
1 0	N/A
1 1	AND

Fig. 16 LU: Minimum MCT circuit and its operation assignment.

cuits is shown in Fig. 16.

5. Reduced Quantum Circuits

A number of reversible circuits of adder/subtractor blocks [5], [12], [16] have been proposed with using various reversible gate libraries under the different constraints of constant and garbage lines. The common way to compare the total quality of them is to evaluate the quantum cost (QC) of the circuits. In this section, we obtain quantum circuits of our reversible adder/subtractors and ALUs to make a comparison of the QC with the existing counterparts.

5.1 Reduction of the QCs

This section describes the transformation of MCT circuits into quantum circuits with the NCV gate library and the reduction of the quantum circuits with a simple greedy algorithm. The algorithm is given in Fig. 17. In this algorithm, the function ObtainQuantumCircuit takes an MCT circuit as the argument and produces the reduced quantum circuit as the return value. Unlike the MCT circuits in Sect. 4, the resulting quantum circuits are sub-optimal. Although the optimality is not guaranteed, the QC of our quantum circuits of the reversible adder/subtractors and ALUs are lower than that of the existing counterparts.

The function QuantumCircuits transforms an MCT circuit *mctC* into a set of initial quantum circuits. For Toffoli gates in *mctC*, QuantumCircuits generates all possible combinations of Toffoli decomposition described in Sect. 2.6 during the transformation.

The function ReduceQuantum reduces the QC of a quantum circuit *qC* by a greedy approach. In ReduceQuantum, the function MoveGateAndReduce searches a pair of adjacent gates in C by moving the gate g according to the moving rule of Property 1. If the pair of gates forms the identity function (according to deletion rules shown in Fig. 5), then the pair is deleted. If a pair of gates can be merged into a single gate (according to the merging rules shown in Fig. 4), then the pair is replaced by the single gate. The execution of MoveGateAndReduce is repeated if the quantum gates in the circuit C are reduced successively.

5.2 Comparison of Quantum Costs

We have implemented the algorithm in Fig. 17 in MATLAB. Tables 10 and 11 show the QC of half and full adder/subtractors for all minimum MCT circuits shown in Tables 7 and 9, respectively, in which the ‘After Reduction’ columns indicate the QC after executing our program.

```
function ObtainQuantumCircuit($mctC$)
                                        ▷ Input: $mctC$ is an MCT circuit.
                                        ▷ Output: a reduced quantum circuit.
    return a circuit with the smallest QC in {ReduceQuantum($qC$) | $qC \in$ QuantumCircuits($mctC$)};
end function
function ReduceQuantum($qC$)
                                        ▷ Input: $qC$ is a quantum circuit.
                                        ▷ Output: a reduced quantum circuit.
    var $C$ : Circuit;
    var $C^* \leftarrow qC$ : Circuit;
    repeat
        $C \leftarrow C^*$;
        for all gate $g$ in $C$ do
            $C^* \leftarrow$ MoveGateAndReduce($C, g$);
            if $|C^*| < |C|$ then
                break;                  ▷ Exiting the for loop.
            end if
        end for
    until $|C^*| = |C|$
    return $C^*$;
end function
```

$|C|$: the number of gates in the quantum circuit C.

Fig. 17 Simple algorithm for obtaining reduced quantum circuits.

Table 10 Half adder/subtractor.

Circuit	Quantum Cost	
No	Before Reduction	After Reduction
1	7	7
2	11	8
3	11	8
4	7	7
5	11	10
6	11	10

The smallest quantum circuits in Tables 10 and 11 are shown in Fig. 11(b) and Fig. 12(b), respectively. Although, in Fig. 12(b), the number of elementary quantum gates is nine, we count the QC as eight according to the 2-qubit gate rule in Fig. 6 in Sect. 2.5. Table 12 shows the comparison with existing full adder/subtractor designs with different reversible gate libraries. According to our knowledge, the proposed design of adder/subtractor is better than the existing counterparts in terms of the QC, the number of constant input, and the garbage outputs.

Table 13 shows the comparison of 1-bit ALUs, in which ‘Gates’ and ‘QC’ denote the number of MCT gates and the QC, respectively. ‘QC*’ is the QC of the 2-qubit gate rule. The QC of Gupta’s-LU and Mini-ALU has been reduced drastically in our design compared with the original.

6. Conclusion

For most of the computing devices, adder/subtractor blocks and ALUs are crucial, and the development of cost-efficient arithmetic blocks will improve the efficiency of the whole system. In this paper, we gave the minimum MCT circuits that realize half and full adder/subtractors and some benchmark ALUs. To improve the function design of this type of

Table 11 Full adder/subtractor.

Circuit No	Quantum Cost	
	Before Reduction	After Reduction
1	9	8
2	9	8
3	13	11
4	13	11
5	9	8
6	9	8
7	13	10
8	13	10
9	13	9
10	13	11
11	13	9
12	13	9
13	13	11
14	13	11
15	13	11
16	13	11
17	13	12
18	13	12
19	13	12
20	13	12
21	17	16
22	17	16
23	26	23
24	26	23
25	17	13
26	17	11
27	17	12
28	17	14
29	17	15
30	17	13

Table 12 Comparison between proposed reversible full adder/subtractor and the existing counterparts.

Design	Cost Metrics				
	Input Lines	Reversible Gates	Constant Inputs	Garbage Outputs	QC
1 [12]	7	8	3	5	21
2 [12]	5	4	1	3	14
3 [12]	5	4	1	3	10
[16]	4	4	0	2	11
[5]	5	3	1	2	9
Proposed	4	5	0	2	8

Table 13 Comparison of 1-bit ALUs.

Design	Operations	Lines	Original Gates/QC	Our Gates /QC(QC*)
Gupta's-LU [37]	AND, NAND, OR, NOR, XOR, XNOR, Constant 1, 0	5	18/114	3/15(14)
Mini-ALU [38]	AND, OR, ADD, ID	4	6/62	5/15(14)
Proposed ALU	ADD, OR, AND, SUB	4		5/17(15)
Proposed LU	XOR, OR, AND, N/A	4		3/11(10)

QC*: QC under the merged 2-qubit gate rule.

arithmetic blocks, we proposed a combination of the equivalence classes $PEC(OAEC(F))$ of ISRFs F and extended the minimization problem of F into $\gamma(\mathcal{F})$ and $OPT(\mathcal{F})$ for the set of ISRFs $\mathcal{F}$. The quantum circuits of those MCT circuits were also given as the goal of the reversible synthesis. By the experimental results, we have shown that our proposed quantum circuits are better than the existing design in terms of the number of input lines, constant inputs, garbage outputs, and the quantum cost.

Recently, linear nearest neighbor (LNN) architecture have gotten intensive attention to a practical point of view and with respect to various nanotechnologies, such as quantum optics, linear ion Trap, nuclear magnetic resonance (NMR), where every qubit can interact with at most one neighbor above and one neighbor below. It is, therefore, important to develop a design methodology for the new architecture. According to the different technological constraints, we are considering to revise our synthesis approach and design more practical reversible and quantum adder/subtractors and ALUs.

References

[1] M.K. Thomsen, R. Gluck, and H.B. Axelsen, "Reversible arithmetic logic unit for quantum arithmetic," J. Phys. A: Math. Theor., vol.43, no.38, 2010.

[2] M. Morrison and N. Ranganathan, "Design of reversible ALU based on novel programmable reversible logic gate structures," IEEE Computer Society Annual Symposium on VLSI, pp.126–131, 2011.

[3] R. Aradhaya, K.N. Muralidhara, and B. Kumar, "Design of low power arithmetic unit based on reversible logic," International Journal of VLSI and Signal Processing Applications, vol.1, no.1, pp.30–38, 2011.

[4] B.K. Sikdar, "Design of fault tolerant reversible arithmetic logic unit in QCA," International Symposium on Electronic System Design, 2012.

[5] S. Sultan and K. Radecka, "Reversible architecture of computer arithmetic," Int. J. Comput. Appl., vol.93, no.14, pp.6–14, May 2014.

[6] M.H.A. Khan, "Quantum realization of ternary adder circuits," Proc. Third International Conference on Electrical and Computer Engineering, pp.249–252, 2004.

[7] Y.V. Rentergem and A.D. Vos, "Optimal design of a reversible full adder," International Journal of Unconventional Computing, vol.1, pp.339–355, 2005.

[8] H. Thapliyal and A.P. Vinod, "Transistor realization of reversible TSG gate and reversible adder architectures," Proc. IEEE Asia Pacific Conference on Circuits and Systems, pp.418–421, 2006.

[9] H. Thapliyal and M.B. Srinivas, "Novel design and reversible logic synthesis of multiplexer based full adder and multipliers, Forty Eight Midwest Symposium on Circuits and Systems, vol.2, pp.1593–1596, 2006.

[10] R.K. James, T.K. Shahana, K.P. Jacob, and S. Sasi, "A new look at reversible logic implementation of decimal adder," International Symposium on System-On-Chip, pp.1–4, 2007.

[11] M. Majid, M. Eshghi, M. Haghparast, and A. Bahrololoom, "Design and optimization of reversible BCD adder/subtractor circuit for quantum and nanotechnology based systems," World Applied Sciences Journal, vol.4, no.6, pp.787–792, 2008.

[12] H.G. Rangaraju, U. Venugopal, K.N. Muralidhara, and K.B. Raja, "Low power reversible parallel binary adder/subtractor," International Journal of VLSI Design and Communication System (VLSICS), vol.1, no.3, pp.23–34, 2010.

[13] L. Ni, Z. Guan, and W. Zhu, "A general method of constructing the reversible full-adder," Third International Symposium on Intelligent Information Technology and Security Informatics, pp.109–113, 2010.

[14] V. Kamalakannan, V. Shilpakala, and H.N. Ravi, "Design of adder/subtractor circuit based on reversible gates," International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering, vol.2, no.8, pp.738–742, Aug. 2013.

[15] J. Kaur and H. Kaur, "Synthesis and designing of reversible adder/subtracter circuits," International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering, vol.3, no.5, pp.9325–9332, 2014.

[16] S. Moghimi and M.R. Reshadinezhad, "A novel 4×4 universal reversible gate as a cost efficient full adder/subtractor in terms of reversible and quantum metrics," International Journal of Modern Education and Computer Science, vol.7, no.11, pp.28–34, DOI: 10.5815/ijmecs.2015.11.04, 2015.

[17] C. Bennett, "Logical reversibility of computation," IBM J. Res. Dev., vol.17, no.6, pp.525–532, 1973.

[18] R. Landauer, "Irreversibility and heat generation in the computing process," IBM J. Res. Dev., vol.5, no.3, pp.183–191, July 1961.

[19] D. Maslov, Reversible logic synthesis, Ph.D. thesis, The Faculty of Computer Science, The University of New Brunswick, Canada, 2003.

[20] J. Donald and N.K. Jha, "Reversible logic synthesis with Fredkin and Peres gates," ACM J. Emerg. Technol. Comput. Syst., vol.4, no.1, pp.1–19, 2008.

[21] D. Maslov and M. Saeedi, "Reversible circuit optimization via leaving the boolean domain," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol.30, no.6, pp.806–816, 2011.

[22] M. Saeedi and I.L. Markov, "Synthesis and optimization of reversible circuits a survey," ACM Comput. Surv., vol.45, no.2, pp.21–34, 2013.

[23] D. Maslov and G.W. Dueck, "Garbage in reversible designs of multiple output functions," Proc. 6th International Symposium on Representations and Methodology of Future Computing Technologies (RM 2003), pp.162–170, Germany, March 2003.

[24] D. Maslov and G.W. Dueck, "Reversible cascades with minimal garbage," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol.23, no.11, pp.1497–1509, Nov. 2004.

[25] D. Miller, R. Wille, and R. Drechsler, "Reducing reversible circuit cost by adding lines," IEEE International Symposium on Multiple-Valued Logic, pp.217–222, 2010.

[26] D.M. Miller, "Lower cost quantum gate realizations of multiple-control Toffoli gates," IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp.308–313, 2009.

[27] D.M. Miller and Z. Sasanian, "Lowering the quantum gate cost of reversible circuits," 53rd IEEE International Midwest Symposium on Circuits and Systems, pp.260–263, 2010.

[28] M. Szyprowski and P. Kerntopf, "Reducing quantum cost in reversible Toffoli circuits," Proc. 10th Reed-Muller Workshop, pp.127–136, 2011.

[29] P. Kerntopf and M. Szyprowski, "Reducing quantum cost of pairs of multi-control Toffoli gates," International workshop on Boolean Problems, pp.263–268, 2012.

[30] D. Grosse, X. Chen, G.W. Dueck, and R. Drechsler, "Exact SAT-based Toffoli network synthesis," Proc. 17th Great Lakes Symposium on VLSI, pp.96–101, Italy, 2007.

[31] G.W. Dueck, D. Maslov, and D.M. Miller, "Techniques for the synthesis of reversible Toffoli networks," ACM Trans. Des. Autom. Electron. Syst., vol.12, no.4, pp.1–28, 2007.

[32] O. Golubitsky, S.M. Falconer, and D. Maslov, "Synthesis of the optimal 4-bit reversible circuits," Proc. 47th Design Automation Conference, pp.653–656, USA, June 2010.

[33] Z. Sasanian and D.M. Miller, "Mapping a multiple-control Toffoli gate cascade to an elementary quantum gate circuit," Proc. 2nd Workshop on Reversible Computation, pp.83–90, 2010.

[34] Z. Sasanian and D.M. Miller, "Transforming MCT circuits to NCVW circuits," Reversible Computation, RC 2011, A. De. Vos and R. Wille, eds., LNCS 7165, pp.77–88, Springer-Verlag, 2011.

[35] R. Wille, D.M. Miller, and Z. Sasanian, "Elementary quantum gate realizations for multiple-control Toffoli gates," Proc. 41st IEEE International Symposium on Multiple-Valued Logic, pp.288–293, 2011.

[36] R.P. Feynman, "Quantum mechanical computers," Optic News, vol.11, no.2, p.11, Feb. 1985.

[37] P. Gupta, A. Agrawal, and N.K. Jha, "An algorithm for synthesis of reversible logic circuits," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol.25, no.11, pp.2317–2330, 2006.

[38] RevLib: online resource for reversible benchmarks, http://www.revlib.org

[39] M.M. Rahman, Synthesis of Reversible Logic, Ph.D. dissertation, In the Graduate Academic Unit of Faculty of Computer Science, The University of New Brunswick, Dec. 2014.

[40] M.A. Nielsen and I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.

[41] W. Hung, X. Song, G. Yang, and M. Perkowski, "Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol.25, no.9, pp.1652–1663, 2006.

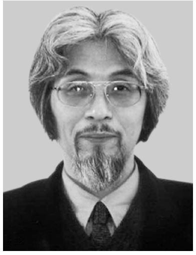
[42] R. Wille and R. Drechsler, Towards a Design Flow for Reversible Logic, Springer, 2010.

[43] O. Golubitsky and D. Maslov, "A study of optimal 4-bit reversible Toffoli circuits and their synthesis," IEEE Trans. Comput., vol.61, no.9, pp.1341–1353, Sept. 2012.

Md Belayet Ali received his B.Sc. degree from Mawlana Bhashani Science and Technology University, Tangail, Bangladesh, in 2008. He received his M.E. degree from Iwate University, Morioka, Japan, in 2016. He is currently working toward his Ph.D. at Iwate University. His research interests include reversible/quantum logic synthesis and optimization algorithms.

Takashi Hirayama received his B.E., M.E., and Ph.D. degrees in computer science from Gunma University in 1994, 1996, and 1999, respectively. From 1999 to 2001 he was a research assistant in the Department of Electrical and Electronics Engineering, Ashikaga Institute of Technology. He is currently a lecturer in the Department of Electrical Engineering and Computer Science, Faculty of Engineering, Iwate University. His research interests include high level and logic synthesis and design for testability of VLSIs.

Katsuhisa Yamanaka received his B.E., M.E., and Ph.D. degrees in computer science from Gunma University in 2003, 2005 and 2007, respectively. He is an assistant professor of the Department of Electrical Engineering and Computer Science, Faculty of Engineering, Iwate University. His research interests include combinatorial algorithms and graph algorithms.



Yasuaki Nishitani received his B.E. degree in electrical engineering, M.E. and Ph.D. degrees in computer science from Tohoku University in 1975, 1977, and 1984, respectively. In 1981 he joined the Software Product Engineering Laboratory at the NEC Corporation. From 1987 to 2000 he was an associate professor in the Department of Computer Science, Gunma University. From 2000 to 2017, he was a professor in the Department of Electrical Engineering and Computer Science, Faculty of Engineering, Iwate University. His research interests include switching theory, software engineering, and distributed algorithms.