IEEE Transactions on Quantum Engineering

Received May 24, 2021; revised September 16, 2021; accepted December 2, 2021; date of publication December 17, 2021; date of current version February 8, 2022.

Digital Object Identifier 10.1109/TQE.2021.3136195

Efficient Construction of a Control Modular Adder on a Carry-Lookahead Adder Using Relative-Phase Toffoli Gates

KENTO OONISHI[1,2], TOMOKI TANAKA[3,4], SHUMPEI UNO[4,5], TAKAHIKO SATOH[4,6], RODNEY VAN METER[4,7] (Senior Member, IEEE), AND NOBORU KUNIHIRO[8]

[1]Graduate School of Information Science and Technology, The University of Tokyo, Tokyo 113-8656, Japan
[2]Graduate School of Science and Technology, Keio University, Yokohama 223-8522, Japan
[3]Mitsubishi UFJ Financial Group (MUFG), Inc. and MUFG Bank, Ltd., Tokyo 100-8388, Japan
[4]Quantum Computing Center, Keio University, Yokohama 223-8522, Japan
[5]Mizuho Research and Technologies, Ltd., Tokyo 101-8443, Japan
[6]Graduate School of Media and Governance, Keio University Shonan Fujisawa Campus, Fujisawa 252-0882, Japan
[7]Faculty of Environment and Information Studies, Keio University Shonan Fujisawa Campus, Fujisawa 252-0882, Japan
[8]University of Tsukuba, Tsukuba 305-8573, Japan

Corresponding author: Kento Oonishi (e-mail: oonishi059@gmail.com)

This work was supported by Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP20J11754 and Grant JP21H03440, in part by the Ministry of Education, Culture, Sports, Science and Technology (MEXT) Quantum Leap Flagship Program under Grant Number JPMXS0118067285, and in part by Japan Science and Technology CREST under Grant JPMJCR14D6. The work of K. Oonishi was supported by a JSPS Fellowship for Young Scientists.

ABSTRACT Control modular addition is a core arithmetic function, and we must consider the computational cost for actual quantum computers to realize efficient implementation. To achieve a low computational cost in a control modular adder, we focus on minimizing KQ (where K is the number of logical qubits required by the algorithm, and Q is the elementary gate step), defined by the product of the number of qubits and the depth of the circuit. In this article, we construct an efficient control modular adder with small KQ by using relative-phase Toffoli gates in two major types of quantum computers: fault-tolerant quantum computers (FTQ) on the logical layer and noisy intermediate-scale quantum computers (NISQ). We give a more efficient construction compared with Van Meter and Itoh's, based on a carry-lookahead adder. In FTQ, T gates incur heavy cost due to distillation, which fabricates ancilla for running T gates with high accuracy but consumes a lot of especially prepared ancilla qubits and a lot of time. Thus, we must reduce the number of T gates. We propose a new control modular adder that uses only 20% of the number of T gates of the original. Moreover, when we take distillation into consideration, we find that we minimize KQ_T (the product of the number of qubits and T-depth) by running $\Theta(n/\sqrt{\log n})$ T gates simultaneously. In NISQ, CNOT gates are the major error source. We propose a new control modular adder that uses only 35% of the number of CNOT gates of the original. Moreover, we show that the $\mathrm{KQ}_{\mathrm{CX}}$ (the product of the number of qubits and CNOT-depth) of our circuit is 38% of the original. Thus, we realize an efficient control modular adder, improving prospects for the efficient execution of arithmetic in quantum computers.

INDEX TERMS Carry-lookahead adder, control modular adder, fault-tolerant quantum computers (FTQ), noisy intermediate-scale quantum computers (NISQ), Shor's algorithm.

I. INTRODUCTION

Recently, functional but imperfect quantum computers, called noisy intermediate-scale quantum computers (NISQ) [1] with machines from IBM [2], [3], Google [4], Rigetti [5], IonQ [6], and Honeywell [7] all accessible via the web, have emerged.

Many researchers have constructed simple quantum circuits for NISQ machines. Researchers at IBM implemented the first $15 = 3 \times 5$ factoring circuit on a liquid nuclear magnetic resonanc machine in 2001 [8]. Since then, researchers have implemented Shor's algorithm on a variety of machines [9]–[14], although care must be taken not to

extrapolate too far from these demonstrations [15]. Researchers have also demonstrated small instances of Grover's algorithm [16], [17].

However, we cannot realize large-scale quantum computation on NISQ, due to the high error rate. These errors propagate as the calculation proceeds, and we cannot extract the correct result. Thus, we must reduce the error rate in quantum computers. To realize computation with high accuracy, research on fault-tolerant quantum computers (FTQ) is proceeding [18]–[20].

Jones *et al.* [21] proposed a method for constructing FTQ as a layered architecture. Specifically, we conduct the accurate computation on the Logical layer, which is achieved using large numbers of physical qubits with errors.

However, T gates impose an additional cost when run on FTQ. By the Gottesman–Knill theorem [22], we can conduct classical simulation of quantum circuits composed only of Clifford gates, but to realize universal quantum computation, we require non-Clifford gates, such as a T gate, taking us into a realm that cannot be simulated classically. We achieve high-fidelity T gates by incorporating distillation [23], which requires a lot of logical qubits and a lot of time; research on optimization of distillation is being carried out [24]. In FTQ, we may realize large-scale quantum algorithms, such as Shor's algorithm [25] and Grover's algorithm [26]. Shor's algorithm is of particular interest *if* it can implemented effectively because it solves the factorization problem or the discrete logarithm problem in polynomial time, breaking the security of current cryptosystems, such as RSA [27] or elliptic curve [28], [29], whose security is based on the factorization problem or the discrete logarithm problem, respectively.

In Shor's algorithm, the control modular exponentiation step dominates the total cost, leading many researchers to study its construction [30]–[40]. The control modular exponentiation calculates

$$|j\rangle|1\rangle \to |j\rangle|a^j \bmod N\rangle \tag{1}$$

where a, N, and j are nonnegative integers satisfying $a < N$.

One strategy realizes a control modular exponentiation by the repeated calling of control modular additions. More precisely, this construction is realized by following two steps:

1) decomposition of a control modular exponentiation into control modular multipliers;
2) decomposition of a control modular multiplier into control modular adders.

The first decomposition is based on the following equation where j can be expressed in n-bit, namely $j = (j_{n-1} \dots j_0)_2$:

$$a^j \bmod N = \prod_{l=0}^{n-1} \left(a^{2^l} \bmod N\right)^{j_l} \bmod N \tag{2}$$

where a^j is decomposed into n-times multiplications, namely $a^{2^l j_l}$. For example, a exponentiation 2^5 is decomposed into $2^5 = 2^{101_2} = 2^4 \times 2^1$.

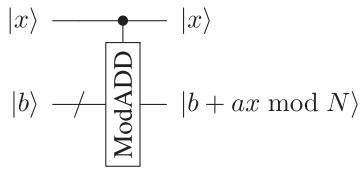

FIGURE 1. Overview of a control modular adder. The first register has a single qubit, which is used as a control bit. The second register has *n* qubits, which are used to store the result. *a* and *N* are *n*-bit classical numbers.

Next, we consider the second decomposition. In quantum computation, a multiplication ka is based on the following operation:

$$|k\rangle|0\rangle \to |k\rangle|0 + ka\rangle \to |k - a^{-1}(ka)\rangle|ka\rangle = |0\rangle|ka\rangle. \tag{3}$$

The operation of (3) requires an addition by the result of multiplication. This addition can be decomposed as follows, where a and b are nonnegative integers less than N, and k is a n-bit number expressed as $k = (k_{n-1} \dots k_0)_2$:

$$b + ak \bmod N = b + \sum_{l=0}^{n-1} \left(a2^{k_l} \bmod N\right) \bmod N. \tag{4}$$

For example, $6 \cdot 5$ is decomposed into $6 \cdot 5 = 6 \cdot (4 + 1) = 6 \cdot 4 + 6 \cdot 1$, because $5 = 101_2 = 4 + 1$.

From the aforementioned discussion, a control modular exponentiation is decomposed into control modular adders. Thus, if we reduce the cost of a control modular adder, the total cost of Shor's algorithm will shrink. In this article, we focus on the efficient construction of a control modular addition.

A. BACKGROUND

A control modular addition is defined by a control qubit x and n-bit numbers a, b, and N. a and b satisfy $0 \le a, b \le N - 1$, and a and N are classical numbers. A control modular addition calculates

$$|x\rangle|b\rangle \to |x\rangle|b + xa \bmod N\rangle. \tag{5}$$

An overview is shown in Fig. 1.

However, the optimal construction of a control modular adder is not obvious. A control modular adder is constructed from simple adders [31], [32], [34], [37]–[40], and there are many kinds of adders [38], [39], [41]–[45]. Previous constructions follow similar overall structure, but differ in detail. We need to determine which combination is the best.

This article focuses on minimizing KQ [46] to construct a circuit with low execution cost, where K is the number of logical qubits required by the algorithm, and Q is the elementary gate step. KQ is defined by the product of the number of qubits and the depth of the circuit. Minimizing KQ benefits both FTQ [21] and NISQ [17]. Much previous research focuses only on depth or the number of qubits, but reducing only one metric improves only one performance. We believe KQ more accurately represents the total resource

IEEE Transactions on Quantum Engineering

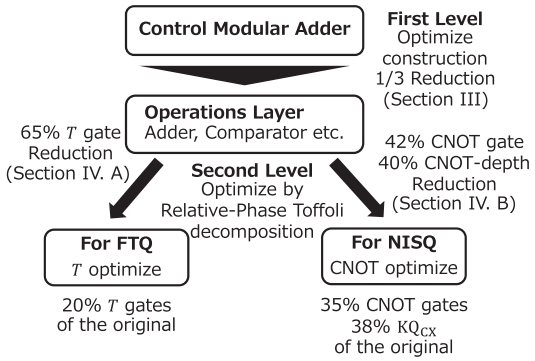

FIGURE 2. Optimization of a control modular adder. In first-level optimization, we optimize the construction of a control modular adder. In second-level optimization, we minimize KQ for FTQ or NISQ by using relative-phase Toffoli gates.

consumption, especially for deep circuits, capturing the total "qubit-time steps" of a circuit.

This article proposes our new circuit based on Van Meter and Itoh's construction [38], which uses three of Draper *et al.*'s carry-lookahead adders [42]. Van Meter and Itoh's construction has small KQ values than the other constructions but has room for further minimization of KQ. For example, Thapliyal *et al.* [47] proposed a means of minimizing the number of T gates in a carry-lookahead adder by using Gidney's relative-phase Toffoli gates (GRT) [44]. Thapliyal *et al.*'s construction reduces KQ in FTQ, but similar optimization can be applied to NISQ by Maslov's relative-phase Toffoli gates [48]. Thus, we reduce KQ by applying relative-phase Toffoli gates on the Van Meter-Itoh construction.

B. OUR CONTRIBUTION

In this article, we propose a method for optimizing a control modular adder based on a carry-lookahead adder for both FTQ and NISQ. We apply two-level optimization on the original Van Meter-Itoh construction [38] as in Fig. 2.

In first-level optimization, we optimize the construction of a control modular adder (see Section III). Specifically, we optimize by focusing on the efficiency of the comparator in a carry-lookahead adder and reduce some control operations by taking advantage of the classicality of a and N.

In second-level optimization, we minimize KQ for FTQ or NISQ by using relative-phase Toffoli gates (see Section IV). In this study, we assumed all qubits are connected, without considering the physical or logical topology [32], [49], [50]. We assume full connectivity because some current NISQ machines, such as IonQ [6] and Honeywell [7], realize full connectivity. Future work must consider the problem of mapping circuits to other NISQ machines and to FTQ machines.

First, we clarify the definition of KQ in each device, because the cost of gates is different in FTQ or NISQ. For many implementations, the most expensive gates are T gates [21] and CNOT gates [2], [3], respectively. We define KQ_T for use with FTQ and $\mathrm{KQ}_{\mathrm{CX}}$ on NISQ as the product of the number of qubits and T-depth or CNOT-depth, respectively. Next, we use GRT [44] in FTQ circuits and Maslov's relative-phase Toffoli gates [48] in NISQ circuits, instead of the standard Toffoli gates. However, the construction for FTQ does not consider the cost of distillation, and there is a tradeoff between T-depth and the number of T gates running simultaneously. We show that we achieve smallest KQ_T when we run $\Theta(n/\sqrt{\log n})$ T gates simultaneously.

II. PRELIMINARIES

In this article, we optimize a carry-lookahead adder by replacing Toffoli gates with relative-phase Toffoli gates. To maintain an accurate calculation, we must consider the role of Toffoli gates well. Moreover, we reduce computational costs by decomposing Toffoli gates into single-qubit gates and CNOT gates.

In Section II-A, we explain the quantum gate set used in this article. Next, to clarify the role of Toffoli gates, we review Draper *et al.*'s carry-lookahead adder [42] briefly in Section II-B. We explain T-minimization [47] by using GRT [44] in Section II-C. Finally, we review the general construction of a control modular adder in Section II-D.

A. QUANTUM GATE SET

In this article, we use the following quantum gate set.

1) *Clifford gates*: X gate, Y gate, Z gate, H gate, S gate, and CNOT gate.
2) *Non-Clifford gates*: T gate.

The CNOT gate is a two-qubit gate, and the others are one-qubit gates. We express X gates as $\oplus$ in the circuit.

In this article, we focus on the following two gates: T and CNOT:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & \exp\left(\frac{\mathrm{i}\pi}{4}\right) \end{bmatrix},$$

$$\mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{6}$$

B. DRAPER ET AL.'S CARRY-LOOKAHEAD ADDER

In this subsection, we use the same notations as in Draper *et al.*'s article [42]. First, we explain the calculation of $a+b$ when a and b are n-bit numbers. We express a as $a_{n-1}a_{n-2}\ldots a_0$ and b as $b_{n-1}b_{n-2}\ldots b_0$, where a_i and b_i are 0 or 1. To calculate $a+b$, we employ a carry c_i. Carry c_i is defined as an overflow from the $(i-1)$th bit to the ith bit. In more detail, we define c_i as

$$c_i = \begin{cases} 0 & \text{if } i = 0 \\ \left\lfloor \dfrac{a_{i-1} + b_{i-1} + c_{i-1}}{2} \right\rfloor & \text{otherwise.} \end{cases} \tag{7}$$

Then, $(a+b)_i$, the ith bit of $a+b$, is calculated as

$$(a+b)_i = a_i \oplus b_i \oplus c_i. \tag{8}$$

Thus, we need carries to calculate an addition.

Now, we give a brief explanation of a carry-lookahead adder. Before calculating an addition, we determine the propagation of a carry from the ith bit to the jth bit as a function of the following three conditions.

1) *propagate*: A carry is propagated from the ith bit to the jth bit. Namely, $c_j = c_i$.
2) *generate*: A carry is generated in the jth bit, namely $c_j = 1$, regardless of the value of c_i.
3) *kill*: A carry is killed in the jth bit, namely $c_j = 0$, regardless of the value of c_i.

To calculate the propagation, we define two functions $p[i, j], g[i, j] \in \{0, 1\}$. $p[i, j]$ is true when the carry from the ith bit to the jth bit should be propagated. Similarly, $g[i, j]$ is true when the carry out at the jth bit is true independent of the value of the carry in at the i bit. We do not need a separate function for *kill*, as its value can be inferred from p and g. By using these functions, we can calculate the propagation state over a wider span. Specifically, when $i < k < j$

$$p[i, j] = p[i, k] \wedge p[k, j] \tag{9}$$

$$g[i, j] = g[k, j] \oplus (g[i, k] \wedge p[k, j]) \tag{10}$$

where $\wedge$ is Boolean AND, and $\oplus$ is Boolean XOR. By using these properties, we calculate $c_j = g[0, j]$.

Now, we explain Draper *et al.*'s carry-lookahead adder for $|a\rangle|b\rangle \to |a\rangle|b+a\rangle$. This requires an additional n qubits for the carry register $|c\rangle$ and n qubits for register $|p\rangle$, containing $p[i, j]$. Thus, a carry-lookahead adder requires $4n$ qubits.

Now, we explain the implementation briefly. This implementation consists of five phases: initialization, P-rounds, G-rounds, C-rounds, and inverse P-rounds. In each round, we calculate the following, and clean $|p\rangle$ in inverse P-rounds.

1) *Initialization*: We calculate $g[i, i+1]$ in $|c_{i+1}\rangle$ and $p[i, i+1]$ in $|b_i\rangle$.
2) *P-rounds*: We calculate the p-function and write result in $|p\rangle$.
3) *G-rounds*: We calculate $|c_{2^k}\rangle$ $(k \in \mathbb{N})$ by calculating some g-function.
4) *C-rounds*: We calculate all carry $|c\rangle$ by calculating some g-function.

After inverse P-rounds, we calculate each bit of $a+b$ by using these carries $|c\rangle$. In this calculation, we run P-rounds and G-rounds simultaneously, and we run C-rounds and inverse P-rounds simultaneously. However, the value of carries remain on $|c\rangle$. Thus, we must clean $|c\rangle$ to $|0\rangle$ except for c_n. Draper *et al.* found that the value of carries c_i except c_n in $a+b$ is the same in $a + (2^n - 1 - a - b)$. Therefore, we erase carries by performing the addition $a + (2^n - 1 - a - b)$ on the lower $n-1$ qubits. The block-level circuit is shown in Fig. 3.

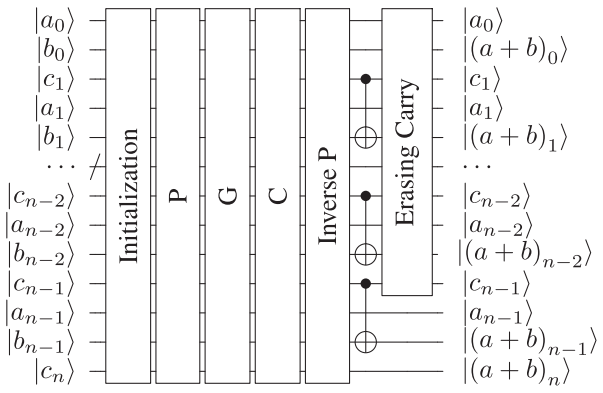

FIGURE 3. Block-level figure of Draper *et al.*'s carry-lookahead adder. In this figure, we sort qubits from the lowest qubits to the highest qubits, which is different from Fig. 1. $|c_i\rangle$ is given as $|0\rangle$ at the beginning of this circuit and these are cleared to $|0\rangle$ after erasing carry. The detailed circuit is shown in Appendix A.

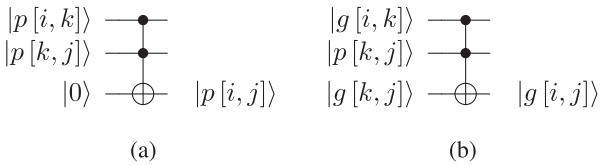

FIGURE 4. Calculation circuit of $p[i, j]$ and $g[i, j]$. (a) Calculation circuit of $p[i, j]$ as (9). (b) Calculation circuit of $g[i, j]$ as (10).

As noted previously, a carry-lookahead adder is mainly constructed by a calculation on p and g. We calculate p and g with (9) or (10), respectively, and those are implemented by Toffoli gates as shown in Fig. 4. The detailed explanation of Draper *et al.*'s adder, including which p-function or g-function we calculate, is given in Appendix A. In total, a carry-lookahead adder requires $10n$ Toffoli gates and $4n$ CNOT gates. Moreover, the Toffoli depth is $4 \log n$.

Up to this point, we have explained the construction of an adder. Draper *et al.* also proposed other operations, such as a subtractor and a comparator, based on their adder. The number of gates and the depth in a subtractor is almost the same as those in an adder. In a comparator, the number of gates is 60% of an adder and the depth is 50% of an adder. Draper *et al.* implement a comparator using only Initialization, P-rounds, G-rounds, and their inverses. More precisely, Draper *et al.* regard a and b as $2^{\lceil \log n \rceil}$-bit numbers by padding 0 in higher bits, but we do not use these qubits. If we calculate $p[i, j]$ or $g[i, j]$ when $i \leq n-1$ and $j \geq n$, we calculate $p[i, n]$ or $g[i, n]$, respectively. Then, we calculate $g[0, n]$ after G-rounds.

C. T-COUNT MINIMIZATION OF A CARRY-LOOKAHEAD ADDER

Thapliyal *et al.* [47] proposed T-count minimization by using relative-phase Toffoli gates. The standard Toffoli gate (ST) [22] decomposition is given in Fig. 5. However, we can calculate correctly even if we replace some Toffoli

IEEE Transactions on Quantum Engineering

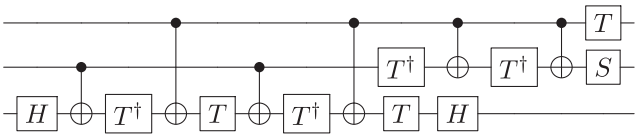

FIGURE 5. Standard decomposition of a Toffoli gate [22]. We call this decomposition ST. The control bits are the first and second qubits, and the target bit is the third qubit. This calculation preserves the phase.

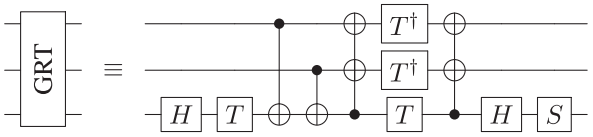

FIGURE 6. Gidney's relative-phase Toffoli gate [44] given by the unitary matrix (11). We call this decomposition GRT. The control bits are the first and second qubits, and the target bit is the third qubit. This calculation preserves the phase only when the target qubit is $|0\rangle$ on input.

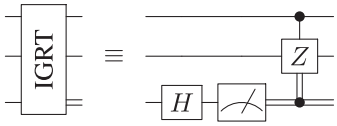

FIGURE 7. Inverse of Gidney's relative-phase Toffoli gate [44]. We call this decomposition IGRT. This calculation preserves the phase when we input $|000\rangle$, $|010\rangle$, $|100\rangle$, or $|111\rangle$, which are outputs of GRT having valid phase. Control-Z is a Clifford gate, and we use no *T* gate.

gates with GRT or its inverse (IGRT) [44]. GRT is shown in Fig. 6 and the corresponding unitary matrix of GRT in the computational basis is

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & i & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -i & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -i & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -i \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{11}$$

and we calculate correctly when the target bit is $|0\rangle$. IGRT is shown in Fig. 7. In the carry-lookahead adder, as in many circuits, we must clean our ancilla qubits, returning them to a known, disentangled state, typically $|0\rangle$. In this case, we can reduce our cost by measuring the ancilla on IGRT. Gidney's article [44] shows that using measurement reduces 2 T gates. Using measurement is better because one accurate T gate requires many measurements.

Thapliyal *et al.* proposed two constructions. The first construction replaced Toffoli gates in Initialization and P-rounds with GRT, and Toffoli gates in the inverse rounds with IGRT. Other Toffoli gates are replaced with ST. Thapliyal *et al.* call this construction qubit optimize. The number of qubits is $4n$ and the number of T gates is $40n$.

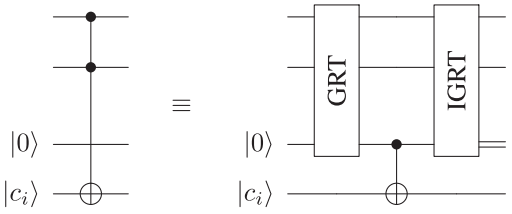

FIGURE 8. Replacing Toffoli gates in G-rounds and C-rounds in a *T*-optimized carry-lookahead adder. We call this decomposition PGRT. We replace the first Toffoli gate with GRT and the second Toffoli gate with IGRT. The third qubit is an ancilla qubit. This qubit is measured as part of executing IGRT and will be $|0\rangle$ after running PGRT.

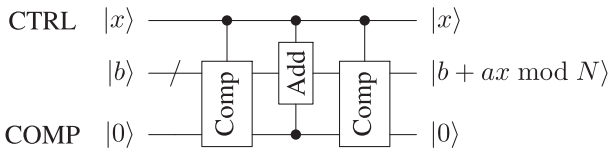

FIGURE 9. General construction of a control modular adder. Add means an adder, and Comp means a comparator. CTRL has a single qubit, which is used to hold the value of the control. $|b\rangle$ has n qubits, which are used to hold the result of a control modular addition. COMP has a single qubit, which is used to hold the result of a comparison. a and N are classical numbers.

The second construction replaced all Toffoli gates with GRT or IGRT, increasing the required number of ancilla qubits. Thapliyal *et al.* call this construction T-optimize. Specifically, we replace Toffoli gates in Initialization, P-rounds, and the inverse of these similarly as the first construction. We replace Toffoli gates in G-rounds and C-rounds by the pair of GRT and IGRT as in Fig. 8. We call these gates PGRT, where P is the abbreviation of "pair." In this construction, Thapliyal *et al.* claim that the number of qubits is $6n$ and the number of T gates is $20n$. However, we recalculated these results and our results differ from results in [47]. In our result, the number of qubits is $4.5n$ and the number of T gates is $28n$. The difference in the number of qubits occurs from our method for preparing ancilla qubits. Thapliyal *et al.* prepare new ancilla qubits for G-rounds and C-rounds, respectively, while they recycle ancilla qubits for P-rounds. We apply this to G-rounds and C-rounds similarly.

D. GENERAL CONSTRUCTION OF A CONTROL MODULAR ADDER

In this subsection, we explain the calculation of

$$|x\rangle|b\rangle|0\rangle \rightarrow |x\rangle|b + ax \bmod N\rangle|0\rangle. \tag{12}$$

The general construction of a control modular adder is shown in Fig. 9. The first register has a single qubit, which is used to hold the value of the control. We call this the CTRL qubit. The second register has n qubits, which are used to hold the result of a control modular addition. The third register has a single qubit, which is used to hold the result of a comparison temporarily. We call this the COMP qubit. Specifically, we determine whether we subtract N or not based on COMP.

We conduct a comparator with one control qubit and an adder with two control qubits, and we write these as a C-comparator and a CC-adder, respectively.

To execute a control modular adder, we conduct operations in the following order.

1) We compare the second register $|b\rangle$ and the classical value $N-a$. If $b \geq N-a$, namely $a+b \geq N$, we flip COMP.
2) If both CTRL and COMP are 1, we subtract $N-a$ from the second register. If CTRL is 1 and COMP is 0, we add a. Otherwise, we add no value.
3) If the second register is strictly less than a, we flip COMP.

III. FIRST-LEVEL OPTIMIZATION: OUR CONSTRUCTION OF A CONTROL MODULAR ADDER

In this section, we explain first-level optimization on the original construction [38]. In the general construction, a comparator has about half the depth of a carry-lookahead adder. Thus, by constructing a carry-lookahead adder using the same general construction, the depth is about the same as two adders, because a carry-lookahead adder is composed of two comparators and one adder. In the original construction, we use three adders. Thus, we use only two-thirds of KQ of the original construction when comparing two constructions of a control modular adder. The original construction applies two optimizations on repeating control modular adders. Our construction will be more efficient by adopting the same optimizations with some overhead, but the detail, such as the amount of overhead, should be evaluated in future work.

Based on the aforementioned discussion, we need to give the construction of the following on a carry-lookahead adder:

1) C-comparator (see Section III-A);
2) CC-adder (see Section III-B).

In this construction, we do not decompose Toffoli gates, because the decomposition of Toffoli gates is different in FTQ or NISQ, respectively. Thus, we leave Toffoli gates as they are, and we consider the decomposition of Toffoli gates in Section IV.

In our construction, we consider the classicality of a and N as described by Markov and Saeedi [35] to realize higher efficiency. Moreover, we give a construction of C-comparator that is not given in the original article. By doing these, we propose a circuit construction of a control modular adder.

Based on Fig. 9, we construct our circuit as shown in Fig. 10. We add the second n-qubit ancilla register for embedding value with CTRL. In addition to these registers, we use the carry register $|c\rangle$ with n qubits and the p-function register $|p\rangle$ with n qubits to realize the carry-lookahead adder, not represented in Fig. 10. Thus, our control modular adder requires $4n+2$ qubits. The number of gates and the depth is given in Table I, and the breakdown of this is given in Table V in Appendix B. Now, we explain the C-comparator and the CC-adder briefly.

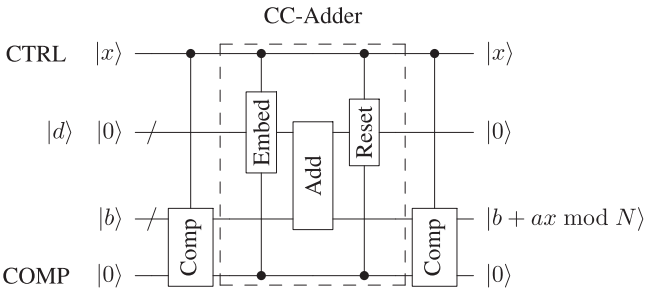

FIGURE 10. Our construction of a control modular adder based on Fig. 9. A CC-adder is constructed by embedding, an adder, and resetting. Then, we add the second register $|d\rangle$ as an n-qubit ancilla for embedding the value based on CTRL. The carry register $|c\rangle$ with n qubits and the p-function register $|p\rangle$ with n qubits are not represented in this figure for visibility. In a C-comparator, we do not use the second register. In total, our control modular adder requires $4n+2$ qubits.

TABLE I Gate Count and Depth of Our Proposed Control Modular Adder

	Count		Depth	
Operation	Toffoli	CNOT	Toffoli	CNOT
C-comparator (twice)	$4n$	n	$2\log n$	$O(1)$
CC-adder	$9.5n$	$4.75n$	$4\log n$	$2\log n$
Total	$17.5n$	$6.75n$	$8\log n$	$2\log n$

The breakdown of this is shown in Table V in Appendix B.

A. CONSTRUCTION OF A C-COMPARATOR

In a C-comparator, only COMP is changed and other qubits do not change. Thus, to implement a C-comparator, it is sufficient that we add control operations only on the gates including COMP and remain other gates.

In our construction of a control modular adder, we use two types of C-comparators. In the first C-comparator, we flip COMP if CTRL is 1 and $b \geq N-a$. In the final C-comparator, we flip COMP if CTRL is 1 and $b < a$. In both cases, we judge whether $b \geq d$ or $b < d$ with a classical value of d.

We construct these operations taking advantage of the classicality of d. The intuitive explanation of this operation is that we calculate $b+(2^n-d)$ and check whether there is an overflow in the nth bit. Specifically

$$b+\left(2^n-d\right)=2^n+(b-d) \tag{13}$$

and there is an overflow when $b \geq d$. This construction is similar to previous constructions by Markov and Saeedi [35], but slightly different from them because our construction does not require X gates on $|b\rangle$. The number of gates and the depth is given in Table I. The detailed construction is given in Appendix B. The block-level construction of our C-comparator is given in Fig. 11, and the example circuits are shown in Figs. 22 and 24.

B. CONSTRUCTION OF A CC-ADDER

In a CC-adder, we embed values before and after an adder, similar to a C-adder [39]. Based on this construction, we apply optimization by considering the classicality of a and

IEEE Transactions on Quantum Engineering

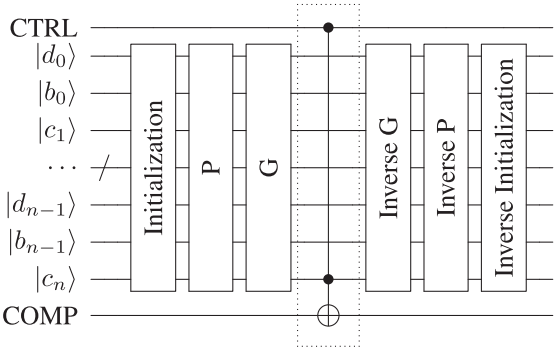

FIGURE 11. Block-level view of our construction of a C-comparator. In this figure, we sort qubits from the low-order qubits to the high-order qubits, top to bottom. This circuit is symmetric about the Toffoli gate surrounded by a dotted box. $|c_i\rangle$ is given as $|0\rangle$ at the beginning of this circuit and these are cleared back to $|0\rangle$ after the computation. The example circuits are shown in Figs. 22 and 24 in Appendix B.

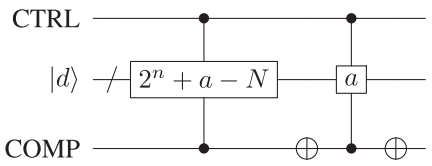

FIGURE 12. Block-level diagram of the embedding circuit. We omit $|b\rangle$ in Fig. 10. We embed $2^n + a - N$ or a on $|d\rangle$ based on CTRL and COMP. The example circuit of the embedding is shown in Fig. 23 in Appendix B.

N. From this point forward, we mainly focus on embedding on $|d\rangle$. In a CC-adder, we conduct the following.

1) If CTRL is 1 and COMP is 1, we add a and subtract N. This operation can be realized by adding $2^n + a - N$ and disregarding the calculation of a carry c_n.
2) If CTRL is 1 and COMP is 0, we add a.
3) Otherwise, we add no value.

Thus, the embedding is conducted as in Fig. 12. The resetting is conducted by inverting the embedding circuit.

After embedding, we apply a standard adder. Then, we conduct two optimizations as follows.

1) Disregarding gates including $|g[0, n]\rangle$.
2) Eliminating gates in Initialization where we know the control bit is 0.

The number of gates and the depth is given in Table I. The detailed construction and example circuit of a CC-adder are given in Appendix B.

IV. SECOND-LEVEL OPTIMIZATION: CONSTRUCTING A CONTROL MODULAR ADDER FOR FTQ AND NISQ DEVICES

In this section, we explain our second-level optimization. We evaluate the computational cost for both FTQ on the logical layer, and NISQ, focusing on the decomposition of Toffoli gates. We define KQ more specifically for FTQ and NISQ and minimize this value. For FTQ, we minimize the number

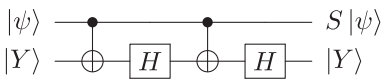

FIGURE 13. Running an S gate [21]. The second qubit is $|Y\rangle = (|0\rangle + i|1\rangle)/\sqrt{2}$. Assuming correct operation on top of error correction, this ancilla passes through the gate execution unmodified, allowing it to be reused.

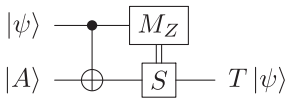

FIGURE 14. Running a T gate [21]. The second qubit $|A\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$; the $|A\rangle$ state is consumed in the process, with the consequence that creation of high-fidelity $|A\rangle$ states is one factor limiting performance.

of T gates by using Gidney's relative-phase Toffoli gates. However, this construction does not take into consideration the cost of distillation. We take into account the cost of distillation by finding the maximal number of T gates that should be run simultaneously, optimizing KQ_T. For NISQ, we apply Maslov's relative-phase Toffoli gates with a small number of CNOT gates [48] and minimize KQ_{CX}. By doing these, we propose a control modular adder that is more efficient than Van Meter and Itoh [38], called the original construction in this section. In the following discussion, we disregard the rounds with $O(1)$ gates. In this section, we explain the optimization for FTQ in Section IV-A and the optimization for NISQ in Section IV-B.

A. COMPUTATIONAL COST ON THE FTQ LOGICAL LAYER

Next, we consider the optimal circuit for FTQ on the Logical layer, using Jones *et al.*'s [21] architecture as a model. This architecture, in common with other error corrected-architectures, provides a fundamental gate set consisting of X, Y, Z, CNOT, and H gates, and measurement; here, we ignore qubit movement in the surface code. To run an S gate, we prepare an ancilla qubit $|Y\rangle = (|0\rangle + i|1\rangle)/\sqrt{2}$ and run the circuit shown in Fig. 13. An $S^\dagger$ gate can be realized by the reverse circuit of Fig. 13.

To achieve universal computation, we also need a non-Clifford gate; the choice of T is typical. To run a T gate, we prepare an ancilla qubit $|A\rangle = (|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$ and run the circuit shown in Fig. 14. To run a $T^\dagger$ gate, we apply an $S^\dagger$ gate instead of an S gate. To realize accurate T gates, we must prepare accurate $|A\rangle$ state defined by $(|0\rangle + e^{i\pi/4}|1\rangle)/\sqrt{2}$. Preparing $|A\rangle$ is done by *distillation*, as shown in Fig. 25 in Appendix E. This distillation circuit requires 15 qubits and six time steps, even assuming all of the CNOT gates can be implemented concurrently, but this is difficult to realize. Distillation is an expensive operation, and its optimization is an ongoing topic of research [24]. Thus, a T gate is the greatest factor in the cost of an FTQ circuit, leading us to focus on reducing the number of T gates.

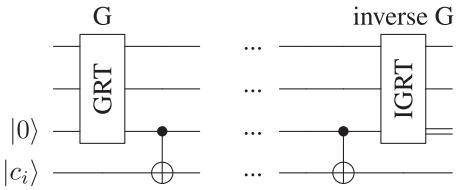

FIGURE 15. Our construction of G-rounds and inverse G-rounds in a C-comparator. In Fig. 8, we apply IGRT after the first cnot gate immediately in G-rounds and inverse G-rounds. In our construction, we calculate the result of GRT in the third ancilla qubit and preserve this qubit until the corresponding Toffoli gate in inverse G-rounds. Then, we clear this ancilla qubit by IGRT.

TABLE II *T*-Count of Our Control Modular Adder and Prior Work

Construction	#comparators	#adders	Total T-count
Van Meter and Itoh [39]	0	3	$210n$
Draper *et al.* [43]	2	1	$122.5n$
Thapliyal *et al.* (qubit-optimize) [48]	2	1	$75n$
Thapliyal *et al.* (T-optimize) [48]	2	1	$51n$
Ours	2	1	$43n$

The latter four constructions are based on our construction proposed in Section III. The breakdown of the latter four constructions is shown in Table VI in Appendix D.

We now minimize the number of T gates on our control modular adder. We adopt the Thapliyal construction with minor modifications, namely the replacement into relative-phase Toffoli gates, except G-rounds in a C-comparator. We employ GRT in G-rounds and IGRT in inverse G-rounds as Fig. 15. Our construction calculates correctly because Toffoli gates in G-rounds and inverse G-rounds are symmetric about the Toffoli gate surrounded by a dotted box as the block-level circuit of a C-comparator shown in Fig. 11.

Our construction requires an additional n qubits to preserve in a C-comparator. Fortunately, we do not use n qubits for $|d\rangle$ in Fig. 10. Thus, we realize this construction without an overhead of qubits. We give example circuits as Figs. 22 and 24 in Appendix B.

The computational cost of our control modular adder is shown in Table II, and the breakdown of constructions based on our construction is given in Table VI in Appendix D. From Table II, our construction requires $43n$ T gates. We call this construction a T-optimal control modular adder. The original construction requires $30n$ Toffoli gates, which when implemented using ST (each requiring seven T gates) gives $210n$ T gates in total. Thus, our construction requires only $43/210 \approx 20\%$ T of the number of T gates of the original construction.

Now, we focus on KQ of a T-optimal control modular adder. In this circuit, we use $O(n)$ qubits and $O(\log n)$ depth, giving a KQ of $O(n \log n)$. However, we do not consider the computational costs for distillation in this calculation. We can trade space for time, with substantial flexibility, by allocating more qubits to ancilla "factories," corresponding to increasing the number of T gates that are in concurrent execution [20], [51].

To realize an efficient circuit, we should consider the trade-off between the depth and the number of qubits allocated for distillation. For example, Kim *et al.* [52] showed that it is possible to run Shor's algorithm with as little as 2% of the qubits dedicated to distillation, but this construction runs only a single T gate at a time. Since the circuit still requires $O(n)$ T gates, this construct is unable to run in depth $O(\log n)$ and is instead still constrained to $O(n)$ depth. To realize smaller KQ, we must run many T gates parallel. However, there is an upper bound on the number of T gates that can be usefully run in parallel, with the depth limited by the cascading reuse of the qubits. Paler and Basmadjian also consider this problem [53], and they have concluded that we must determine optimal scheduling methods for T gates. To realize an accurate estimate of the cost and to enable fair comparison with prior research, we must take into account the T gate costs, including the space for distillation [21], [51], [54], allowing a circuit to run at "the speed of data" [54].

However, it is difficult to calculate computational costs for distillation precisely because the cost depends on many architecture-specific parameters. Instead of KQ, we define a new index KQ_T, defined as the product of the number of logical qubits and T-depth. We define n_T as the T-width, the upper bound of the number of T gates running simultaneously. We assume that we require a constant c_g logical qubits for the distillation step. By calculating n_T minimizing KQ_T, we reduce the computational cost of our control modular adder.

In the aforementioned discussion, our control modular adder uses $4n + 2$ qubits for calculation, as explained in Section III. In addition, we require ancilla qubits for running n_T T gates. Specifically, to run one T gate, we require one qubit $|Y\rangle$ for running S gates and c_g qubits for generating $|A\rangle$. Thus, when we run n_T T gates simultaneously, we use the following qubits:

1) $|y\rangle$ (Contains $|Y\rangle$ states) n_T qubits;
2) $|g\rangle$ (Generates $|A\rangle$ states) $c_g n_T$ qubits.

The number of qubits in $|y\rangle$ is given as n_T because we consume one S gate in each T gate. Then, the number of qubits is

$$4n + (c_g + 1)n_T + 2. \tag{14}$$

Now, we calculate T-depth. To calculate T-depth, we assume that we run GRT with the same timing, and each GRT has 2 T-depth from Fig. 6. T-depth depends on n_T as Fig. 16. Then, T-depth is

$$\frac{86n}{n_T} + 12 \log n_T - 12. \tag{15}$$

The detailed calculation is given in Appendix C.

Now, we minimize KQ_T on n_T. KQ_T is

$$\left(4n + (c_g + 1)n_T + 2\right)\left(\frac{86n}{n_T} + 12 \log n_T - 12\right). \tag{16}$$

We minimize this on $n_T > 0$.

IEEE Transactions on Quantum Engineering

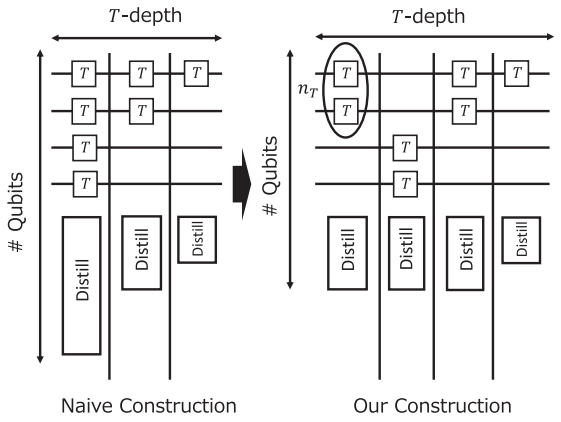

FIGURE 16. Calculating *T*-depth. Distill means distillation circuits. In the naive construction, we run as many *T* gates as possible. In our construction, we restrict the upper bound of the number of simultaneous *T* gates to n_T. When we reduce n_T, the total number of qubits is smaller and *T*-depth is larger.

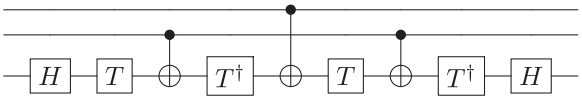

FIGURE 17. Relative-phase Toffoli gate with three cnot (RT3). This calculation changes the phase when we input $|1\rangle|0\rangle|1\rangle$, $|1\rangle|1\rangle|0\rangle$, and $|1\rangle|1\rangle|1\rangle$. We call the inverse circuit of RT3, IRT3.

Letting the expression in (16) be $f(n_T)$, we see that

$$\frac{\mathrm{d}^2 f(n_T)}{\mathrm{d}n_T^2} > 0 \tag{17}$$

in $n_T > 0$. Thus, $f(n_T)$ is a convex function and it is sufficient to search for only one optimal value of n_T. Then, the optimal value

$$n_T = \sqrt{\frac{86}{3(c_g+1)}}\frac{n}{\sqrt{\log n}}. \tag{18}$$

Thus, $O(\frac{n}{\sqrt{\log n}})$ T-width minimizes KQ_T. Plugging this value into (16)

$$4n + (c_g+1)n_T + 2 \sim 4n \tag{19}$$

$$\frac{86n}{n_T} + 12\log n_T - 12 \sim 12\log n. \tag{20}$$

Therefore, the dominant term of KQ_T is $48n\log n$.

B. OPTIMIZATION FOR NISQ

Now, we propose a form of the control modular adder reducing CNOT gates. To reduce this number, we review the decomposition of Toffoli gates into CNOT gates. We use relative-phase Toffoli gates with differences in phase as in Figs. 17 and 18, proposed by Maslov [48]. The corresponding unitary

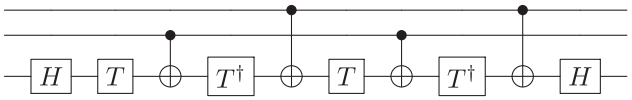

FIGURE 18. Relative-phase Toffoli gate with four cnot (RT4). This calculation change the phase when both control bits are 1. We call the inverse circuit of RT4, IRT4.

matrix of Fig. 17 in the computational basis is

$$\begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&-1&0&0\\ 0&0&0&0&0&0&0&-i\\ 0&0&0&0&0&0&i&0 \end{bmatrix}. \tag{21}$$

This calculation changes the phase when we input $|1\rangle|0\rangle|1\rangle$, $|1\rangle|1\rangle|0\rangle$, or $|1\rangle|1\rangle|1\rangle$. We call this relative-phase Toffoli gate RT3, and we call its inverse IRT3. The corresponding unitary matrix of Fig. 18 in the computational basis is

$$\begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&-i\\ 0&0&0&0&0&0&-i&0 \end{bmatrix}. \tag{22}$$

This calculation changes the phase when both control bits are 1. We call this relative-phase Toffoli gate RT4, and its inverse IRT4. By using these relative-phase Toffoli gates, we reduce the number of CNOT gates. Next, we address which Toffoli gates can be replaced with relative-phase Toffoli gates.

First, we consider which Toffoli gates can be replaced in a C-comparator. The structure of a C-comparator is shown in Fig. 11, and we give an example circuit in Figs. 22 and 24 in Appendix B. In these figures, all Toffoli gates are symmetric about the Toffoli gate surrounded by a dotted box, in the middle of the circuit. Thus, we can replace the Toffoli gates to the left of the dotted box by RT3 and those to the right of the box by IRT3. Therefore, we can replace all of the Toffoli gates in a C-comparator except this middle one with RT3 or IRT3.

Next, we address which Toffoli gates can be replaced in a CC-adder, and find that those in P-rounds can be replaced by RT3 and those in inverse P-rounds by IRT3. The other Toffoli gates used calculate the value of carries, and these carries are cleared after the calculation. In the calculation of carries, the values of the control bits change between calculating a carry

TABLE III CNOT Count of Our Control Modular Adder and Prior Work

Construction	#comparators	#adders	Total CNOT count
Van Meter and Itoh [39]	0	3	$184.5n$
Draper *et al.* [43]	2	1	$111.75n$
Thapliyal *et al.* (qubit-optimize) [48]	2	1	$88n$
Thapliyal *et al.* (T-optimize) [48]	2	1	$104n$
Ours	2	1	$64.75n$

The latter four constructions are based on our construction proposed in Section III. The breakdown of the latter four constructions is Shown in Table VII in Appendix D.

TABLE IV KQ_{CX} of Our Control Modular Adder and Prior Work

Construction	#qubits	The depth of the circuit	KQ_{CX}
Van Meter and Itoh [39]	$4n$	$78 \log n$	$312n \log n$
Draper *et al.* [43]	$4n$	$50 \log n$	$200n \log n$
Thapliyal *et al.* (qubit-optimize) [48]	$4n$	$50 \log n$	$200n \log n$
Thapliyal *et al.* (T-optimize) [48]	$4.5n$	$66 \log n$	$297n \log n$
Ours	$4n$	$30 \log n$	$120n \log n$

The latter four constructions are based on our construction proposed in Section III. The breakdown of the latter four constructions are shown in Table VIII in Appendix D.

and erasing it, which would seem to rule out using anything but pure Toffoli gates. However, looking more closely, we see that the value of a carry changes at most once, namely when both control bits are $|1\rangle$. Thus, if we calculate correctly in the other situations, we can calculate and clear carries correctly. RT4 satisfies this. Therefore, we can replace Toffoli gates by RT4 in the Initialization, G-rounds, and C-rounds, and we can replace Toffoli gates by IRT4 in the inverse rounds.

As a result, the cost of our control modular adder is shown in Tables III and IV. The breakdown of those based on our construction are shown in Tables VII and VIII in Appendix D. From Tables III and IV, our construction is better in terms of both the number of CNOT gates and CNOT-depth. Now, we compare our circuit to the original construction.

First, we compare the CNOT count. Our construction requires $64.75n$ CNOT gates. The original construction requires $30n$ Toffoli gates implemented by ST using six CNOT gates, and we use an additional $4.5n$ CNOT gates in embedding or resetting. Thus, the original construction requires $184.5n$ CNOT gates in total. Therefore, our construction reduces the number of CNOT gates to only 35% of the number in the original.

Next, we compare KQ_{CX}, defined as the product of the number of qubits and CNOT-depth. Our construction requires $120n \log n$ KQ_{CX}. The original construction requires $12 \log n$ Toffoli depth implemented by ST requiring 6CNOT-depth, and we require $6 \log n$ CNOT-depth for the embedding step. Thus, the original construction requires $78 \log n$ CNOT-depth and $312n \log n$ KQ_{CX}. Therefore, our construction requires only 38% of the KQ_{CX} of the original construction.

V. CONCLUSION

In this article, we proposed a method of optimizing a control modular adder based on a carry-lookahead adder [42] and Van Meter and Itoh's construction [38]. First, we show that the general construction given as Fig. 9 is about two-third of the KQ of the original construction. Then, we construct a more efficient circuit. We evaluate the computational cost in FTQ and we show that our circuit requires only 20% of the T gates of the original. Moreover, we show that our circuit achieves its minimum KQ_T when we run $\Theta(\frac{n}{\sqrt{\log n}})$ T gates simultaneously. Finally, we propose an efficient circuit for use in the NISQ era, and we show that our circuit requires only 35% of the CNOT gates and 38% KQ_{CX} of the original.

In this article, we have focused on optimizing Toffoli gates by using relative-phase Toffoli gates. However, in previous research [55], [56], other researchers have used gates such as Fredkin and Peres gates. These gates also may be simplified by replacing them with relative-phase gates. Thus, we expect that those circuits would also show an improvement with these techniques applied.

In this article, we have considered only the single control modular addition. In additional future work, the circuits that postpone and summarize multiple modular arithmetic operations, as proposed by Van Meter and Itoh [38], should be addressed using similar optimization techniques. In addition, it is important to minimize KQ by reordering gates [36], [57].

Our construction does not consider the architecture of quantum computers as linear nearest neighbor architecture [32], [49], [50]. Thus, in the next step, we will consider the appropriate architecture and additional cost for our construction.

Finally, we focused only on the Logical layer of FTQ in this study. In future work, we must consider the mapping to physical qubits, as well as distillation protocols.

APPENDIX

A. DETAILED EXPLANATION OF DRAPER ET AL.'S CARRY-LOOKAHEAD ADDER

Draper *et al.*'s carry-lookahead adder is given as follows.

1) *Initialization* (n Toffoli gates and n CNOT gates). We calculate $g[i, i+1]$ and $p[i, i+1]$ $(0 \leq i \leq n-1)$, as

IEEE Transactions on Quantum Engineering

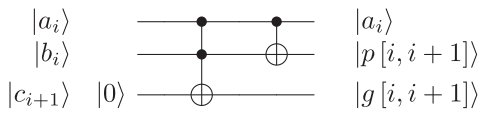

FIGURE 19. Calculation circuit of $g[i, i+1]$ and $p[i, i+1]$ $(0 \le i \le n-1)$. We use $|c_{i+1}\rangle$ as the third qubit. We can run these gates simultaneously for $i = 0$ to $n-1$.

follows:

$$g[i, i+1] = \begin{cases} 1, & \text{if } a_i = b_i = 1 \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

$$p[i, i+1] = \begin{cases} 1, & \text{if } a_i + b_i = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

The circuit calculating these is shown in Fig. 19.

2) *P-rounds* (n Toffoli gates and $\log n$ Toffoli depth). We calculate the p-function by using (9). We use a parameter t_p representing the range of the propagation of carry. We increase t_p from 1 to $\lfloor \log n \rfloor - 1$. In each t_p, we calculate $p[2^{t_p} i, 2^{t_p}(i+1)]$ $(1 \le i \le \lfloor n/2^{t_p} \rfloor - 1)$ by setting $|p[2^{t_p} i, 2^{t_p}(i+1/2)]\rangle$ and $|p[2^{t_p}(i+1/2), 2^{t_p}(i+1)]\rangle$ as the control qubits in Toffoli gate in Fig. 4(a). These Toffoli gates are applied simultaneously in each t_p.
3) *G-rounds* (n Toffoli gates and $\log n$ Toffoli depth). We calculate $|c_{2^k}\rangle$ $(k \in \mathbb{N} \cup \{0\})$ by using (10). We use a parameter t_g similar to the way we used it in P-rounds. We increase t_g from 1 to $\lfloor \log n \rfloor$. In each t_g, we calculate $g[2^{t_g} i, 2^{t_g}(i+1)]$ $(0 \le i \le \lfloor n/2^{t_g} \rfloor - 1)$ by setting $|c_{2^{t_g} i + 2^{t_g - 1}}\rangle$ and $|p[2^{t_g}(i+1/2), 2^{t_g}(i+1)]\rangle$ as the control qubits and $|c_{2^{t_g}(i+1)}\rangle$ as the target qubit in the Toffoli gate in Fig. 4(b). These Toffoli gates are applied simultaneously to each t_g. Moreover, G-rounds with t_g can be run in parallel with former P-rounds with $t_g + 1$.
4) *C-rounds* (n Toffoli gates and $\log n$ Toffoli depth). We calculate all carries $|c\rangle$ by using (10). We use a parameter t_c similar to the way we used it in P-rounds. We decrease t_c from $\lfloor \log(2n/3) \rfloor$ to 1. In each t_c, we calculate $|c_{2^{t_c} i + 2^{t_c - 1}}\rangle$ $(1 \le i \le \lfloor (n - 2^{t_c - 1})/2^{t_c} \rfloor - 1)$ by setting $|c_{2^{t_c} i}\rangle$ and $|p[2^{t_c} i, 2^{t_c} i + 2^{t_c - 1}]\rangle$ as the control qubits and $|c_{2^{t_c} i + 2^{t_c - 1}}\rangle$ as the target qubit in Toffoli gate in Fig. 4(b). These Toffoli gates are applied simultaneously in each t_c.
5) *Inverse P-rounds* (n Toffoli gates and $\log n$ Toffoli depth). We apply the same gates as P-rounds in the reverse order. Rounds with t_p can be run in parallel with former C-round with $t_p + 1$.
6) *Calculating* $|a+b\rangle$ (n CNOT gates). We calculate $(a+b)_i$ $(0 \le i \le n-2)$ on $|b_i\rangle$. We apply CNOT gates with the control qubit of $|c_{i+1}\rangle$ and the target qubit of $|b_{i+1}\rangle$. These CNOT gates are applied simultaneously.
7) *Erasing Carry* ($5n$ Toffoli gates, $2n$ CNOT gates, and $2 \log n$ Toffoli depth). We erase all carries by applying the inverse circuit of $a + (2^n - 1 - a - b)$ on the lower $n-1$ bits, as shown in Fig. 20. We apply gates before P-rounds and after inverse Initialization to erase carries. We call these gates PE-rounds and inverse PE-rounds, respectively.

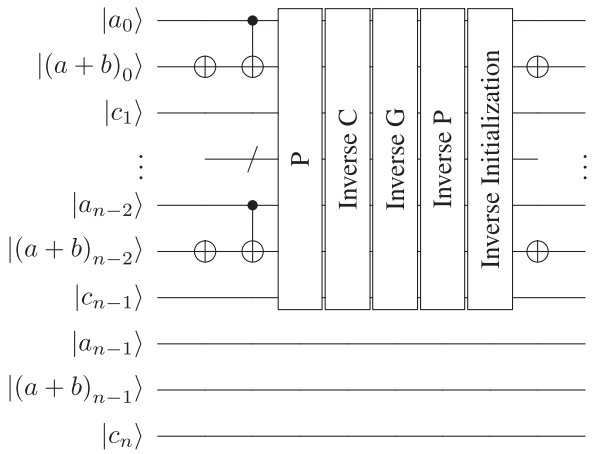

FIGURE 20. Erasing $|c\rangle$. We apply gates only on the lower $n-1$ qubits of $|a\rangle$, $|b\rangle$, and $|c\rangle$. We apply the same gates in omitted qubits $|a_i\rangle$, $|(a+b)_i\rangle$, and $|c_{i+1}\rangle$. The P-rounds and inverse C-rounds can be run in parallel, as can the inverse G-rounds and inverse P-rounds. We define PE-rounds as the gates before P-rounds, and inverse PE-rounds as the gates after inverse Initialization.

Now, we show the example circuit of Draper *et al.*'s carry-lookahead adder as given in Fig. 21. In this example, we define a and b as 6-bit values, and we calculate $|a\rangle|b\rangle \to |a\rangle|a+b\rangle$. In Fig. 21, in contrast to Fig. 9, qubits are sorted from low order to high order.

B. DETAILED CONSTRUCTION OF OUR CONTROL MODULAR ADDER

In this section, we explain detail of our control modular adder. We show the example figures of our control modular adder too.

1) C-COMPARATOR

Now, we explain the construction of a C-comparator in more detail. In a C-comparator, we judge whether or not $b \ge d$, where b is a quantum value and d is a classical value. As noted in Section III. A, we conduct this by calculating the carry out of the entire circuit $b + (2^n - d)$. Our construction is given as follows.

1) *Initialization*: If we conduct Initialization naively, we apply a Toffoli gate and a CNOT gate for each bit. However, the compilation of a quantum algorithm often requires compilation (selection of the sequence of gates) to be adapted to the specific classical values that are inputs to the overall algorithm. Because $2^n - d$ is a classical value, we can convert some Toffoli gates to CNOT gates and eliminate other gates. Then, we calculate each $(2^n - d)_i$ $(0 \le i \le n-1)$. If $(2^n - d)_i = 1$,

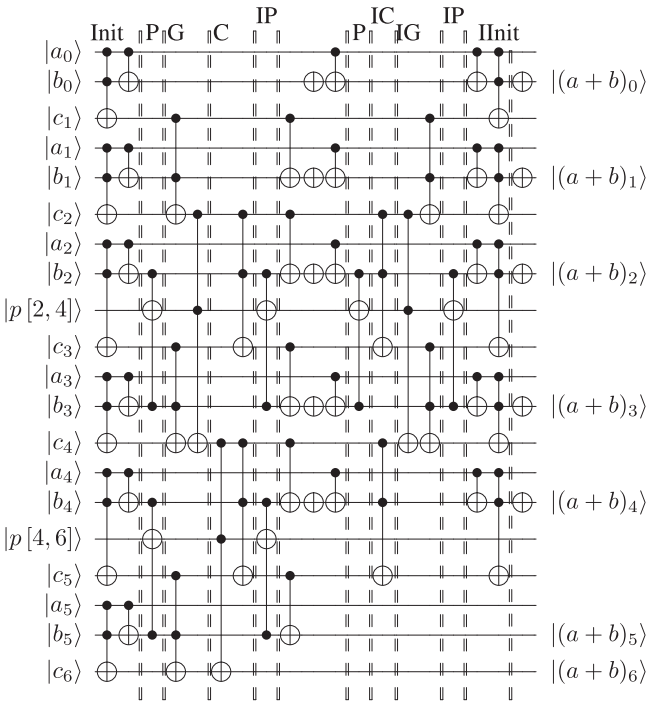

FIGURE 21. Example of Draper *et al.*'s carry-lookahead adder. This circuit adds two 6-bit numbers a and b, namely $|a\rangle|b\rangle \to |a\rangle|a+b\rangle$. In this figure, we sort qubits from the lowest qubits to the highest qubits. The labels at the top are the rounds including Toffoli gates. Init means Initialization. IP, IC, and IInit mean Inverse P-rounds, Inverse C-rounds, Inverse G-rounds, and Inverse Initialization respectively.

   a) we apply CNOT gates with the control qubit $|b_i\rangle$ and the target qubit $|c_{i+1}\rangle$;
   b) we apply X gates with on $|b_i\rangle$.

   These operations correspond to Toffoli gates or CNOT gates in the Initialization phase in Draper *et al.*'s construction, respectively.
2) *P-rounds and G-rounds:* We conduct P-rounds and G-rounds similar to Draper *et al.*'s construction.
3) *Writing result on the COMP qubit* ($O(1)$ gates and $O(1)$ depth): If we want to flip COMP when $b \geq d$, we apply Toffoli gates with the control qubits of CTRL and $|g[0,n]\rangle$, and with the target qubit of COMP. If we want to flip COMP when $b < d$, we apply Toffoli gates similarly to $b \geq d$, but we apply NOT gates on $|g[0,n]\rangle$ before and after the Toffoli gate.
4) *Resetting qubits*: We conduct inverse G-rounds and inverse P-rounds similar to Draper *et al.*'s construction. Moreover, we conduct the inverse of our Initialization. Then, we reset all qubits except COMP as the initial values.

2) CC-ADDER

First, we explain the construction of embedding in more detail. We want to embed as follows.

1) If CTRL is 1 and COMP is 1, we embed $2^n + a - N$.
2) If CTRL is 1 and COMP is 0, we embed a.
3) Otherwise, we embed no value.

Therefore, we embed on the second register on Fig. 10 as follows.

1) If CTRL is 1 and $(2^n + a - N)_i = a_i = 1$, the ith qubit is $|1\rangle$.
2) If CTRL is 1, COMP is 1, $(2^n + a - N)_i = 1$, and $a_i = 0$, the ith qubit is $|1\rangle$.
3) If CTRL is 1, COMP is 0, $(2^n + a - N)_i = 0$, and $a_i = 1$, the ith qubit is $|1\rangle$.
4) Otherwise, we do nothing.

In the aforementioned condition, the values of $(2^n + a - N)_i$ and a_i are classical information, and CTRL and COMP are quantum information. Thus, embedding in the first condition can be realized by CNOT gates with the control qubit of CTRL. Moreover, embedding in the second and third condition can be realized by Toffoli gates with the control qubits of CTRL and COMP. However, the set of i in each classical condition has no overlap. Therefore, once we embed one of i, we can embed the remaining value as CNOT gates. In each set, we have average $n/4$ elements requiring $n/4$ CNOT gates, $O(1)$ additional gates. Thus, these embedding can be implemented by $3n/4$ CNOT gates. Moreover, because we can run these simultaneously, embedding requires $\log n$ CNOT depth. The reset of embedding can be implemented similarly.

Next, we explain the optimization in an adder. In our calculation, there is no carry for $g[0,n]$ whether we subtract $N - a$ or add a. Thus, we can disregard calculation of carry qubit $g[0,n]$. To realize this, we omit calculation of $p[i,n]$ and $g[i,n]$ $(i < n)$. Moreover, by using classicality of a and N, we know that we embed no value in average $n/4$ qubits on the second register of Fig. 10. In these qubits, we can omit Initialization, inverse Initialization, and CNOT gates with the control qubit of $|a_i\rangle$ and the target qubit of $|b_i\rangle$ in erasing carry. By considering these optimizations, we reduce $n/2$ Toffoli gates and $3n/4$ CNOT gates.

The gate count and depth is shown in Table V.

3) EXAMPLE OF OUR CONTROL MODULAR ADDER

We show an example of a 6-bit control modular adder when $N = 59$ and $a = 37$. Circuits are given in Figs. 22–24.

In these example figures, registers are shown with low-order qubits at the top, in contrast to Fig. 10. In this section, the register $|b\rangle$ contains a quantum value.

The algorithm follows in the following order.

1) Conduct a C-comparator with the control qubit CTRL. Compare $|b\rangle$ and $N - a = 22$. If $b \geq 22$, flip COMP. This is implemented by adding $2^6 - (N - a) = 42$ and using the carry out.
2) Conduct a CC-adder. If both CTRL and COMP are 1, subtract $N - a = 22$. This is implemented by adding $2^6 - (N - a) = 42$ without calculating carry c_6. If CTRL is 1 and COMP is 0, add $a = 37$, otherwise, add no value.

IEEE Transactions on Quantum Engineering

TABLE V Gate Count and Depth of Our Proposed Control Modular Adder

		Count		Depth	
Operation	Rounds	Toffoli	CNOT	Toffoli	CNOT
C-comparator (twice)	Initialization	0	$0.5n$	0	$O(1)$
	P	n	0	$\}\log n$	0
	G	n	0		
	Inverse G	n	0	$\}\log n$	0
	Inverse P	n	0		
	Inverse Initialization	0	$0.5n$	0	$O(1)$
	Total	$4n$	n	$2\log n$	$O(1)$
CC-adder	Embedding	$O(1)$	$0.75n$	$O(1)$	$\log n$
	Initialization	$0.75n$	$0.75n$	$O(1)$	$O(1)$
	P	n	0	$\}\log n$	0
	G	n	0		
	C	n	0	$\}\log n$	0
	Inverse P	n	0		
	Calculating $\|a+b\rangle$	0	n	0	$O(1)$
	PE	0	$0.75n$	0	$O(1)$
	P	n	0	$\}\log n$	0
	Inverse C	n	0		
	Inverse G	n	0	$\}\log n$	0
	Inverse P	n	0		
	Inverse Initialization	$0.75n$	$0.75n$	$O(1)$	$O(1)$
	Resetting	$O(1)$	$0.75n$	$O(1)$	$\log n$
	Total	$9.5n$	$4.75n$	$4\log n$	$2\log n$
Total		$17.5n$	$6.75n$	$8\log n$	$2\log n$

We omit the rounds whose gate count is $O(1)$ and whose depth is $O(1)$.

3) Conduct a C-comparator with the control qubit CTRL. Compare $|b\rangle$ and $a = 37$. If $b < 37$, flip COMP. This is implemented by calculating carry of adding $2^6 - a = 27$.

These steps correspond to Figs. 22–24, respectively.

C. DETAILED CALCULATION OF T-DEPTH

In this section, we analyze the T-depth of our T-optimal control modular adder. We assume that we run GRT with the same timing, and each GRT has T-depth 2 from Fig. 6. We focus on the parts that can be run concurrently. Except for Initialization, we run the following:

1) P-rounds and G-rounds simultaneously;
2) C-rounds and inverse P-rounds simultaneously;
3) P-rounds and inverse C-rounds simultaneously;
4) inverse G-rounds and inverse P-rounds simultaneously.

In the first and third steps, we run many T gates simultaneously at the start and fewer T gates as the calculation progresses. In the second and fourth steps, we run only a few T gates simultaneously initially and more as the calculation progresses. Thus, there is a difference in the number of T gates we can run simultaneously.

As noted in Section IV-A, we define n_T as the upper bound of the number of T gates running simultaneously, and we calculate T-depth based on n_T as in Fig. 16. In each round, there are parts where we can run more than n_T T gates. However, by setting n_T, we run these T gates separately. Compared with this, in the parts having less than n_T T gates, we can run these T gates simultaneously.

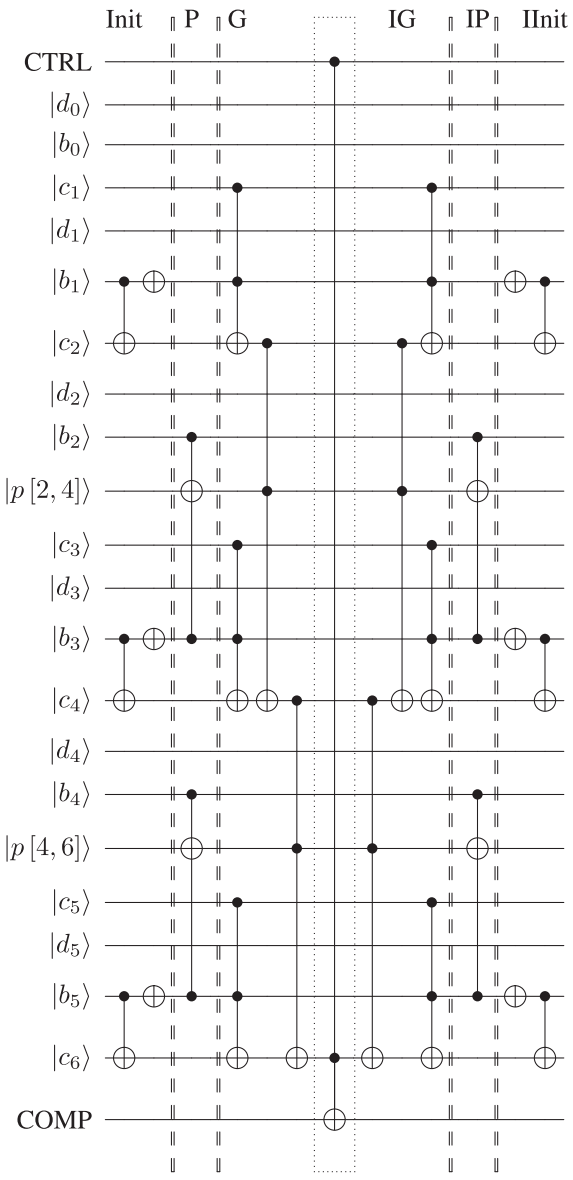

FIGURE 22. Example circuit of the first C-comparator for flipping the COMP qubit if $b \geq 22$. To achieve this, We add $2^6 - 22 = 42 = 101010_2$ and use the COMP qubit as the carry out of the adder. The Init phase consists of pairs of gates, a cnot, and an X, on the second, fourth, and sixth groups of qubits including $|d_i\rangle$, $|b_i\rangle$, and $|c_{i+1}\rangle$ from the lowest bit. This circuit is symmetric about the Toffoli gate surrounded by a dotted box. Init, IP, IG, and IInit mean Initialization, Inverse P-rounds, Inverse C-rounds, Inverse G-rounds, and Inverse Initialization, respectively.

First, we consider the parts having fewer than n_T T gates, which happens when we run P-rounds and G-rounds simultaneously, C-rounds and inverse P-rounds simultaneously, P-rounds and inverse C-rounds simultaneously, and inverse G-rounds and inverse P-rounds simultaneously. In these rounds, if we have no restriction on running T gates, patterns are given as follows.

1) In the first and the third cases, the number of T gates we can run simultaneously decreases by one half as the

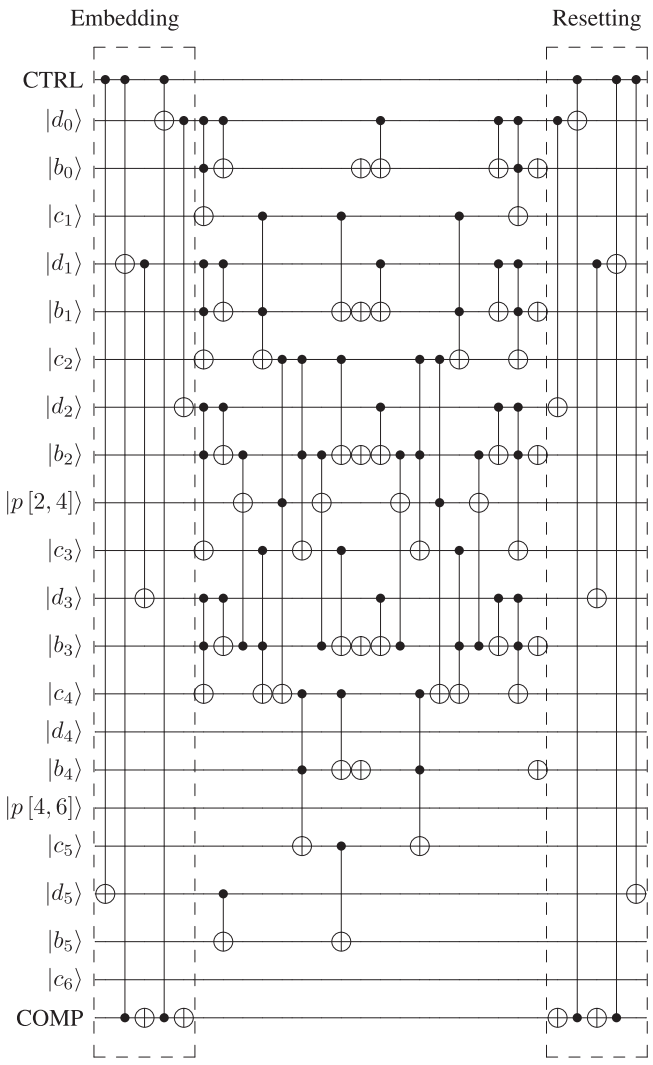

FIGURE 23. Example of the CC-adder. If both CTRL and COMP are 1, we subtract $N - a = 22$. This is implemented by adding $2^6 - (N - a) = 42 = 101010_2$ without calculating carry c_6. If CTRL is 1 and COMP is 0, we add $a = 37 = 100101_2$. Based on these, we conduct embedding and resetting. The remaining part is an adder, and we omit the calculation of $p[i, 6]$ and $g[i, 6]$ ($i < 6$).

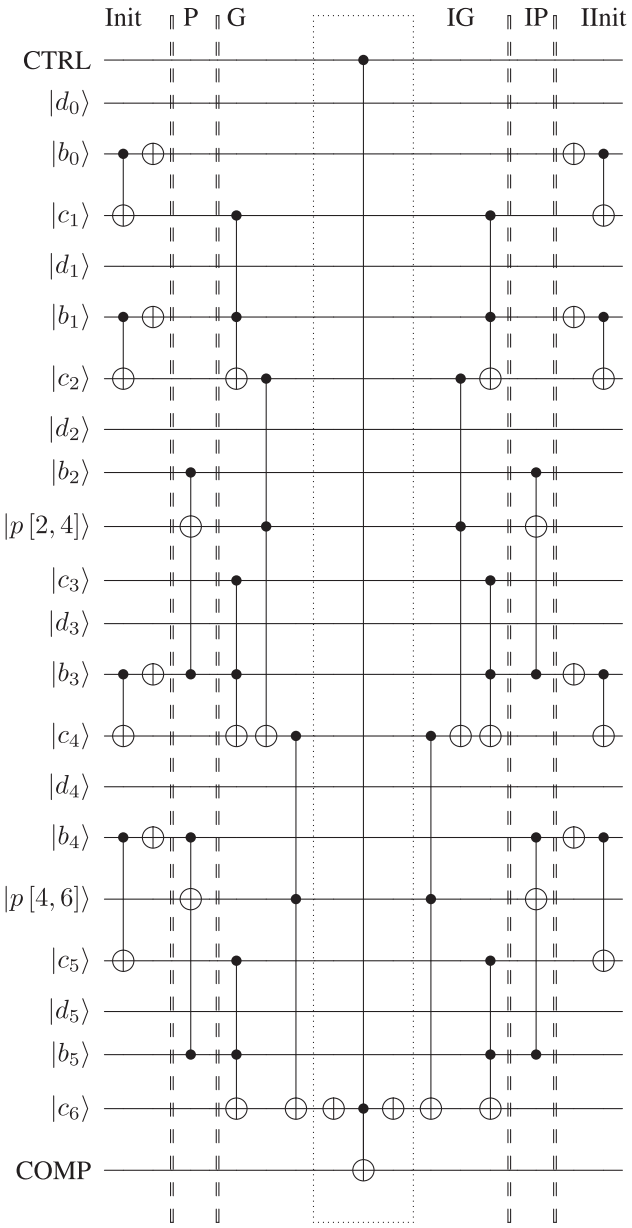

FIGURE 24. Example of the last C-comparator. We flip the COMP qubit if $b < 37$. This is achieved by adding $2^6 - 37 = 27 = 011011_2$ and using the carry out. First, we apply pairs of gates, a cnot and an X gate, on the first, second, fourth, and fifth groups of qubits. In contrast to Fig. 22, we apply X gates before and after the center Toffoli gate. This circuit is symmetric about the Toffoli gate surrounded by a dotted box. Init, IP, IG, and IInit means Initialization, Inverse P-rounds, Inverse C-rounds, Inverse G-rounds, and Inverse Initialization, respectively.

calculation progresses. Thus, in the latter part of the calculation, we run fewer than n_T T gates simultaneously. This part has T-depth $2 \log n_T$ and n_T T gates in total.

2) In the second and the fourth cases, the number of T gates we can run simultaneously doubles as the calculation progresses. Thus, in the former part of the calculation, we run less than n_T T gates simultaneously. This part has T-depth $2 \log n_T$ and n_T T gates in total.

We have six parts each with a small number of T gates, as follows:

1) P-rounds and G-rounds in the first C-comparator;
2) P-rounds and G-rounds in the CC-adder;
3) C-rounds and inverse P-rounds in the CC-adder;
4) P-rounds and inverse C-rounds in the CC-adder;
5) inverse G-rounds and inverse P-rounds in the CC-adder;
6) P-rounds and G-rounds in the final C-comparator.

Thus, we consume $12 \log n_T$ T-depth and $6n_T$ T gates in these.

TABLE VI Breakdown of Toffoli Count and *T*-Count of Our Control Modular Adder

			Toffoli Decomposition											
			Draper *et al.* [43]			Thapliyal *et al.* [48] (qubit-optimize)			Thapliyal *et al.* [48] (T-optimize)			**Ours**		
Operation	Rounds	Tof	gate	cost	count	gate	cost	count	gate	cost	count	gate	cost	count
C-comp (twice)	P	n	ST	7	$7n$	GRT	4	$4n$	GRT	4	$4n$	GRT	4	$4n$
	G	n	ST	7	$7n$	ST	7	$7n$	PGRT	4	$4n$	GRT	4	$4n$
	InvG	n	ST	7	$7n$	ST	7	$7n$	PGRT	4	$4n$	IGRT	0	0
	InvP	n	ST	7	$7n$	IGRT	0	0	IGRT	0	0	IGRT	0	0
	Total	$4n$	–	–	$28n$	–	–	$18n$	–	–	$12n$	–	–	$8n$
CC-add	Init	$0.75n$	ST	7	$5.25n$	GRT	4	$3n$	GRT	4	$3n$	GRT	4	$3n$
	P	n	ST	7	$7n$	GRT	4	$4n$	GRT	4	$4n$	GRT	4	$4n$
	G	n	ST	7	$7n$	ST	7	$7n$	PGRT	4	$4n$	PGRT	4	$4n$
	C	n	ST	7	$7n$	ST	7	$7n$	PGRT	4	$4n$	PGRT	4	$4n$
	InvP	n	ST	7	$7n$	IGRT	0	0	IGRT	0	0	IGRT	0	0
	P	n	ST	7	$7n$	GRT	4	$4n$	GRT	4	$4n$	GRT	4	$4n$
	InvC	n	ST	7	$7n$	ST	7	$7n$	PGRT	4	$4n$	PGRT	4	$4n$
	InvG	n	ST	7	$7n$	ST	7	$7n$	PGRT	4	$4n$	PGRT	4	$4n$
	InvP	n	ST	7	$7n$	IGRT	0	0	IGRT	0	0	IGRT	0	0
	InvInit	$0.75n$	ST	7	$5.25n$	IGRT	0	0	IGRT	0	0	IGRT	0	0
	Total	$9.5n$	–	–	$66.5n$	–	–	$39n$	–	–	$27n$	–	–	$27n$
Total		$17.5n$	–	–	$122.5n$	–	–	$75n$	–	–	$51n$	–	–	$43n$

Tof means the number of Toffoli gates in each round. Gate means the type of using relative-phase Toffoli gates in each round. Cost means the number of T gates in each relative-phase Toffoli Gate. Count means T-count in each round. We omit the rounds whose T-count is $O(1)$. Inv means inverse, C-Comp means a C-comparator, CC-Add means a CC-adder, and Init means Initialization.

TABLE VII Breakdown of Toffoli Count and CNOT Count of Our Control Modular Adder

				Toffoli Decomposition											
				Draper *et al.* [43]			Thapliyal *et al.* [48] (qubit-optimize)			Thapliyal *et al.* [48] (T-optimize)			**Ours**		
Operation	Rounds			gate	cost	count	gate	cost	count	gate	cost	count	gate	cost	count
C-Comp (twice)	Init	CNOT	$0.5n$	–	–	$0.5n$	–	–	$0.5n$	–	–	$0.5n$	–	–	$0.5n$
	P	Toffoli	n	ST	6	$6n$	GRT	6	$6n$	GRT	6	$6n$	RT3	3	$3n$
	G	Toffoli	n	ST	6	$6n$	ST	6	$6n$	PGRT	8	$8n$	RT3	3	$3n$
	InvG	Toffoli	n	ST	6	$6n$	ST	6	$6n$	PGRT	8	$8n$	IRT3	3	$3n$
	InvP	Toffoli	n	ST	6	$6n$	IGRT	1	n	IGRT	1	n	IRT3	3	$3n$
	InvInit	CNOT	$0.5n$	–	–	$0.5n$	–	–	$0.5n$	–	–	$0.5n$	–	–	$0.5n$
	Total			–	–	$25n$	–	–	$20n$	–	–	$24n$	–	–	$13n$
CC-add	Embed	CNOT	$0.75n$	–	–	$0.75n$	–	–	$0.75n$	–	–	$0.75n$	–	–	$0.75n$
	Init	Toffoli	$0.75n$	ST	6	$4.5n$	GRT	6	$4.5n$	GRT	6	$4.5n$	RT4	4	$3n$
		CNOT	$0.75n$	–	–	$0.75n$	–	–	$0.75n$	–	–	$0.75n$	–	–	$0.75n$
	P	Toffoli	n	ST	6	$6n$	GRT	6	$6n$	GRT	6	$6n$	RT3	3	$3n$
	G	Toffoli	n	ST	6	$6n$	ST	6	$6n$	PGRT	8	$8n$	RT4	4	$4n$
	C	Toffoli	n	ST	6	$6n$	ST	6	$6n$	PGRT	8	$8n$	RT4	4	$4n$
	InvP	Toffoli	n	ST	6	$6n$	IGRT	1	n	IGRT	1	n	IRT3	3	$3n$
	Calc	CNOT	n	–	–	n	–	–	n	–	–	n	–	–	n
	PE	CNOT	$0.75n$	–	–	$0.75n$	–	–	$0.75n$	–	–	$0.75n$	–	–	$0.75n$
	P	Toffoli	n	ST	6	$6n$	GRT	6	$6n$	GRT	6	$6n$	RT3	3	$3n$
	InvC	Toffoli	n	ST	6	$6n$	ST	6	$6n$	PGRT	8	$8n$	IRT4	4	$4n$
	InvG	Toffoli	n	ST	6	$6n$	ST	6	$6n$	PGRT	8	$8n$	IRT4	4	$4n$
	InvP	Toffoli	n	ST	6	$6n$	IGRT	1	n	IGRT	1	n	IRT3	3	$3n$
	InvInit	Toffoli	$0.75n$	ST	6	$4.5n$	IGRT	1	$0.75n$	IGRT	1	$0.75n$	RT4	4	$3n$
		CNOT	$0.75n$	–	–	$0.75n$	–	–	$0.75n$	–	–	$0.75n$	–	–	$0.75n$
	Reset	CNOT	$0.75n$	–	–	$0.75n$	–	–	$0.75n$	–	–	$0.75n$	–	–	$0.75n$
	Total			–	–	$61.75n$	–	–	$48n$	–	–	$56n$	–	–	$38.75n$
Total				–	–	$111.75n$	–	–	$88n$	–	–	$104n$	–	–	$64.75n$

Gate means the type of using relative-phase Toffoli gates in each round. Cost means the number of CNOT gates in each relative-phase Toffoli gate. Count means CNOT count in each round. We do not show the rounds whose CNOT count is $O(1)$. Inv means Inverse, C-Comp means a C-Comparator, CC-Add means a CC-Adder, Init means Initialization, Embed means Embedding, Calc means Calculating of $|a+b\rangle$, and Reset means resetting.

Next, we consider the remaining parts. In these parts, we run T gates n_T each. The number of total T gates is $43n$ from Table II, and we run $43n - 6n_T$ T gates. Thus, T-depth of this part is given by

$$\frac{2\,(43n - 6n_T)}{n_T} = \frac{86n}{n_T} - 12. \tag{25}$$

In conclusion, T-depth is given by

$$\frac{86n}{n_T} + 12\log n_T - 12. \tag{26}$$

D. DETAILED GATE COUNT ON FTQ OR NISQ

In this section, we detail the T gate count on FTQ or NISQ. The FTQ count is shown in Table VI. The detailed CNOT gate count on NISQ is shown in Table VII. The detailed CNOT depth count and KQ_{CX} on NISQ are shown in Table VIII.

E. DISTILLATION CIRCUIT FOR A T GATE

A distillation circuit for a T gate is given as Fig. 25.

TABLE VIII Breakdown of Toffoli Count and CNOT-Depth of Our Control Modular Adder

				Toffoli Decomposition											
				Draper *et al.* [43]			Thapliyal *et al.* [48] (qubit-optimize)			Thapliyal *et al.* [48] (T-optimize)			**Ours**		
Operation	Rounds			gate	cost	depth	gate	cost	depth	gate	cost	depth	gate	cost	depth
C-Comp (twice)	P	Toffoli	$\}\log n$	ST	6	$\}6\log n$	GRT	6	$\}6\log n$	GRT	6	$\}8\log n$	RT3	3	$\}3\log n$
	G	Toffoli		ST	6		ST	6		PGRT	8		RT3	3	
	InvG	Toffoli	$\}\log n$	ST	6	$\}6\log n$	ST	6	$\}6\log n$	PGRT	8	$\}8\log n$	IRT3	3	$\}3\log n$
	InvP	Toffoli		ST	6		IGRT	1		IGRT	1		IRT3	3	
	Total			–	–	$12\log n$	–	–	$12\log n$	–	–	$16\log n$	–	–	$6\log n$
CC-add	Embed	CNOT	$\log n$	–	–	$\log n$	–	–	$\log n$	–	–	$\log n$	–	–	$\log n$
	P	Toffoli	$\}\log n$	ST	6	$\}6\log n$	GRT	6	$\}6\log n$	GRT	6	$\}8\log n$	RT3	3	$\}4\log n$
	G	Toffoli		ST	6		ST	6		PGRT	8		RT4	4	
	C	Toffoli	$\}\log n$	ST	6	$\}6\log n$	ST	6	$\}6\log n$	PGRT	8	$\}8\log n$	RT4	4	$\}4\log n$
	InvP	Toffoli		ST	6		IGRT	1		IGRT	1		IRT3	3	
	P	Toffoli	$\}\log n$	ST	6	$\}6\log n$	GRT	6	$\}6\log n$	GRT	6	$\}8\log n$	RT3	3	$\}4\log n$
	InvC	Toffoli		ST	6		ST	6		PGRT	8		IRT4	4	
	InvG	Toffoli	$\}\log n$	ST	6	$\}6\log n$	ST	6	$\}6\log n$	PGRT	8	$\}8\log n$	IRT4	4	$\}4\log n$
	InvP	Toffoli		ST	6		IGRT	1		IGRT	1		IRT3	3	
	Reset	CNOT	$\log n$	–	–	$\log n$	–	–	$\log n$	–	–	$\log n$	–	–	$\log n$
	Total			–	–	$26\log n$	–	–	$26\log n$	–	–	$34\log n$	–	–	$18\log n$
Total				–	–	$50\log n$	–	–	$50\log n$	–	–	$66\log n$	–	–	$30\log n$
#qubits				–	–	$4n$	–	–	$4n$	–	–	$4.5n$	–	–	$4n$
KQ_{CX}				–	–	$200n\log n$	–	–	$200n\log n$	–	–	$297n\log n$	–	–	$120n\log n$

Gate means the type of using relative-phase Toffoli gates in each round. Cost means the number of CNOT gates in each relative-phase Toffoli gate. Depth means CNOT-depth in each round. We omit the rounds whose CNOT-depth is $O(1)$. Inv means Inverse, C-Comp means a C-Comparator, CC-Add means a CC-Adder, Init means Initialization, Embed means Embedding, and Reset means resetting.

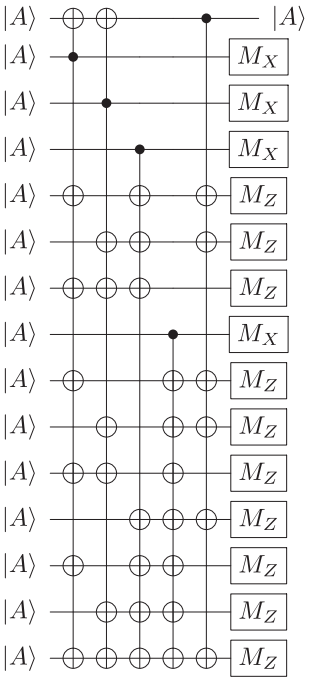

FIGURE 25. Distillation circuit of $|A\rangle$ [23]. By this distillation circuit, we reduce the error rate of $|A\rangle$ from p to $35p^3$.

ACKNOWLEDGMENT

This article is written based on Chapter 6 of the dissertation by Oonishi [58].

REFERENCES

[1] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018, doi: 10.22331/q-2018-08-06-79.

[2] A. D. Córcoles *et al.*, "Challenges and opportunities of near-term quantum computing systems," 2019, *arXiv:1910.02894*.

[3] IBM Quantum Experience. Accessed: Mar. 31, 2021. [Online]. Available: https://quantum-computing.ibm.com

[4] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019, doi: 10.1038/s41586-019-1666-5.

[5] R. S. Smith, M. J. Curtis, and W. J. Zeng, "A practical quantum instruction set architecture," 2016, *arXiv:1608.03355*.

[6] P. Murali, D. M. Debroy, K. R. Brown, and M. Martonosi, "Architecting noisy intermediate-scale trapped ion quantum computers," 2020, *arXiv:2004.04706*.

[7] S. Moses *et al.*, "Demonstration of the QCCD trapped-ion quantum computer architecture," *Bull. Amer. Phys. Soc.*, vol. 65, no. 4, pp. Q01–163,2020.

[8] L. M. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, vol. 414, no. 6866, pp. 883–887, 2001, doi: 10.1038/414883a.

[9] A. Politi, J. C. Matthews, and J. L. O'brien, "Shor's quantum factoring algorithm on a photonic chip," *Science*, vol. 325, no. 5945, pp. 1221–1221, 2009, doi: 10.1126/science.1173731.

[10] E. Lucero *et al.*, "Computing prime factors with a Josephson phase qubit quantum processor," *Nat. Phys.*, vol. 8, no. 10, pp. 719–723, 2012, doi: 10.1038/nphys2385.

[11] E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'brien, "Experimental realization of Shor's quantum factoring algorithm using qubit recycling," *Nat. Photon.*, vol. 6, no. 11, pp. 773–776, 2012, doi: 10.1038/nphoton.2012.259.

[12] T. Monz *et al.*, "Realization of a scalable Shor algorithm," *Science*, vol. 351, no. 6277, pp. 1068–1070, 2016, doi: 10.1126/science.aad9480.

[13] C. Lu, D. Browne, T. Yang, and J. Pan, "Demonstration of a compiled version of Shor's quantum factoring algorithm using photonic qubits," *Phys. Rev. Lett.*, vol. 99, no. 25, 2007, Art. no. 250504, doi: 10.1103/PhysRevLett.99.250504.

[14] B. Lanyon *et al.*, "Experimental demonstration of a compiled version of Shor's algorithm with quantum entanglement," *Phys. Rev. Lett.*, vol. 99, no. 25, 2007, Art. no. 250505, doi: 10.1103/PhysRevLett.99.250505.

[15] J. A. Smolin, G. Smith, and A. Vargo, "Oversimplifying quantum factoring," *Nature*, vol. 499, no. 7457, pp. 163–165, 2013, doi: 10.1038/nature12290.

[16] M. Briański, J. Gwinner, V. Hlembotskyi, W. Jarnicki, S. Pliś, and A. Szady, "Introducing structure to expedite quantum search," 2020, *arXiv:2006.05828*.

[17] T. Satoh, Y. Ohkura, and R. Van Meter, "Subdivided phase oracle for NISQ search algorithms," *IEEE Trans. Quantum Eng.*, vol. 1, 2020, Art. no. 3100815, doi: 10.1109/TQE.2020.3012068.

[18] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Rep. Prog. Phys.*, vol. 76, no. 7, 2013, Art. no. 076001, doi: 10.1088/0034-4885/76/7/076001.

[19] J. Preskill, "Fault-tolerant quantum computation," in *Introduction to Quantum Computation and Information*. Singapore: World Scientific, 1998, pp. 213–269, doi: 10.1142/9789812385253_0008.

[20] A. M. Steane, "Space, time, parallelism and noise requirements for reliable quantum computing," *Fortschritte Phys.*, vol. 46, no. 4-5, pp. 443–457, 1998, doi: 10.1002/(SICI)1521-3978(199806)46:4/5<443::AID-PROP443>3.0.CO;2-8.

[21] N. C. Jones *et al.*, "Layered architecture for quantum computing," *Phys. Rev. X*, vol. 2, no. 3, 2012, Art. no. 031007, doi: 10.1103/PhysRevX.2.031007.

[22] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," *Amer. J. Phys.*, 70, p. 558, 2002, doi: 10.1119/1.146374.

[23] A. G. Fowler, A. M. Stephens, and P. Groszkowski, "High-threshold universal quantum computation on the surface code," *Phys. Rev. A*, vol. 80, no. 5, 2009, Art. no. 052312, doi: 10.1103/PhysRevA.80.052312.

[24] C. Gidney and A. G. Fowler, "Efficient magic state factories with a catalyzed $|CCZ\rangle \rightarrow 2|T\rangle$ transformation," *Quantum*, vol. 3, p. 135, 2019, doi: 10.22331/q-2019-04-30-135.

[25] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, 1999, doi: 10.1137/S0097539795293172.

[26] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput.*, 1996, pp. 212–219, doi: 10.1145/237814.237866.

[27] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978, doi: 10.1145/359340.359342.

[28] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, 1987, doi: 10.1090/S0025-5718-1987-0866109-5.

[29] V. S. Miller, "Use of elliptic curves in cryptography," in *Proc. Conf. Theory Appl. Cryptographic Techn.*, 1985, pp. 417–426, doi: 10.1007/3-540-39799-X_31.

[30] D. Beckman, A. N. Chari, S. Devabhaktuni, and J. Preskill, "Efficient networks for quantum factoring," *Phys. Rev. A*, vol. 54, no. 2, pp. 1034–1063, 1996, doi: 10.1103/PhysRevA.54.1034.

[31] J. Davies, C. J. Rickerd, M. A. Grimes, and D. Ö. Güney, "An n-bit general implementation of Shor's quantum factoring algorithm," *Quantum Inf. Comput.*, vol. 16, no. 7-8, pp. 700–718, 2016, doi: 10.26421/QIC16.7-8-6.

[32] A. G. Fowler, S. J. Devitt, and L. C. Hollenberg, "Implementation of Shor's algorithm on a linear nearest neighbor qubit array," *Quantum Inf. Comput.*, vol. 4, no. 4, p. 237, 2004, doi: 10.5555/2011827.2011828.

[33] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," 2019, *arXiv:1905.09749*.

[34] T. Häner, S. Jaques, M. Naehrig, M. Roetteler, and M. Soeken, "Improved quantum circuits for elliptic curve discrete logarithms," in *Proc. Int. Conf. Post-Quantum Cryptogr.*, 2020, pp. 425–444, doi: 10.1007/978-3-030-44223-1_23.

[35] I. L. Markov and M. Saeedi, "Constant-optimized quantum circuits for modular multiplication and exponentiation," *Quantum Inf. Comput.*, vol. 12, no. 5-6, pp. 361–394, 2012, doi: 10.5555/2230996.2230997.

[36] A. Pavlidis and D. Gizopoulos, "Fast quantum modular exponentiation architecture for Shor's factoring algorithm," *Quantum Inf. Comput.*, vol. 14, no. 7&8, pp. 649–682, 2014, doi: 10.5555/2638682.2638690.

[37] M. Roetteler, M. Naehrig, K. M. Svore, and K. Lauter, "Quantum resource estimates for computing elliptic curve discrete logarithms," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2017, pp. 241–270, doi: 10.1007/978-3-319-70697-9_9.

[38] R. Van Meter and K. M. Itoh, "Fast quantum modular exponentiation," *Phys. Rev. A*, vol. 71, no. 5, 2005, Art. no. 052320, doi: 10.1103/PhysRevA.71.052320.

[39] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Phys. Rev. A*, vol. 54, no. 1, p. 147, 1996, doi: 10.1103/PhysRevA.54.147.

[40] C. Zalka, "Fast versions of Shor's quantum factoring algorithm," 1998, *arXiv:quant-ph/9806084*.

[41] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," 2004, *arXiv:quant-ph/0410184*.

[42] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "A logarithmic-depth quantum carry-lookahead adder," *Quantum Inf. Comput.*, vol. 6, no. 4, pp. 351–369, 2006, doi: 10.5555/2012086.2012090.

[43] T. G. Draper, "Addition on a quantum computer," 2000, *arXiv:quant-ph/0008033*.

[44] C. Gidney, "Halving the cost of quantum addition," *Quantum*, vol. 2, p. 74, 2018, doi: 10.22331/q-2018-06-18-74.

[45] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," *Quantum Inf. Comput.*, vol. 10, no. 9, pp. 872–890, 2010, doi: 10.5555/2011464.2011476.

[46] A. M. Steane, "Overhead and noise threshold of fault-tolerant quantum error correction," *Phys. Rev. A*, vol. 68, no. 4, 2003, Art. no. 042322, doi: 10.1103/PhysRevA.68.042322.

[47] H. Thapliyal, E. Muñoz-Coreas, and V. Khalus, "T-count and qubit optimized quantum circuit designs of carry lookahead adder," 2020, *arXiv:2004.01826*.

[48] D. Maslov, "Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization," *Phys. Rev. A*, vol. 93, no. 2, 2016, Art. no. 022311, doi: 10.1103/PhysRevA.93.022311.

[49] B.-S. Choi and R. Van Meter, "A θ ($\sqrt{n}$)-depth quantum adder on the 2D NTC quantum computer architecture," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 8, no. 3, pp. 1–22, 2012, doi: 10.1145/2287696.2287707.

[50] Y. Hirata, M. Nakanishi, S. Yamashita, and Y. Nakashima, "An efficient conversion of quantum circuits to a linear nearest neighbor architecture," *Quantum Inf. Comput.*, vol. 11, no. 1 and 2, pp. 142–166, 2011, doi: 10.5555/2011383.2011393.

[51] R. Van Meter, T. D. Ladd, A. G. Fowler, and Y. Yamamoto, "Distributed quantum computation architecture using semiconductor nanophotonics," *Int. J. Quantum Inf.*, vol. 8, pp. 295–323, 2010, doi: 10.1142/S0219749910006435.

[52] I. H. Kim, E. Lee, Y.-H. Liu, S. Pallister, W. Pol, and S. Roberts, "Fault-tolerant resource estimate for quantum chemical simulations: Case study on Li-ion battery electrolyte molecules," 2021, *arXiv:2104.10653*.

[53] A. Paler and R. Basmadjian, "Clifford gate optimisation and T gate scheduling: Using queueing models for topological assemblies," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit.*, 2019, pp. 1–5, doi: 10.1109/NANOARCH47378.2019.181305.

[54] N. Isailovic, M. Whitney, Y. Patel, and J. Kubiatowicz, "Running a quantum circuit at the speed of data," *ACM SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 177–188, 2008, doi: 10.1145/1394608.1382137.

[55] T. Æ. Mogensen, "Reversible in-place carry-lookahead addition with few ancillae," in *Proc. Int. Conf. Reversible Comput.*, 2019, pp. 224–237, doi: 10.1007/978-3-030-21500-2_14.

[56] H. Thapliyal, H. Jayashree, A. Nagamani, and H. R. Arabnia, "Progress in reversible processor design: A novel methodology for reversible carry look-ahead adder," in *Transactions on Computational Science XVII*. Berlin, Germany: Springer, 2013, pp. 73–97, doi: 10.1007/978-3-642-35840-1_4.

[57] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, "Quantum circuit simplification and level compaction," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 27, no. 3, pp. 436–444, Mar. 2008, doi: 10.1109/TCAD.2007.911334.

[58] K. Oonishi, "Security evaluation of public-key cryptography against side-channel attacks and quantum attacks," Ph.D. dissertation, Math. Informatics 1st Lab., Univ. Tokyo, Tokyo, Japan, 2021.

Kento Oonishi received the B.S. degree in engineering, the M.S. degree in information science and technology, and the Ph.D. degree in mathematical informatics from the University of Tokyo, Tokyo, Japan, in 2016, 2018, and 2021, respectively. His major is security evaluation of public-key cryptography against side-channel and quantum attacks.

During his Ph.D., he studied quantum computation with Keio University. He is currently a Researcher with Mitsubishi Electric, Tokyo. His research interest includes cryptography, quantum computation, and artificial intelligence.

Tomoki Tanaka received the B.S. degree in science and the M.S. degree in mathematical sciences from Nagoya University, Nagoya, Japan, in 2009 and 2011, respectively. His major is topology, especially Knot theory.

In 2011, he joined Mitsubishi UFJ Financial Group, Inc. (MUFG), Tokyo, Japan, and from 2018, MUFG participated in IBM Quantum Network Hub, Keio University and he joined this project as a Project Researcher with Keio Quantum Computing Center. He is currently a Vice President with MUFG. His research interests include quantum computing for using financial applications, such as derivatives simulation, risk management, optimization, and machine learning.

Shumpei Uno received the M.Sc. and Ph.D. degrees in particle physics from Nagoya University, Nagoya, Japan, in 2008 and 2011, respectively.

During the Ph.D., he formulated quantum electrodynamics on finite volume lattice in order to accurately predict the light quark masses. He joined Mizuho Research & Technologies, Ltd. (MHRT), Tokyo, Japan, in 2011, and become a Project Researcher with Keio Quantum Computing Center in 2018. He is currently a Chief Consultant with MHRT. His research interests include quantum computing for using financial applications, such as derivatives simulation, risk management, optimization, and machine learning.

Takahiko Satoh studied at Keio University, Yokohama, Japan, and the University of Tokyo (UT), Tokyo, Japan. He received the bachelor's degree in environmental information from Keio. He received the master's and the Ph.D. degree (2016) in computer science from UT.

He is currently a Project Assistant Professor of Keio University Quantum Computing Center, Yokohama, Japan. His research field is quantum computing and quantum networking, particularly quantum network coding, NISQ algorithm design, and quantum Internet security.

Dr. Satoh is a member of the Physical Society of Japan (JPS).

Rodney Van Meter (Senior Member, IEEE) received the Ph.D. degree in computer science from Keio University, Yokohama, Japan, in 2006.

He is currently a Professor of environment and information studies with Keio University's Shonan Fujisawa Campus, Fujisawa, Japan. He is Vice Center Chair with the Keio Quantum Computing Center, a Board Member of the WIDE Project, and a Member of the Quantum Internet Task Force. His research interests include quantum networking, quantum computing, storage systems, networking, and post-Moore's law computer architecture.

Dr. Van Meter is Member of the Association for Computing Machinery, the American Physical Society, and the American Association for the Advancement of Science.

Noboru Kunihiro received the B.E., M.E., and Ph.D. degrees in mathematical engineering and information physics from the University of Tokyo, Tokyo, Japan, in 1994, 1996, and 2001, respectively.

He has been a Professor with the University of Tsukuba, Tsukuba, Japan, since 2019. He was a Researcher with NTT Communication Science Laboratories from 1996 to 2002. He was an Associate Professor with the University of Electro-Communications from 2002 to 2008. He was an Associate Professor with the University of Tokyo from 2008 to 2019. His research interests include cryptography, information security, and quantum computation.