PAPER *Special Section on Multiple-Valued Logic and VLSI Computing*

A Lower Bound on the Gate Count of Toffoli-Based Reversible Logic Circuits

Takashi HIRAYAMA[†a)], ***Member***, **Hayato SUGAWARA**[†], ***Nonmember***, **Katsuhisa YAMANAKA**[†], ***and*** **Yasuaki NISHITANI**[†], ***Members***

SUMMARY We present a new lower bound on the number of gates in reversible logic circuits that represent a given reversible logic function, in which the circuits are assumed to consist of general Toffoli gates and have no redundant input/output lines. We make a theoretical comparison of lower bounds, and prove that the proposed bound is better than the previous one. Moreover, experimental results for lower bounds on randomly-generated reversible logic functions and reversible benchmarks are given. The results also demonstrate that the proposed lower bound is better than the former one.

key words: *reversible logic circuits, Toffoli gates, lower bound, logic minimization*

1. Introduction

The synthesis of reversible logic circuits is a fundamental part of the quantum logic field. NOT, CNOT, and Toffoli gates are typically used for synthesizing reversible logic circuits [1]–[4] as well as quantum logic ones [5]. Fredkin and SWAP gates are also known [6]–[8] as other types of reversible logic gates. Figure 1 shows an example of Toffoli gates. The standard Toffoli (Fig. 1 (c)) is a 3-bit gate and it can be generalized to k-bit Toffoli like Fig. 1 (d).

In this paper, NOT, CNOT, Toffoli, and k-bit Toffoli are referred to as the general Toffoli library since NOT and CNOT can be considered as 1-bit and 2-bit Toffolis, respectively. We discuss the reversible logic synthesis with the general Toffoli library.

Some types of reversible circuits may have redundant input/output lines [2]. Since the presence of those redundant lines is considered as the cost of the circuits [9], it is desirable not to use the redundant lines. For that reason, the most fundamental model is the reversible circuits without redundant lines [1], [3], [4], [6]–[8], [10]. We deal with the reversible logic circuits without redundant input/output lines.

In addition to redundant lines, there are many technology-specific cost metrics [11]–[13]: the number of elementary gates, quantum cost, delay, depth etc. However, their relevance will change depending on future developments in quantum-circuit technologies. Therefore, this paper adopts the number of gates as the fundamental and technology-independent cost metric. Although different gates require different resources in a precise sense, the metric approximately reflects the complexity of circuits. Szyprowski et al. [22] compared many complexity measures for reversible functions and have reported that the Reed-Muller spectra size gives the best estimation for the complexity of minimum circuits. The comparison was done experimentally, and the estimation was not a kind of theoretical upper or lower bounds on the circuit complexity.

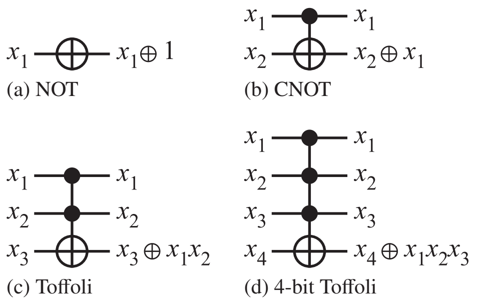

Fig. 1 Example of reversible gates.

For the Toffoli-based logic synthesis, a number of optimization algorithms have been proposed; some of which are the exact minimization and others are the heuristic simplification. Golubitsky et al. [14], [23] and Szyprowski et al. [15] studied the exact minimization. Their algorithms guarantee the minimality of the resulting circuits. However, the extent of applicability of the exact minimizers is limited to smaller functions only. For larger functions, heuristic simplification [1]–[3], [6], [8], [16] is useful, which does not guarantee the minimality but obtains near optimum circuits in a practical computation time. Theoretically the circuits obtained by heuristic simplifiers can be seen as the upper bounds for the exact minimum ones. This means that many methods for obtaining the upper bounds for given functions have been already studied. As analytical bounds, Saeedi et al. [17] discussed the upper bounds for the classes of functions with various length of cycles.

As well as upper bounds, lower bounds have theoretical and practical significance as a measure of the complexity of reversible circuits. However, the works for the lower bounds are far fewer. Maslov et al. [9] and Shende et al. [18] presented the upper and lower bounds on the longest minimum circuits for the class of all n-input reversible functions. With regard to the lower bounds for given functions, Higashiohno

Manuscript received November 29, 2013.
Manuscript revised April 11, 2014.

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

DOI: 10.1587/transinf.2013LOP0013

et al. [19] presented a simple lower bound by evaluating the Positive Polarity Reed-Muller expressions (PPRMs) of the functions. We present a new and better lower bound on the minimum number of Toffoli gates in the reversible circuits that represent a given function. The number of Toffoli gates used in a reversible circuit is also called the gate count in this paper.

The paper is organized as follows. The next section outlines some preliminaries and reviews the previous lower bound. Section 3 presents a new lower bound. In Sect. 4, it is proved mathematically that the proposed bound is better than the previous one. Section 5 describes the fast calculation of the bounds. The experimental results are given in Sect. 6. We conclude in Sect. 7.

2. Preliminaries

In this paper, n-input single-output logic functions ("logic functions" for short) are represented by positive polarity Reed-Muller expressions (PPRMs) [20]. It is known that any logic function can be represented by the PPRM uniquely. For example, the logic function $x_1\bar{x}_2+x_2$ is written as $x_1 \oplus x_2 \oplus x_1x_2$ in PPRM.

The composition of an n-input single-output logic function f and an n-input n-output logic function F is denoted by $f \circ F$ and is defined as the mathematical function composition, i.e., the composite function $f' = f \circ F$ is the n-input single-output logic function such that $f'(X) = f(F(X))$ for all the input vectors $X \in \{0,1\}^n$. Similarly, the composite function $F_1 \circ F_2$ of n-input n-output logic functions F_1 and F_2 is the n-input n-output logic function such that $F'(X) = F_1(F_2(X))$ for all $X \in \{0,1\}^n$. The composition of functions is always associative.

Now we give a notation of an n-input n-output logic function. Let x_i $(1 \le i \le n)$ be the variable that represents the i-th element of the input vector. By regarding x_i as a logic function, the composition $x_i \circ F$ can be seen as the logic function that represents the i-th output of F. By letting $f_i = x_i \circ F$, F is denoted by f_i and x_i in pairs as follows.

$$[f_1/x_1, f_2/x_2, \ldots, f_n/x_n] \quad \text{where } f_i = x_i \circ F$$

Each f_i/x_i describes the i-th output and input. The trivial case where $f_i = x_i \circ F = x_i$, or x_i/x_i is sometimes omitted in this notation, for simplicity.

Example 1: $\{x_1, x_2, x_3\}$ is a set of variables. $F = [x_1 \oplus x_2x_3/x_1, x_2 \oplus 1/x_2, x_3/x_3]$ is a 3-input 3-output logic function, where f_i is written in PPRM. In this case, F can be also represented as $[x_1 \oplus x_2x_3/x_1, x_2 \oplus 1/x_2]$ by omitting the notation of x_3/x_3. Generally, the composition of n-input single-output logic function f and n-input n-output logic function F can be considered as a substitution in PPRMs, in which each variable x of f is replaced with $x \circ F$. Let $f = x_3 \oplus x_1x_2$ and $F = [x_1 \oplus x_2x_3/x_1, x_2 \oplus 1/x_2]$. The composition $f \circ F$ results in $(x_3 \oplus x_1x_2) \circ F = (x_3 \circ F) \oplus (x_1 \circ F)(x_2 \circ F) = x_3 \oplus (x_1 \oplus x_2x_3)(x_2 \oplus 1) = x_3 \oplus x_1x_2 \oplus x_1$. Suppose that $F' = [x_1 \oplus x_1x_2 \oplus x_3/x_1, x_3 \oplus x_1x_2/x_3]$. The composition $F' \circ F$ results in $[x_2x_3 \oplus x_3 \oplus x_1x_2/x_1, x_2 \oplus 1/x_2, x_3 \oplus x_1x_2 \oplus x_1/x_3]$, whose output functions $x \circ (F' \circ F)$ are obtained separately as follows.

$$\begin{aligned} x_1 \circ (F' \circ F) &= (x_1 \circ F') \circ F = (x_1 \oplus x_1x_2 \oplus x_3) \circ F \\ &= x_2x_3 \oplus x_3 \oplus x_1x_2 \\ x_2 \circ (F' \circ F) &= (x_2 \circ F') \circ F = x_2 \circ F \\ &= x_2 \oplus 1 \\ x_3 \circ (F' \circ F) &= (x_3 \circ F') \circ F = (x_3 \oplus x_1x_2) \circ F \\ &= x_3 \oplus x_1x_2 \oplus x_1 \end{aligned}$$

□

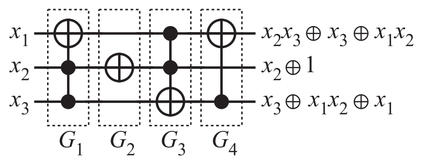
x_1
x_2
x_3
$x_2x_3 \oplus x_3 \oplus x_1x_2$
$x_2 \oplus 1$
$x_3 \oplus x_1x_2 \oplus x_1$
G_1
G_2
G_3
G_4

Fig. 2 Example of reversible logic circuit.

Definition 1: An n-input n-output logic function F is called reversible if F is bijective. □

Definition 2: A k-bit Toffoli gate has $(k-1)$ control lines $x_1, x_2, \ldots, x_{k-1}$ and one target line x_k, and realizes the reversible logic function $[x_k \oplus x_1x_2 \cdots x_{k-1}/x_k]$; the target line maps x_k to $x_k \oplus x_1x_2 \cdots x_{k-1}$ and the other lines pass the signals unaltered. The function of the 1-bit Toffoli, referred to as NOT, is $[x_1 \oplus 1/x_1]$. □

Example 2: The Toffoli gates (a), (b), (c), and (d) in Fig. 1 realize $[x_1 \oplus 1/x_1]$, $[x_2 \oplus x_1/x_2]$, $[x_3 \oplus x_1x_2/x_3]$, and $[x_4 \oplus x_1x_2x_3/x_4]$, respectively. □

To discuss a lower bound for reversible functions, we regard a Toffoli gate as a function rather than a device, hereafter.

Definition 3: A reversible logic circuit is denoted by the sequence of Toffoli gates $G_1G_2 \ldots G_m$, where m is called the gate count (GC) of the circuit. The reversible function realized by the circuit is the composition of gates $G_m \circ G_{m-1} \circ \cdots \circ G_1$. □

Example 3: Figure 2 shows a reversible logic circuit with four gates, which is represented by $G_1G_2G_3G_4 = [x_1 \oplus x_2x_3/x_1][x_2 \oplus 1/x_2][x_3 \oplus x_1x_2/x_3][x_1 \oplus x_3/x_1]$. The reversible function F realized by the circuit is the composition of these gates, i.e., $G_4 \circ G_3 \circ G_2 \circ G_1 = [x_1 \oplus x_3/x_1] \circ [x_3 \oplus x_1x_2/x_3] \circ [x_2 \oplus 1/x_2] \circ [x_1 \oplus x_2x_3/x_1] = [x_1 \oplus x_3/x_1] \circ [x_3 \oplus x_1x_2/x_3] \circ [x_1 \oplus x_2x_3/x_1, x_2 \oplus 1/x_2] = [x_1 \oplus x_3/x_1] \circ [x_1 \oplus x_2x_3/x_1, x_2 \oplus 1/x_2, x_3 \oplus x_1x_2 \oplus x_1/x_3] = [x_2x_3 \oplus x_3 \oplus x_1x_2/x_1, x_2 \oplus 1/x_2, x_3 \oplus x_1x_2 \oplus x_1/x_3]$. This F can be represented by another composition of three gates as $[x_1 \oplus x_2x_3/x_1] \circ [x_3 \oplus x_1x_2/x_3] \circ [x_2 \oplus 1/x_2]$. This means that F can also be realized by the three-gate reversible circuit $[x_2 \oplus 1/x_2][x_3 \oplus x_1x_2/x_3][x_1 \oplus x_2x_3/x_1]$, which is shown in Fig. 3. □

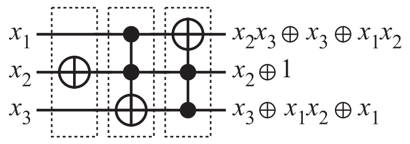
x_1
x_2
x_3
$x_2x_3 \oplus x_3 \oplus x_1x_2$
$x_2 \oplus 1$
$x_3 \oplus x_1x_2 \oplus x_1$

Fig. 3 Example of reversible logic circuit with three gates.

For a given reversible function F, the synthesis of reversible logic circuits can be seen as the problem of obtaining a sequence of Toffoli gates $G_1 \ldots G_{m-1}G_m$ that satisfies $F = G_m \circ G_{m-1} \circ \cdots \circ G_1$. As we have seen in Example 3, there are a number of possible reversible logic circuits that realize a given F. The GC of the resulting circuit varies with the synthesizers. As a theoretical research, we discuss a lower bound on the minimum GC of the circuits of a given function F.

In order to discuss the lower bounds, we define a complexity measure $\sigma(F)$ of a reversible function F, and briefly refer to the previous lower bound [19].

Definition 4: The number of product terms of the PPRM of a logic function f is denoted by $\tau(f)$. We define $\sigma_i(f) = \tau(x_i \oplus f)$, where x_i is a variable. Let F be a reversible function with n variables. $\sigma(F)$ is defined as follows.

$$\sigma(F) = \sum_{1 \le i \le n} \sigma_i(x_i \circ F)$$

□

Example 4: For the reversible function $F = [x_2x_3 \oplus x_3 \oplus x_1x_2/x_1, x_2 \oplus 1/x_2, x_3 \oplus x_1x_2 \oplus x_1/x_3]$, which was given in Example 3 (Fig. 2), $\sigma_1(x_1 \circ F) = \tau(x_1 \oplus x_2x_3 \oplus x_3 \oplus x_1x_2) = 4$, $\sigma_2(x_2 \circ F) = \tau(x_2 \oplus x_2 \oplus 1) = 1$, and $\sigma_3(x_3 \circ F) = \tau(x_3 \oplus x_3 \oplus x_1x_2 \oplus x_1) = 2$. Thus, $\sigma(F)$ is 7. If F is an identity function, e.g., $F = [x_1/x_1, x_2/x_2, x_3/x_3]$, then $\sigma(F) = 0$. □

Definition 5: Among all reversible circuits that realize a reversible function F, those with the exact minimum GC are called the minimum circuits of F. The GC of a minimum circuit of F is called the GC of F and denoted by $\gamma(F)$. □

Theorem 1 (Lower Bound Theorem [19]): For any reversible function F, the inequality $2^{\gamma(F)} - 1 \ge \sigma(F)$, i.e.,

$$\gamma(F) \ge \lceil \log_2(\sigma(F) + 1) \rceil,$$

holds. □

Example 5: For the reversible function $F = [x_2 \oplus x_1x_3 \oplus x_2x_3/x_1, 1 \oplus x_1 \oplus x_3 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3/x_2, x_3 \oplus 1 \oplus x_2 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3/x_3]$, $\sigma(F) = 16$ holds. From the lower bound theorem, $\lceil \log(\sigma(F) + 1) \rceil = \lceil \log 17 \rceil = 5$ or more gates are required in the reversible circuits to realize F. Meanwhile, F can be represented by the composition of five gates: $[x_2 \oplus x_3/x_2] \circ [x_3 \oplus x_1x_2/x_3] \circ [x_1 \oplus x_2x_3/x_1] \circ [x_2 \oplus x_1/x_2] \circ [x_3 \oplus 1/x_3]$. This means that the upper bound is also 5. Therefore, we can conclude $\gamma(F) = 5$ and the above result is exact minimum. Like this, lower bounds may be used to guarantee the minimality of the results from a logic synthesizer. □

3. A New Lower Bound on the Gate Count of the Reversible Circuits

We propose a new lower bound based on the numerical calculation while Theorem 1 is an analytic lower bound. Theorem 1 uses $\sigma(F)$, which is a scalar information obtained from the reversible function F. However, it is more natural to consider the lower bound with a collection of multiple information since F has multiple outputs. Therefore, we discuss a lower bound utilizing the vector operations.

Definition 6: The exclusive-or set of two sets Q_1 and Q_2 is denoted by $Q_1 \oplus Q_2$, i.e., $Q_1 \oplus Q_2 = (Q_1 \cup Q_2) - (Q_1 \cap Q_2)$ □

Definition 7: $PPRM(f)$ denotes the set of product terms of the PPRM of the logic function f. A product term p is said to appear in the PPRM of f if $p \in PPRM(f)$. We say that f is independent of a variable x if no product terms that have the literal x appear in the PPRM of f. □

To argue lower bounds, we briefly review some essential properties of PPRMs. Properties 2 and 3 will be used to prove Lemma 1. Since these properties are well-known basics of PPRMs, their detailed proofs are omitted.

Property 1: For any logic functions f and g, $PPRM(f \oplus g) = PPRM(f) \oplus PPRM(g)$. □

From Property 1, we have the following.

Property 2: For any logic functions f and g, $\tau(f \oplus g) \le \tau(f) + \tau(g)$ □

If logic functions f and g are independent of a variable x, $PPRM(xf) \cap PPRM(g) = \emptyset$ holds. Hence, we have the following from Property 1.

Property 3: If logic functions f and g are independent of a variable x, $\tau(xf \oplus g) = \tau(xf) + \tau(g)$. □

In this paper, we use boldface for representing arithmetic functions that produce a vector like $\boldsymbol{\Phi}$ in Definition 8.

Definition 8: Suppose that $\boldsymbol{S}$ is a vector of non-negative integers $[s_1, s_2, \ldots, s_n]$, and k is an integer with $1 \le k \le n$. We define $\boldsymbol{\Phi}(\boldsymbol{S}, k)$ as the vector $[r_1, r_2, \ldots, r_n]$ such that

$$r_i = \begin{cases} \lceil \frac{s_i - 1}{2} \rceil & \text{if } i = k \\ \lceil \frac{s_i}{2} \rceil & \text{otherwise.} \end{cases}$$

□

Note that $r_k = \lceil \frac{s_k - 1}{2} \rceil = 0$ holds if $s_k = 0$. This means that the resulting $r_1, r_2, \ldots, r_n$ are always non negative.

Example 6: Let $\boldsymbol{S} = [2, 1, 4]$. $\boldsymbol{\Phi}(\boldsymbol{S}, 1) = [\lceil \frac{2-1}{2} \rceil, \lceil \frac{1}{2} \rceil, \lceil \frac{4}{2} \rceil] = [1, 1, 2]$, $\boldsymbol{\Phi}(\boldsymbol{S}, 2) = [\lceil \frac{2}{2} \rceil, \lceil \frac{1-1}{2} \rceil, \lceil \frac{4}{2} \rceil] = [1, 0, 2]$, $\boldsymbol{\Phi}(\boldsymbol{S}, 3) = [\lceil \frac{2}{2} \rceil, \lceil \frac{1}{2} \rceil, \lceil \frac{4-1}{2} \rceil] = [1, 1, 2]$. □

$\boldsymbol{\Phi}$ is an important operator utilizing the vector information of the reversible function, and therefore is used many times throughout this paper. $\boldsymbol{\Phi}(\boldsymbol{S}, k)$ nearly halves the elements of $\boldsymbol{S}$. By applying the operation $\boldsymbol{\Phi}$ with proper indices k recursively to the resulting vector, the vector can be

$[0, 0, \ldots, 0]$ finally. Since counting the minimum depth of such recursion is a key process of our lower bound, we give the definition of the process as *σ-lb*.

Definition 9: Let $\boldsymbol{S}$ be a vector of non-negative integers and n be the length of $\boldsymbol{S}$. $\sigma\text{-}lb(\boldsymbol{S})$ is defined recursively with $\boldsymbol{\Phi}$ as follows.

$$\sigma\text{-}lb(\boldsymbol{S}) = \begin{cases} 0 & \text{if } \boldsymbol{S} = \boldsymbol{0} \\ 1 + \min\limits_{1 \le i \le n}\{\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i))\} & \text{otherwise,} \end{cases}$$

where $\boldsymbol{0}$ denotes $[0, 0, \ldots, 0]$. □

We have the following property directly from the above definition.

Property 4: Let $\boldsymbol{S}$ be a vector of non-negative integers and n be the length of $\boldsymbol{S}$. For any i $(1 \le i \le n)$, $\sigma\text{-}lb(\boldsymbol{S}) \le 1 + \sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i))$. □

Definition 10: For two vectors $\boldsymbol{S}_1 = [a_1, a_2, \ldots, a_n]$ and $\boldsymbol{S}_2 = [b_1, b_2, \ldots, b_n]$, we say $\boldsymbol{S}_1 \ge \boldsymbol{S}_2$ if $a_i \ge b_i$ for all i $(1 \le i \le n)$. □

Property 5: Let $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ be vectors of non-negative integers with the same length. If $\boldsymbol{S}_1 \ge \boldsymbol{S}_2$, then $\sigma\text{-}lb(\boldsymbol{S}_1) \ge \sigma\text{-}lb(\boldsymbol{S}_2)$. □

The proof of Property 5 is omitted since it can be easily done by the mathematical induction.

Definition 11: Let F be a reversible function with n variables. $\boldsymbol{\Lambda}(F)$ denotes the vector $[\sigma_1(x_1 \circ F), \sigma_2(x_2 \circ F), \ldots, \sigma_n(x_n \circ F)]$. □

Lemma 1: For an arbitrary reversible function F and an arbitrary Toffoli gate $G = [x_k \oplus p/x_k]$, $\boldsymbol{\Lambda}(F) \ge \boldsymbol{\Phi}(\boldsymbol{\Lambda}(F \circ G), k)$ holds. □

Proof. From the definition of $\boldsymbol{\Phi}$, the lemma is proved if we have two inequalities: $2\sigma_k(x_k \circ F) \ge \sigma_k(x_k \circ F \circ G) - 1$ for the variable x_k used in the Toffoli gate $G = [x_k \oplus p/x_k]$, and $2\sigma_i(x_i \circ F) \ge \sigma_i(x_i \circ F \circ G)$ for any variable x_i except x_k. We prove these inequalities below.

(Case for x_k) Let f denote $x_k \circ F$. By using the positive Davio expansion [20], the logic function f can be represented in the form of $f = x_k f_2 \oplus f_0$, where the subfunctions f_2 and f_0 are independent of the variable x_k. Since $\sigma_k(f \circ G) = \sigma_k((p \oplus x_k) f_2 \oplus f_0) = \sigma_k(p f_2 \oplus x_k f_2 \oplus f_0) = \tau(x_k \oplus p f_2 \oplus x_k f_2 \oplus f_0)$, we show that $2\sigma_k(f) + 1 \ge \sigma_k(f \circ G) = \tau(x_k \oplus p f_2 \oplus x_k f_2 \oplus f_0)$.

$\tau(x_k f_2) = \tau(f_2) \ge \tau(p f_2)$ since f_2 is independent of x_k, and $\tau(f_0) \le 2\tau(f_0)$ since τ is non negative. From Property 3, $\tau(x_k \oplus x_k f_2 \oplus f_0) = \tau(x_k(1 \oplus f_2) \oplus f_0) = \tau(x_k(1 \oplus f_2)) + \tau(f_0) = \tau(x_k \oplus x_k f_2) + \tau(f_0)$. From these formulae and Property 2, we have the following.

$$\begin{aligned}
&\sigma_k(f \circ G) \\
&= \tau(x_k \oplus p f_2 \oplus x_k f_2 \oplus f_0) \\
&\le \tau(x_k \oplus x_k f_2) + \tau(p f_2) + \tau(f_0) \\
&\le \tau(x_k \oplus x_k f_2) + \tau(x_k f_2) + \tau(f_0) \\
&= \tau(x_k \oplus x_k f_2) + \tau(x_k \oplus x_k \oplus x_k f_2) + \tau(f_0) \\
&\le \tau(x_k \oplus x_k f_2) + \tau(x_k) + \tau(x_k \oplus x_k f_2) + \tau(f_0) \\
&= 2\tau(x_k \oplus x_k f_2) + 1 + \tau(f_0) \\
&\le 2\tau(x_k \oplus x_k f_2) + 1 + 2\tau(f_0) \\
&= 2(\tau(x_k \oplus x_k f_2) + \tau(f_0)) + 1 \\
&= 2\tau(x_k \oplus x_k f_2 \oplus f_0) + 1 \\
&= 2\tau(x_k \oplus f) + 1 \\
&= 2\sigma_k(f) + 1
\end{aligned}$$

Thus, we have $2\sigma_k(f) + 1 \ge \sigma_k(f \circ G)$.

(Case for $x_i \ne x_k$) Let h denote $x_i \circ F$, where $x_i \ne x_k$, and let $h = x_k h_2 \oplus h_0$ be the positive Davio expansion of h with the variable x_k. The subfunctions h_2 and h_0 are independent of x_k. Since $\sigma_i(h \circ G) = \tau(x_i \oplus p h_2 \oplus x_k h_2 \oplus h_0)$, we show that $2\sigma_i(h) \ge \sigma_i(h \circ G) = \tau(x_i \oplus p h_2 \oplus x_k h_2 \oplus h_0)$.

As well as in the preceding case, $\tau(x_k h_2) \ge \tau(p h_2)$, $\tau(x_i \oplus h_0) \le 2\tau(x_i \oplus h_0)$, and $\tau(x_i \oplus h_0 \oplus x_k h_2) = \tau(x_i \oplus h_0) + \tau(x_k h_2)$ holds. From these formulae and Property 2, we have the following.

$$\begin{aligned}
&\sigma_i(h \circ G) \\
&= \tau(x_i \oplus p h_2 \oplus x_k h_2 \oplus h_0) \\
&\le \tau(x_i \oplus h_0) + \tau(p h_2) + \tau(x_k h_2) \\
&\le \tau(x_i \oplus h_0) + \tau(x_k h_2) + \tau(x_k h_2) \\
&= \tau(x_i \oplus h_0) + 2\tau(x_k h_2) \\
&\le 2\tau(x_i \oplus h_0) + 2\tau(x_k h_2) \\
&= 2(\tau(x_i \oplus h_0) + \tau(x_k h_2)) \\
&= 2\tau(x_i \oplus h_0 \oplus x_k h_2) \\
&= 2\tau(x_i \oplus h) \\
&= 2\sigma_i(h)
\end{aligned}$$

Thus, we have $2\sigma_i(h) \ge \sigma_i(h \circ G)$. □

Lemma 2: For an arbitrary reversible function F and an arbitrary Toffoli gate G, $\sigma\text{-}lb(\boldsymbol{\Lambda}(F)) \ge \sigma\text{-}lb(\boldsymbol{\Lambda}(F \circ G)) - 1$ holds. □

Proof. Let $G = [x_k \oplus p/x_k]$. From Property 4,

$$\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{\Lambda}(F \circ G), k)) \ge \sigma\text{-}lb(\boldsymbol{\Lambda}(F \circ G)) - 1. \tag{1}$$

Since $\boldsymbol{\Lambda}(F) \ge \boldsymbol{\Phi}(\boldsymbol{\Lambda}(F \circ G), k)$ holds from Lemma 1, we have the following from Property 5.

$$\sigma\text{-}lb(\boldsymbol{\Lambda}(F)) \ge \sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{\Lambda}(F \circ G), k)) \tag{2}$$

From the inequalities (1) and (2), we have $\sigma\text{-}lb(\boldsymbol{\Lambda}(F)) \ge \sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{\Lambda}(F \circ G), k)) \ge \sigma\text{-}lb(\boldsymbol{\Lambda}(F \circ G)) - 1$. □

By using Lemma 2, we have a new lower bound on $\gamma(F)$.

Theorem 2 (Lower Bound Theorem): For any reversible function F,

$$\gamma(F) \ge \sigma\text{-}lb(\boldsymbol{\Lambda}(F))$$

holds. □

Proof. The proof is by the mathematical induction on $\gamma(F)$. If $\gamma(F) = 0$, F is the identity function. Then, $\boldsymbol{\Lambda}(F) = \mathbf{0}$. From the definition of $\sigma\text{-}lb$, $\sigma\text{-}lb(\mathbf{0}) = 0$. Thus, the base case was proved.

Assume that the theorem holds for any reversible functions F_m such that $\gamma(F_m) = m$ $(m \geq 0)$. Using this assumption, we prove the theorem for $\gamma(F) = m + 1$. From the induction hypothesis,

$$\gamma(F_m) \geq \sigma\text{-}lb(\boldsymbol{\Lambda}(F_m)). \tag{3}$$

For any F with $\gamma(F) = m + 1$, there exist some reversible function F_m with $\gamma(F_m) = m$ and some Toffoli gate G, and F can be written by the composition of F_m and G as $F = F_m \circ G$. From the inequality (3) and Lemma 2,

$$\begin{aligned} \gamma(F) &= m + 1 \\ &= \gamma(F_m) + 1 \\ &\geq \sigma\text{-}lb(\boldsymbol{\Lambda}(F_m)) + 1 \\ &\geq (\sigma\text{-}lb(\boldsymbol{\Lambda}(F_m \circ G)) - 1) + 1 \\ &= \sigma\text{-}lb(\boldsymbol{\Lambda}(F_m \circ G)) \\ &= \sigma\text{-}lb(\boldsymbol{\Lambda}(F)). \end{aligned}$$

Thus, the inductive step was proved. □

Theorem 2 shows that the lower bound on $\gamma(F)$ can be obtained by calculating $\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$ by calling $\sigma\text{-}lb$ recursively according to Definition 9. Although the lower bound $\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$ requires a complex computation, the bound is better than the previous $\lceil \log(\sigma(F) + 1) \rceil$ as we will see in Sect. 4. The efficient calculation of $\sigma\text{-}lb$ will be discussed in Sect. 5.

Example 7: For the reversible function $F = [x_1 \oplus x_2 \oplus x_3/x_1, x_2 \oplus x_1x_2 \oplus x_1x_3/x_2, x_3 \oplus x_1x_2 \oplus x_2x_3/x_3]$, we have $\boldsymbol{\Lambda}(F) = [2, 2, 2]$ and $\sigma(F) = 6$. Theorem 1 produces $\lceil \log(\sigma(F) + 1) \rceil = \lceil \log 7 \rceil = 3$ while Theorem 2 does $\sigma\text{-}lb(\boldsymbol{\Lambda}(F)) = 4$, which is a better lower bound. □

As well as GC, quantum cost (QC) [13], [17] is a major cost metric of the circuit complexity. A trivial lower bound on QC can be obtained from $\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$ by considering the least QC of gates. Since every gate in the general Toffoli library has at least 1 quantum cost, the GC of a circuit can be seen as a lower bound on the QC of the circuit. Consequently, $\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$ is also a lower bound on QC of the circuit of F.

4. Theoretical Comparison of Lower Bounds

This section shows that the proposed lower bound $\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$ is better than the previous one $\lceil \log(\sigma(F) + 1) \rceil$. Specifically we prove $\sigma\text{-}lb(\boldsymbol{\Lambda}(F)) \geq \lceil \log(\sigma(F) + 1) \rceil$ for any F.

Definition 12: For an integer vector $\boldsymbol{S}$ with length n, $\boldsymbol{S}[i]$ denotes the i-th element of $\boldsymbol{S}$. The total sum of the elements $\sum_{1 \leq i \leq n} \boldsymbol{S}[i]$ is simply denoted by $\sum \boldsymbol{S}$.

Definition 13: Suppose that $\boldsymbol{S}$ is a vector of non-negative integers $[s_1, s_2, \ldots, s_n]$, and k is an index $(1 \leq k \leq n)$. We define $\boldsymbol{\Psi}(\boldsymbol{S}, k)$ as the vector $[r_1, r_2, \ldots, r_n]$ such that

$$r_i = \begin{cases} 2s_i + 1 & \text{if } i = k \\ 2s_i & \text{otherwise.} \end{cases}$$

□

$\boldsymbol{\Psi}(\boldsymbol{S}, k)$ doubles almost all elements of $\boldsymbol{S}$. Therefore, $\boldsymbol{\Psi}$ is the inverse operation of $\boldsymbol{\Phi}$; $\boldsymbol{\Phi}(\boldsymbol{\Psi}(\boldsymbol{S}, k), k) = \boldsymbol{S}$ holds for any $\boldsymbol{S}$ and k. Meanwhile $\boldsymbol{\Psi}(\boldsymbol{\Phi}(\boldsymbol{S}, k), k) \geq \boldsymbol{S}$ holds since $\boldsymbol{\Phi}$ uses the ceiling function. From this property, we have the following lemma.

Lemma 3: For an arbitrary vector of non-negative integers $\boldsymbol{S}$ and an arbitrary index k, $\sum \boldsymbol{\Psi}(\boldsymbol{\Phi}(\boldsymbol{S}, k), k) = 1 + 2 \sum \boldsymbol{\Phi}(\boldsymbol{S}, k) \geq \sum \boldsymbol{S}$ holds. □

Proof. From the definition of $\boldsymbol{\Psi}$, $\sum \boldsymbol{\Psi}(\boldsymbol{S}', k) = 1 + 2 \sum \boldsymbol{S}'$ holds for any $\boldsymbol{S}'$. By replacing $\boldsymbol{S}'$ with $\boldsymbol{\Phi}(\boldsymbol{S}, k)$, we have $\sum \boldsymbol{\Psi}(\boldsymbol{\Phi}(\boldsymbol{S}, k), k) = 1 + 2 \sum \boldsymbol{\Phi}(\boldsymbol{S}, k)$. On the other hand, $\sum \boldsymbol{\Psi}(\boldsymbol{\Phi}(\boldsymbol{S}, k), k) \geq \sum \boldsymbol{S}$ holds from $\boldsymbol{\Psi}(\boldsymbol{\Phi}(\boldsymbol{S}, k), k) \geq \boldsymbol{S}$. Thus we have the lemma. □

Lemma 4: For an arbitrary vector of non-negative integers $\boldsymbol{S}$, $2^{\sigma\text{-}lb(\boldsymbol{S})} \geq 1 + \sum \boldsymbol{S}$. □

Proof. The proof is by the mathematical induction on $\sigma\text{-}lb(\boldsymbol{S})$. If $\sigma\text{-}lb(\boldsymbol{S}) = 0$, $2^{\sigma\text{-}lb(\boldsymbol{S})} = 1$. From $\sigma\text{-}lb(\boldsymbol{S}) = 0$, $\boldsymbol{S} = \mathbf{0}$, therefore $\sum \boldsymbol{S} = 0$ holds. Then, $1 + \sum \boldsymbol{S} = 1$. Thus, the base case was proved.

Assume that the lemma holds for any vector of non-negative integers $\boldsymbol{S}_m$ such that $\sigma\text{-}lb(\boldsymbol{S}_m) \leq m$. Using this assumption, we prove the lemma for $\sigma\text{-}lb(\boldsymbol{S}) = 1 + m$. From the induction hypothesis,

$$2^{\sigma\text{-}lb(\boldsymbol{S}_m)} \geq 1 + \sum \boldsymbol{S}_m. \tag{4}$$

Since $\sigma\text{-}lb(\boldsymbol{S}) = 1 + m$, $\boldsymbol{S} \neq \mathbf{0}$ holds. Then, we have $\sigma\text{-}lb(\boldsymbol{S}) = 1 + \min_{1 \leq i \leq n}\{\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i))\}$ from the definition of $\sigma\text{-}lb$, and hence $\min_{1 \leq i \leq n}\{\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i))\} = m$. Then, there exists some index k such that $\min_{1 \leq i \leq n}\{\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i))\} = \sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, k)) = m$. From this equation, Eq. (4), and Lemma 3,

$$\begin{aligned} 2^{\sigma\text{-}lb(\boldsymbol{S})} &= 2^{1+m} \\ &= 2^{1+\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S},k))} = 2 \cdot 2^{\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S},k))} \\ &\geq 2(1 + \sum \boldsymbol{\Phi}(\boldsymbol{S}, k)) = 2 + 2 \sum \boldsymbol{\Phi}(\boldsymbol{S}, k) \\ &\geq 1 + \sum \boldsymbol{S}. \end{aligned}$$

Thus, the inductive step was proved. □

By replacing $\boldsymbol{S}$ with $\boldsymbol{\Lambda}(F)$ in Lemma 4, we have the following theorem.

Theorem 3: For any reversible function F, the inequality $2^{\sigma\text{-}lb(\boldsymbol{\Lambda}(F))} \geq \sum \boldsymbol{\Lambda}(F) + 1 = \sigma(F) + 1$, i.e.,

$$\sigma\text{-}lb(\boldsymbol{\Lambda}(F)) \geq \lceil \log_2(\sigma(F) + 1) \rceil$$

holds. □

Theorem 3 guarantees that the proposed lower bound $\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$ is not smaller than the previous $\lceil\log(\sigma(F)+1)\rceil$.

5. Fast Calculation of the Lower Bound

This section discusses the fast calculation of σ-*lb* by making changes in the algorithm incrementally.

5.1 SigmaLB1

SigmaLB1 in Fig. 4 is a naive algorithm for $\sigma\text{-}lb(\boldsymbol{S})$ that uses depth-first search. SigmaLB1 only calls the helper procedure helper$(\boldsymbol{S}, 0)$ on the line 17, and helper actually computes $\boldsymbol{\Phi}(\boldsymbol{S}, i)$ by calling itself recursively. From Definition 9, $\sigma\text{-}lb(\boldsymbol{S})$ counts the minimum depth of recursion. In helper, the argument c is the counter to record the depth of recursion, which is incremented by a recursive call as helper$(\boldsymbol{\Phi}(\boldsymbol{S}, i), c+1)$ on the line 12. As a result, helper$(\boldsymbol{S}, c)$ obtains $\sigma\text{-}lb(\boldsymbol{S}) + c$. The current minimum depth is stored in *min*, which is updated if a smaller result is obtained by helper. Since searching for a result bigger than *min* is a waste, the line 4 limits the searching depth up to *min*. As the output of SigmaLB1, *min* is returned finally.

To avoid the infinite recursive calls, the recursion is done only if $\boldsymbol{\Phi}(\boldsymbol{S}, i) \neq \boldsymbol{S}$. It is clear that a recursion for $\boldsymbol{\Phi}(\boldsymbol{S}, i) = \boldsymbol{S}$ $(\boldsymbol{S} \neq \boldsymbol{0})$ does not result in $\sigma\text{-}lb(\boldsymbol{S})$, i.e., for the index k such that $\boldsymbol{\Phi}(\boldsymbol{S}, k) = \boldsymbol{S}$, $\min_{1\le i\le n}\{\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i))\} \neq \sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, k))$ holds. The proof is by contradiction. By assuming $\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, k)) = \min_{1\le i\le n}\{\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i))\}$ for the index k such that $\boldsymbol{\Phi}(\boldsymbol{S}, k) = \boldsymbol{S}$, we have $\sigma\text{-}lb(\boldsymbol{S}) = \min_{1\le i\le n}\{\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i))\}$, which contradicts Definition 9.

5.2 SigmaLB2

SigmaLB2 in Fig. 5 is a revised version of SigmaLB1, in which the condition for recursion has been changed from "$\boldsymbol{\Phi}(\boldsymbol{S}, i) \neq \boldsymbol{S}$" to "$\boldsymbol{S}[i]$ is odd." In the search for the minimum depth, recursions for the even $\boldsymbol{S}[i]$ can be omitted because $\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i))$ for such i does not become smaller as shown in the lemma below.

Lemma 5: Let $\boldsymbol{S}$ be a vector of non-negative integers and k be an index. If $\boldsymbol{S}[k]$ is even, $\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, k)) \ge \sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i))$ holds for any index i. □

Proof. Let $\boldsymbol{S}_k$ and $\boldsymbol{S}_i$ denote $\boldsymbol{\Phi}(\boldsymbol{S}, k)$ and $\boldsymbol{\Phi}(\boldsymbol{S}, i)$, respectively. From Definition 8, $\boldsymbol{S}_k[k] = \lceil(\boldsymbol{S}[k]-1)/2\rceil$. Since $\boldsymbol{S}[k]$ is even from the assumption of the lemma, $\boldsymbol{S}_k[k] = \lceil(\boldsymbol{S}[k]-1)/2\rceil = \lceil\boldsymbol{S}[k]/2\rceil$. Meanwhile $\boldsymbol{S}_i[k] = \lceil\boldsymbol{S}[k]/2\rceil$ holds from the definition. Hence we have $\boldsymbol{S}_k[k] = \boldsymbol{S}_i[k]$ for index k. Similarly $\boldsymbol{S}_k[i] = \lceil\boldsymbol{S}[i]/2\rceil \ge \lceil(\boldsymbol{S}[i]-1)/2\rceil = \boldsymbol{S}_i[i]$ for index i. For other indices j $(j \neq k, j \neq i)$, $\boldsymbol{S}_k[j] = \lceil\boldsymbol{S}[j]/2\rceil = \boldsymbol{S}_i[j]$. From these comparisons, $\boldsymbol{S}_k \ge \boldsymbol{S}_i$ holds. Thus, we have $\sigma\text{-}lb(\boldsymbol{S}_k) \ge \sigma\text{-}lb(\boldsymbol{S}_i)$ from Property 5. □

From Lemma 5, if all elements in $\boldsymbol{S}$ are even, $\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i))$ will be the same for all indices i.

```
 1: function SigmaLB1($S$): Integer        ▹ Input: $S$ is a vector of non-negative integers.
 2:     Var $min$: Integer;
 3:     procedure helper($S, c$)   ▹ Input: $S$ is a vector and $c$ is an integer.
                                          ▹ Side Effect: Updating $min$.
 4:         if $c \ge min$ then return ;
 5:         end if
 6:         if $S = 0$ then
 7:             $min \leftarrow c$;
 8:             return ;
 9:         end if
10:         for $i \leftarrow 1$ to $n$ do
11:             if $\Phi(S, i) \neq S$ then
12:                 helper($\Phi(S, i), c + 1$);
13:             end if
14:         end for
15:     end procedure
16:     $min \leftarrow$ a large integer;
17:     helper($S, 0$);
18:     return $min$;
19: end function
```

Fig. 4 SigmaLB1: a naive algorithm for $\sigma\text{-}lb(\boldsymbol{S})$.

```
 1: function SigmaLB2($S$): Integer        ▹ Input: $S$ is a vector of non-negative integers.
 2:     Var $min$: Integer;
 3:     procedure helper($S, c$)   ▹ Input: $S$ is a vector and $c$ is an integer.
                                          ▹ Side Effect: Updating $min$.
 4:         if $c + \lceil\log(\sum S + 1)\rceil \ge min$ then return ;
 5:         end if
 6:         if $S = 0$ then
 7:             $min \leftarrow c$;
 8:             return ;
 9:         end if
10:         if (all elements of $S$ are even) then
11:             helper($\Phi(S, 1), c + 1$);
12:         else
13:             for $i \leftarrow 1$ to $n$ do
14:                 if ($S[i]$ is odd) then
15:                     helper($\Phi(S, i), c + 1$);
16:                 end if
17:             end for
18:         end if
19:     end procedure
20:     $min \leftarrow$ a large integer;
21:     helper($S, 0$);
22:     return $min$;
23: end function
```

Fig. 5 SigmaLB2: a revised algorithm for $\sigma\text{-}lb(\boldsymbol{S})$.

Corollary 1: Let $\boldsymbol{S}$ be a vector of non-negative integers. If all elements in $\boldsymbol{S}$ are even, $\sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, i)) = \sigma\text{-}lb(\boldsymbol{\Phi}(\boldsymbol{S}, j))$ holds for an arbitrary pair of indices i and j. □

In the special case where all elements in $\boldsymbol{S}$ are even, Corollary 1 guarantees that calling helper for just one index, e.g., $\boldsymbol{\Phi}(\boldsymbol{S}, 1)$, is enough like the lines 10–11 in Fig. 5. The statement "If $\boldsymbol{S}[i]$ is odd, $\boldsymbol{\Phi}(\boldsymbol{S}, i) \neq \boldsymbol{S}$" is true for any $\boldsymbol{S}$ and i, but its converse is not true in general. Therefore, SigmaLB2 requires fewer recursive calls than SigmaLB1. That is why SigmaLB2 is faster.

In addition, $\lceil \log(\sum S+1) \rceil$ can be used as a lower bound on σ-$lb(\boldsymbol{S})$ since σ-$lb(\boldsymbol{S}) \geq \lceil \log(\sum \boldsymbol{S}+1) \rceil$ from Lemma 4. By pruning recursive calls by evaluating $\lceil \log(\sum \boldsymbol{S}+1) \rceil$, we can make the algorithm faster. This idea is implemented as the line 4 in SigmaLB2.

5.3 SigmaLB3

During the execution of the algorithm, the branched recursion may process the same pairs of $\boldsymbol{S}$ and c many times. For avoiding such cases, memory cache is a useful technique. To reduce the cache memory consumption, we introduce an equivalent relation on the set $\boldsymbol{S}$ of all vectors of non-negative integers with length n.

Definition 14: Let $\boldsymbol{S}$ and $\boldsymbol{S}'$ be vectors of non-negative integers with the same length n, and π be a permutation of the set $\{1, 2, \ldots, n\}$. If there exists a permutation π such that $\boldsymbol{S}[\pi(i)] = \boldsymbol{S}'[i]$ for all i ($1 \leq i \leq n$), then $\boldsymbol{S}$ is P-equivalent to $\boldsymbol{S}'$, which is denoted by $\boldsymbol{S} \sim \boldsymbol{S}'$. The sorted vector of $\boldsymbol{S}$ in ascending order is represented by $\boldsymbol{rep}(\boldsymbol{S})$, and is called the representative vector of $\boldsymbol{S}$. □

Note that $\boldsymbol{S} \sim \boldsymbol{rep}(\boldsymbol{S})$ for any vector $\boldsymbol{S}$.

Property 6: Let $\boldsymbol{S}$ and $\boldsymbol{S}'$ be vectors of non-negative integers. If $\boldsymbol{S} \sim \boldsymbol{S}'$, then σ-$lb(\boldsymbol{S}) = \sigma$-$lb(\boldsymbol{S}')$. □

Recall that HELPER$(\boldsymbol{S}, c)$ is a procedure that computes the minimum depth of recursion, which results in σ-$lb(\boldsymbol{S})+c$. Consider another HELPER$(\boldsymbol{S}', c')$ such that $\boldsymbol{S} \sim \boldsymbol{S}'$ and $c \geq c'$. Since σ-$lb(\boldsymbol{S}) = \sigma$-$lb(\boldsymbol{S}')$ from Property 6, σ-$lb(\boldsymbol{S}) + c \geq \sigma$-$lb(\boldsymbol{S}') + c'$ holds. In such a case, HELPER$(\boldsymbol{S}, c)$ cannot be smaller than HELPER$(\boldsymbol{S}', c')$. Consequently, the computation of HELPER$(\boldsymbol{S}, c)$ can be pruned away if HELPER$(\boldsymbol{S}', c')$ like above has been computed before. Based on this idea, SigmaLB2 is modified into SigmaLB3 to store the past results of HELPER to a hash table ht. Helper in SigmaLB3 can reduce the recursion when the cache in ht hits. To reduce the memory usage, SigmaLB3 stores the representative vector $\boldsymbol{rep}(\boldsymbol{S})$, instead of individual vectors $\boldsymbol{S}$.

On the line 11, '$ht[\boldsymbol{rep}(\boldsymbol{S})] = \emptyset$' represents the condition that the value for the key $\boldsymbol{rep}(\boldsymbol{S})$ in ht is empty; meaning that no vectors P-equivalent to $\boldsymbol{S}$ have been computed yet. The line 11 checks ht and determines whether another HELPER$(\boldsymbol{S}', c')$ such that $\boldsymbol{S} \sim \boldsymbol{S}'$ and $c \geq c'$ has already been computed. If so, the condition of the **if** statement evaluates to false, and then the rest of the process, the lines 12–21, is skipped. Otherwise those lines are processed. To memorize the result of HELPER$(\boldsymbol{S}, c)$, the line 21 registers the key $\boldsymbol{rep}(\boldsymbol{S})$ and the value c to the hash table ht.

5.4 Computation Time

To test the time efficiency, we implemented SigmaLB1, SigmaLB2, and SigmaLB3 in Common Lisp (SBCL), and computed σ-$lb(\boldsymbol{\Lambda}(F))$ for all (40,320) 3-variable functions and 50,000 randomly-generated n-variable functions, where $n = 4, 5, \ldots, 10$. The programs were executed on a computer with Ubuntu 12.04LTS / Core i7-3820 CPU (3.6GHz). Table 1 shows the average computation time in milliseconds per function. The signs '–' in the table mean that we could not get the results within 10,000[ms] per function and gave up the computation. Among the three proposed algorithms, only SigmaLB3 obtained the lower bounds for functions with 9 or more variables in a practical computation time.

```
1:  function SigmaLB3(S): Integer        ▷ Input: S is a vector of non-negative integers.
2:      Var min: Integer;
3:      Var ht: Hash Table;
4:      procedure HELPER(S, c)    ▷ Input: S is a vector and c is an integer.
                                          ▷ Side Effect: Updating min.
5:          if $c + \lceil \log(\sum S + 1) \rceil \geq min$ then return ;
6:          end if
7:          if $S = 0$ then
8:              $min \leftarrow c$;
9:              return ;
10:         end if
11:         if $ht[rep(S)] = \emptyset$ or $c < ht[rep(S)]$ then
12:             if (all elements of S are even) then
13:                 HELPER($\Phi(S, 1), c + 1$);
14:             else
15:                 for $i \leftarrow 1$ to $n$ do
16:                     if ($S[i]$ is odd) then
17:                         HELPER($\Phi(S, i), c + 1$);
18:                     end if
19:                 end for
20:             end if
21:             $ht[rep(S)] \leftarrow c$;
22:         end if
23:     end procedure
24:     $min \leftarrow$ a large integer;
25:     HELPER($S, 0$);
26:     return $min$;
27: end function
```

Fig. 6 SigmaLB3: a fast algorithm for σ-$lb(\boldsymbol{S})$.

Table 1 Average computation time [ms] for σ-$lb(\boldsymbol{\Lambda}(F))$.

n	SigmaLB1	SigmaLB2	SigmaLB3
3	0.022	0.0055	0.0083
4	0.84	0.024	0.020
5	57.09	0.31	0.062
6	5396.30	6.66	0.15
7	–	243.15	0.29
8	–	7020.24	0.54
9	–	–	0.94
10	–	–	1.57

6. Experimental Results

To make an experimental comparison between the two lower bounds σ-$lb(\boldsymbol{\Lambda}(F))$ and $\lceil \log(\sigma(F) + 1) \rceil$, we computed them for all (40,320) 3-variable functions and 50,000 randomly-generated n-variable functions, where $n = 4, 5, \ldots, 10$, exactly the same as in Table 1. The averages of lower bounds for those functions are given in Table 2. Not only

Table 3 Distribution of functions over the difference between two lower bounds.

Diff.	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	$n=8$	$n=9$	$n=10$
0	32,021(79.4%)	27.9%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
1	8,299(20.6%)	72.1%	94.2%	0.0%	0.0%	0.0%	0.0%	0.0%
2	0	0.0%	5.8%	97.4%	0.0%	0.0%	0.0%	0.0%
3	0	0.0%	0.0%	2.6%	98.7%	43.3%	0.0%	0.0%
4	0	0.0%	0.0%	0.0%	1.3%	56.7%	99.7%	0.0%
5	0	0.0%	0.0%	0.0%	0.0%	0.0%	0.3%	99.8%
6	0	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.2%
Total	40,320 (100%)	100%	100%	100%	100%	100%	100%	100%

Table 2 Average of lower bounds.

n	$\lceil\log(\sigma(F)+1)\rceil$	$\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$
3	3.88	4.09
4	5.36	6.08
5	6.99	8.05
6	8.00	10.03
7	9.00	12.01
8	10.44	14.01
9	12.00	16.00
10	13.00	18.00

Table 4 Distribution of 3-variable functions over the difference between the minimum gate count and the lower bound.

	Number of functions	
Diff.	$\gamma(F)-\lceil\log(\sigma(F)+1)\rceil$	$\gamma(F)-\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$
0	1,462	2,049
1	9,026	12,055
2	19,043	19,189
3	10,251	6,823
4	538	204
Total	40,320	40,320

$\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$ is larger than $\lceil\log(\sigma(F)+1)\rceil$, but the difference between the two bounds increases with the number of variables.

Table 3 shows the distribution of functions over the difference between two lower bounds, $\sigma\text{-}lb(\boldsymbol{\Lambda}(F)) - \lceil\log(\sigma(F)+1)\rceil$, for $0, 1, \ldots, 6$. Table 3 also shows that the difference increases with n.

To check how much the difference between the minimum GC and the lower bound is, we experimented on all 40,320 3-variable functions since the minimum GCs, $\gamma(F)$, for those functions can be easily obtained by the exhaustive enumeration [18]. We calculated the difference $\gamma(F) - \sigma\text{-}lb(\boldsymbol{\Lambda}(F))$ for every function F and counted the number of functions for specified differences. We also performed the same experiments for $\gamma(F) - \lceil\log(\sigma(F)+1)\rceil$. The results are shown in Table 4. The number of functions whose lower bounds are equal to $\gamma(F)$ is 1,462 for $\lceil\log(\sigma(F)+1)\rceil$ while 2,049 for $\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$.

Table 5 shows the lower bounds for reversible benchmarks from the Reversible Logic Synthesis Benchmarks Page [21]. We added no extra lines (ancilla or garbage bits) to these benchmarks in our experiments. In the table, 'UB' is the known upper bound on GC for each benchmark; the one marked with '*' has been confirmed as exact minimum, and the one marked with '†' was synthesized with the NOT-CNOT-Toffoli (NCT) library instead of general Toffoli. It should be noted that our lower bounds for the general Toffoli library are also lower bounds for the NCT library. $\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$ produced better lower bounds than $\lceil\log(\sigma(F)+1)\rceil$ in many cases; the bounds have been improved particularly for benchmarks with relatively larger n like cycle10_2 and ham15. Although the lower bounds in Table 5 still differ significantly from the corresponding upper bounds, the improved results are valuable as the theoretical information in the field of reversible logic synthesis.

Table 5 Upper and lower bounds for benchmark functions.

Name	n	$\lceil\log(\sigma(F)+1)\rceil$	$\sigma\text{-}lb(\boldsymbol{\Lambda}(F))$	UB [21]
3_17	3	4	4	6*
4_49	4	6	7	12†
4b15g_1	4	5	6	15*
4b15g_2	4	6	7	15*
4b15g_3	4	5	6	15*
4b15g_4	4	5	6	15*
4b15g_5	4	6	6	15*
cycle10_2	12	5	10	19
ham3	3	4	4	5*
ham7	7	6	8	21
ham15	15	7	17	70
hwb4	4	6	7	11†
hwb5	5	7	9	24
hwb6	6	8	11	42
mod5adder	6	7	7	15
nth_prime3_inc	3	3	4	4*
nth_prime4_inc	4	5	6	11*
nth_prime5_inc	5	7	8	25
nth_prime6_inc	6	8	9	55

7. Conclusion

We proposed a new lower bound on the gate count for reversible functions, based on the vector information of the functions. The operation of halving the vector elements are utilized for calculating the bound. By introducing an operation of doubling the vector elements, we proved that the proposed bound is always better than or equal to the previous one. We experimented on randomly-generated reversible functions and reversible benchmarks, and have confirmed that the proposed bound is larger than the previous one in almost all cases.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 2450005.

References

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

[2] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation rules for designing CNOT-based quantum circuits," Proc. 39th DAC, pp.419–424, USA, 2002.

[3] D. Maslov, G. Dueck, and D. Miller, "Toffoli network synthesis with templates," IEEE Trans. Comput.-Aided. Des. Integr. Circuits Syst., vol.24, no.6, pp.807–817, June 2005.

[4] D. Miller and G. Dueck, "Spectral techniques for reversible logic synthesis," Proc. Reed-Muller 2003 Workshop, pp.56–62, Trier, Germany, March 2003.

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

[6] G. Dueck, D. Maslov, and D. Miller, "Transformation-based synthesis of networks of Toffoli/Fredkin gates," Proc. Canadian Conference on Electrical and Computer Engineering, vol.1, pp.211–214, May 2003.

[7] P. Kerntopf, "A new heuristic algorithm for reversible logic synthesis," Proc. 41st DAC, pp.834–837, 2004.

[8] D. Maslov, G. Dueck, and D. Miller, "Synthesis of Fredkin-Toffoli reversible networks," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.13, no.6, pp.765–769, June 2005.

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

[10] L. Storme, A. De Vos, and G. Jacobs, "Group theoretical aspects of reversible logic gates," J. Universal Computer Science, vol.5, no.5, pp.307–321, 1999.

[11] H. Thapliyal and N. Ranganathan, "Design of reversible sequential circuits optimizing quantum cost, delay and garbage outputs," ACM J. Emerging Technologies in Computing Systems, vol.6, no.4, article 14, Dec. 2010.

[12] H. Thapliyal and N. Ranganathan, "Design of efficient reversible logic based binary and BCD adder circuits," ACM J. Emerging Technologies in Computing Systems, vol.9, no.3, pp.17:1–17:31, Oct. 2013.

[13] M. Saeedi and I.L. Markov, "Synthesis and optimization of reversible circuits - A survey," ACM Comput. Surv., vol.45, no.2, article 21, Feb. 2013.

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

[15] M. Szyprowski and P. Kerntopf, "Reducing quantum cost in reversible Toffoli circuits," Proc. Reed-Muller 2011 Workshop, pp.127–136, Tuusula, Finland, May 2011.

[16] N.M. Nayeem and J.E. Rice, "Improved ESOP-based synthesis of reversible logic," Proc. Reed-Muller 2011 Workshop, pp.57–62, Tuusula, Finland, May 2011.

[17] M. Saeedi, M.S. Zamani, M. Sedighi, and Z. Sasanian, "Reversible circuit synthesis using a cycle-based approach," ACM J. Emerging Technologies in Computing Systems, vol.6, no.4, article 13, Dec. 2010.

[18] V.V. Shende, A.K. Prasad, I.L. Markov, and J.P. Hayes, "Synthesis of reversible logic circuits," IEEE Trans. Comput.-Aided. Des. Integr. Circuits Syst., vol.22, no.6, pp.710–722, June 2003.

[19] M. Higashiohno, T. Hirayama, and Y. Nishitani, "A lower bound on the number of Toffoli gates in reversible logic circuits," IEICE Trans. Fundamentals (Japanese Edition), vol.J92-A, no.4, pp.263–266, April 2009.

[20] M. Davio, J.P. Deschamps, and A. Thayse, Discrete and Switching Functions, McGraw-Hill International, 1978.

[21] D. Maslov, "Reversible logic synthesis benchmarks," http://webhome.cs.uvic.ca/˜dmaslov/

[22] M. Szyprowski and P. Kerntopf, "Estimating the quality of complexity measures in heuristics for reversible logic synthesis," Proc. IEEE Congress on Evolutionary Computation, pp.1–8, Barcelona, Spain, July 2010.

[23] 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.

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.

Hayato Sugawara received his B.E. degree from Iwate University, Morioka, Japan, in 2013. He is currently working toward his M.E. degree at Iwate University. His research interests include reversible logic synthesis and optimization algorithms.

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. Since 2000 he has been a professor in the Department of Electrical Engineering and Computer Science, Faculty of Engineering, Iwate University. His current research interests include switching theory, software engineering, and distributed algorithms.