Received 28 June 2023; revised 30 August 2023; accepted 14 September 2023; date of publication 25 September 2023; date of current version 25 October 2023.

Digital Object Identifier 10.1109/TQE.2023.3319044

Shor's Algorithm Using Efficient Approximate Quantum Fourier Transform

KENTO OONISHI[1] AND NOBORU KUNIHIRO[2] (Member, IEEE)

[1]Mitsubishi Electric Corporation, Kamakura 247-8501, Japan
[2]University of Tsukuba, Tsukuba 305-8573, Japan

Corresponding author: Kento Oonishi (e-mail: onishi.kento@ap.mitsubishielectric.co.jp).

This work was supported in part by Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP21H03440 and in part by Japan Science and Technology CREST under Grant JPMJCR2113.

ABSTRACT Shor's algorithm solves the integer factoring and discrete logarithm problems in polynomial time. Therefore, the evaluation of Shor's algorithm is essential for evaluating the security of currently used public-key cryptosystems because the integer factoring and discrete logarithm problems are crucial for the security of these cryptosystems. In this article, a new approximate quantum Fourier transform is proposed, and it is applied to Rines and Chuang's implementation. The proposed implementation requires one-third the number of T gates of the original. Moreover, it requires one-fourth of the T-depth of the original. Finally, a T-scheduling method for running the circuit with the smallest KQ (where K is the number of logical qubits and Q is the circuit depth) is presented.

INDEX TERMS Approximate quantum Fourier transform, controlled modular multiplication, Shor's algorithm.

I. INTRODUCTION

A. BACKGROUND

Evaluation of Shor's algorithm [1] is extremely important. Shor's algorithm is a method for solving the integer factorization and discrete logarithm problems, which take subexponential time in classical computers [2]. These problems are fundamental problems for the security of the current public-key cryptosystems, including the RSA cryptosystems [3] and elliptic curve cryptosystems [4], [5]. Currently, the scale of quantum computers is considerably small for breaking these two public-key cryptosystems [6], [7], [8], [9], [10], [11]. However, the scale of quantum computers is increasing [12], and it is important to estimate the time when Shor's algorithm breaks these two public-key cryptosystems. To estimate the time when Shor's algorithm breaks the current public-key cryptosystems, precise evaluation of Shor's algorithm is important. This article discusses Shor's algorithm on a single quantum computer. If there are more than two computers, recently proposed distributed Shor's algorithm [13] will reduce computational costs. Our results will be able to combine with this result, and a single quantum computer is considered in this article. This article focuses on Shor's algorithm factoring an n-bit composite number N.

Previous researchers have evaluated the computational cost of Shor's algorithm by constructing quantum circuits for it. Shor's algorithm consists of modular exponentiation and a quantum Fourier transform (QFT). The modular exponentiation has a higher computational cost than the QFT. Previous research works [14], [15], [16], [17], [18] have decomposed a modular exponentiation into smaller arithmetic circuits, adders, and multipliers. These methods have different approaches for addressing carry, which is an overflow at each digit. The most widely used methods are the following.

1) *Ripple-Carry [14], [17]:* Carry is calculated sequentially from the least significant bit (LSB) to the most significant bit (MSB).
2) *Carry-Lookahead [16]:* Carry is calculated first before the actual calculation.
3) *Fourier-Basis [15], [18]:* All carries in each digit are calculated by QFT.

The first (ripple-carry) and the second (carry-lookahead) methods calculate each carry only on the corresponding digit. Unlike these two methods, the third method (Fourier basis) calculates a carry on all digits in the Fourier basis. Table 1 shows the computational cost of each method. Without

TABLE 1. Computational Cost for Factorization an n-Bit Number

Adder		Ripple-Carry [14]	Carry-Lookahead [16]	Fourier-Basis [18]
#(Qubits)		$3n$	$5n$	$2n$
Without fault-tolerance (Same Cost)	#(Gates)	$O\left(n^3\right)$	$O\left(n^3\right)$	$O\left(n^3\right)$
	Depth	$O\left(n^3\right)$	$O\left(n^2 \log n\right)$	$O\left(n^2\right)$
With fault-tolerance (Clifford+T)	T-count	$O\left(n^3\right)$	$O\left(n^3\right)$	$\boldsymbol{O\left(n^3 \log n\right)}$
	T-depth	$O\left(n^3\right)$	$O\left(n^2 \log n\right)$	$\boldsymbol{O\left(n^2 \log n\right)}$

Boldface entities show the target of this article.

fault tolerance, the Fourier-basis circuit is better than other constructions. This difference occurs because the Fourier-basis method calculates carry with higher parallelization than the other methods.

To consider the actual computational cost for the future, we must consider fault tolerance. Several researchers [17], [18] have considered fault tolerance in Shor's algorithm. In a fault-tolerant setting, non-Clifford gates, including T gates, have a higher cost than Clifford gates. Usually, we realize non-Clifford gates by magic state distillation [19], resulting in a heavy cost. Therefore, previous researchers have represented a quantum circuit with Clifford+T gates and minimized the computational cost of T, including the number of T gates (called "T-count" in this article) and depth of T gates (called "T-depth" in this article).

This article focuses on Shor's algorithm using the Fourier-basis method. As noted previously, the Fourier-basis method has a lower computational cost than other methods without fault tolerance. However, the computational cost of the Fourier-basis method increases with fault tolerance, as shown in Table 1. In more detail, the T-count and T-depth are $O(\log n)$ times with fault tolerance. This increase occurs because the Fourier-basis method requires phase gates, which require $O(\log n)$ Clifford+T gates [20].

However, the Fourier-basis method proposed by Rines and Chuang [18] needs improvement. Previous research has shown that approximate QFTs, which omit phase gates with a small phase, remain a desired result in Shor's algorithm [21], [22]. Rines and Chuang [18] method adopts QFTs but does not consider approximate QFTs. Therefore, we should consider the computational cost of adopting approximate QFTs for evaluating the Fourier-basis method more precisely.

B. OUR CONTRIBUTIONS

We evaluate Shor's algorithm using the Fourier-basis method [18] more precisely. Especially, the controlled modular multiplier using Montgomery reduction is improved by reducing T-depth. The contributions consist of the following.

1) Montgomery multiplication [18] is improved (see Section III).
2) The proposed Montgomery multiplication is applied to Shor's algorithm (see Section IV).

First, the improvement of Rines and Chuang [18] implementation of the Montgomery multiplication is described in Section III. Their quantum circuit is improved where the dominant term is $2n$, namely, almost the same number of qubits as in Rines and Chuang [18] construction. The method

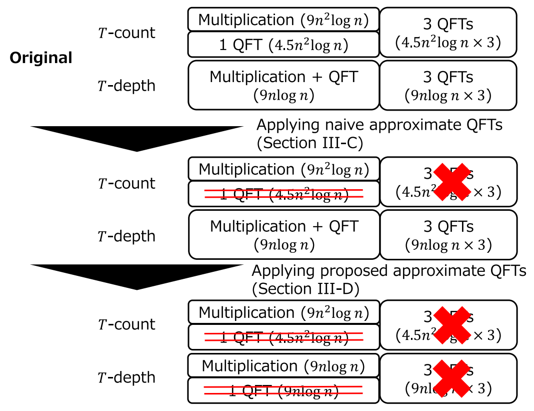
Original
T-count
Multiplication ($9n^2\log n$)
1 QFT ($4.5n^2\log n$)
3 QFTs ($4.5n^2\log n \times 3$)
T-depth
Multiplication + QFT ($9n\log n$)
3 QFTs ($9n\log n \times 3$)
Applying naive approximate QFTs (Section III-C)
T-count
Multiplication ($9n^2\log n$)
~~1 QFT ($4.5n^2\log n$)~~
T-depth
Multiplication + QFT ($9n\log n$)
3 QFTs ($9n\log n \times 3$)
Applying proposed approximate QFTs (Section III-D)
T-count
Multiplication ($9n^2\log n$)
~~1 QFT ($4.5n^2\log n$)~~
T-depth
Multiplication ($9n\log n$)
~~1 QFT ($9n\log n$)~~

FIGURE 1. Intuition of reducing the computational cost for the Montgomery multiplication. Major computational cost of the original circuit consists of multiplication and QFTs. By reducing the computational cost of QFTs, an efficient Montgomery multiplication is realized.

for reducing the computational cost for the Montgomery multiplication is shown in Fig. 1, namely reducing the computational cost of QFTs. In Section III-A, the thresholds in approximation are discussed. Rines and Chuang [18] original Montgomery multiplication is modified for Shor's algorithm in Section III-B because we cannot directly apply their method to Shor's algorithm. Moreover, a method for reducing the computational cost by approximate QFTs is proposed in Sections III-C and III-D. In Section III-C, a naive application of approximate QFTs is discussed. This implementation requires only one-third the T gates of Rines and Chuang [18] original implementation. In addition to this naive method, an approximate QFT with a smaller T-depth for realizing a smaller T-depth Montgomery multiplication is proposed in Section III-D. By applying the proposed approximate QFTs and parallelizing quantum gates by negligible ancilla qubits, this implementation requires only one-fourth of the number of T-depth of Rines and Chuang [18] original implementation.

Finally, the proposed Montgomery multiplication is applied to Shor's algorithm in Section IV. First, the proposed Montgomery multiplication is applied directly in Section IV-A. Next, a small T-depth circuit is proposed in Section IV-B by using many ancilla qubits. Moreover, similar to Oonishi et al. [23] method, a T-scheduling method is proposed for minimizing KQ (where K is the number of logical qubits and Q is the circuit depth) [24], the product of the number of qubits and depth, in Section IV-C. It is then shown that the proposed circuit takes the smallest KQ when $\Theta\left(n^{1.5}\right)$ T gates are run simultaneously.

II. PRELIMINARIES

First, notations are provided in Section II-A. Next, evaluation indices for quantum circuits are introduced in Section II-B. Moreover, the previous approximate QFT is described in Section II-C. Finally, the Fourier-basis method for the multiplication proposed by Rines and Chuang [18] is introduced in Section II-D.

A. NOTATIONS

The following gate set is used.

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

Phase gates $P(\theta)$ are defined as $\begin{bmatrix} 1 & 0 \\ 0 & \exp(i\theta) \end{bmatrix}$. Especially, P_j is defined as $P_j = P(2\pi/2^j)$. For example, $S = P_2$ and $T = P_3$. In addition, the n-qubit register is represented as $|x\rangle_n = |x_{n-1}\rangle \dots |x_1\rangle |x_0\rangle$, where $x = \sum_{j=0}^{n-1} x_j 2^j$, this subscript n is omitted when $n = 1$. Moreover, the Fourier-basis register, namely a register applied QFT, is represented by superscript Φ. For example, we represent QFT$|x\rangle_n$ as $|x\rangle_n^{\Phi}$.

B. EVALUATING INDICES FOR QUANTUM CIRCUITS

This article focuses on the number of qubits, T-count, T-depth, and KQ [24]. The emphasis is on T gates because T gates require the highest cost in this gate set. KQ is the product of the number of qubits and depth. Jones et al. [25] reported that the probability of correct output is gate precision to the power of KQ. A smaller KQ realizes quantum computation faster because quantum circuits with a smaller KQ require lower precision in each gate. Therefore, a smaller KQ is important for realizing quantum computation. This article proposes a T-scheduling method for minimizing KQ$_T$, which is defined as the product of the number of qubits and T-depth.

C. PREVIOUS APPROXIMATE QFT

In this section, approximate QFT is introduced. First, QFT without approximation is discussed in Section II-C1. Next, a naive approximate QFT is introduced in Section II-C2. Several studies have focused on how to implement approximate QFT efficiently. Nam et al. [26] approximate QFT is the focus of one of these studies. Their construction realizes a smaller T-count, and this construction is introduced in Section II-C3. The previous constructions have the same dominant term—the n-bit QFT requires approximately n qubits. Moreover, the approximate QFT of Cleve and Watrous [27] is introduced in Section II-C4. Unlike the previous constructions, their construction achieves a smaller T-depth with many ancilla qubits.

1) QFT WITHOUT APPROXIMATION

The QFT is one of the fundamental transformations in the quantum circuit. Fig. 2 shows the QFT where the number of qubits n is four. As Fig. 2 shows, the QFT consists of H gates and controlled P_j gates. The QFT is now explained in more detail. Let x be an integer satisfying $0 \leq x \leq 2^n - 1$. The QFT transforms a quantum state $|x\rangle_n$ into $\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} \exp\left(2\pi i \frac{xy}{2^n}\right) |y\rangle_n$. This QFT is realized by Algorithm 1. In Algorithm 1, $|\psi\rangle_n$ is the superposition of some $|a\rangle_n$'s. The label in $\mathbb{N}$ is set from the left qubit to the right qubit, namely, the qubit j corresponds to $|x_{n-j}\rangle$.

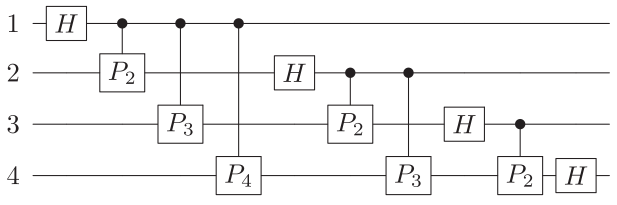

FIGURE 2. Quantum circuit of QFT where $n = 4$.

Algorithm 1: QFT.

```
Input: #(Qubits) $n$, Input state $|\psi\rangle_n$
Output: Output state QFT $|\psi\rangle_n$
1: for $j = 1$ to $n$ do
2:     Apply the $H$ gate on the qubit $j$.
3:     for $k = 1$ to $n - j$ do
4:         Apply controlled $P_{k+1}$ gates whose control qubit
           is the qubit $j$ and target bit is the qubit $j + k$.
5:     end for
6: end for
7: return QFT $|\psi\rangle_n$
```

Algorithm 2: Approximate QFT.

```
Input: #(Qubits) $n$, Input state $|\psi\rangle_n$, Threshold $\varepsilon_q$
Output: Output state AQFT $|\psi\rangle_n$
1: for $j = 1$ to $n$ do
2:     Apply the $H$ gate on the qubit $j$.
3:     for $k = 1$ to $\min(\lceil \log(1/\varepsilon_q) \rceil - 1, n - j)$ do
4:         Apply controlled $P_{k+1}$ gates whose control qubit
           is the qubit $j$ and target bit is the qubit $j + k$.
5:     end for
6: end for
7: return AQFT $|\psi\rangle_n$
```

2) NAIVE APPROXIMATE QFT

The approximate QFT realizes efficient computation by omitting phase gates having a small phase, which have a small effect on computation. In the approximate QFT, we determine threshold ε_q from the overall computation. Then, controlled phase gates whose phase is less than ε_q are omitted. The upper bound of for loop on Step 3 in Algorithm 1 is rewritten. In more detail, $n - j$ is replaced into $\min(\lceil \log(1/\varepsilon_q) \rceil - 1, n - j)$ as Algorithm 2.

3) NAM ET AL. [26] APPROXIMATE QFT

We now explain Nam et al. [26] approximate QFT. Their method parallelizes Steps 3–5 in Algorithm 2 and reduces

T-count using Gidney [28] ripple-carry adder. Let b be $\lceil \log(1/\varepsilon_q) \rceil$. In Step 4, a phase gate works when both the qubit j and the qubit $j+k$ are $|1\rangle$. Let $p_{j,k}$ be one when both the qubit j and the qubit $j+k$ are $|1\rangle$, and otherwise zero. The change of phase from qubit j is then $2\pi \mathrm{i} \sum_{k=1}^{b-1} \frac{p_{j,k} 2^{b-1-k}}{2^b}$.

Based on the earlier discussion, Nam et al. [26] method prepares the following ancilla qubits.

1) $b-1$ qubits for storing $|p_{j,k}\rangle$.
2) An auxiliary quantum state

$$|\phi\rangle_b = \frac{1}{\sqrt{2^b}} \sum_{l=0}^{2^b-1} \exp\left(-2\pi \mathrm{i} \frac{l}{2^b}\right) |l\rangle_b . \tag{1}$$

3) Ancilla $b-1$ qubits for storing carries in the adder.

Nam et al. [26] method then calculates Steps 3–5 in Algorithm 2 as follows.

1) First, $p_{j,k}$ is calculated and stored in an ancilla qubit. Especially, we calculate the AND operation with the qubit j and the qubit $j+k$ using the relative Toffoli gate.
2) Let $|p\rangle_{b-1}$ be $|p_{j,1} \dots p_{j,b-1}\rangle$.
3) Then, $|p\rangle_{b-1}$ is added into $|\phi\rangle_b$ using Gidney [28] ripple-carry adder. After this calculation, the register $|\phi\rangle_b$ changes into $\exp\left(2\pi \mathrm{i} \frac{p}{2^b}\right) |\phi\rangle_b$.
4) Finally, $p_{j,k}$ is reset by measurement.

By adopting the previous method, the T-count is $8n \log(1/\varepsilon_q)$ with $3 \log(1/\varepsilon_q)$ ancilla qubits. In each control qubit j, this method requires $4b$ T gates in the calculation of $|p_{j,k}\rangle$ and $4b$ T gates in the adder. Therefore, this method requires $8n \log(1/\varepsilon_q)$ T gates in total.

4) APPROXIMATE QFT OF CLEVE AND WATROUS [27]

We now explain Cleve and Watrous [27] approximate QFT. QFT on $|x\rangle$ realizes the following state:

$$\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} \exp\left(2\pi \mathrm{i} \frac{xy}{2^n}\right) |y\rangle_n$$
$$= \bigotimes_{k=0}^{n-1} \frac{1}{\sqrt{2}} \left(|0\rangle + \exp\left(2\pi \mathrm{i} \sum_{j=0}^{n-1-k} \frac{x_{n-1-k-j}}{2^{j+1}}\right) |1\rangle \right). \tag{2}$$

Now, let b be $\lceil \log(1/\varepsilon_q) \rceil$, similar to Section II-C. In the approximate QFT, the above phases of $|1\rangle$ in the state (2) are then replaced as

$$2\pi \mathrm{i} \sum_{j=0}^{\min(b-1, n-1-k)} \frac{x_{n-1-k-j}}{2^{j+1}}. \tag{3}$$

Their approximate QFT calculates the Fourier-basis state of all qubits simultaneously. In more detail, their method calculates

$$\frac{1}{\sqrt{2}} \left(|0\rangle + \exp\left(2\pi \mathrm{i} \sum_{j=0}^{\min(b-1, n-1-k)} \frac{x_{n-1-k-j}}{2^{j+1}}\right) |1\rangle \right) \tag{4}$$

corresponding to qubit k simultaneously. To realize the earlier calculation, we prepare $|x\rangle_n |0\rangle_{n(b-1)} |0\rangle_n |0\rangle_{n(b-1)}$, where the first register is an input value, the second register is $b-1$ ancilla qubits for each input qubit, the third register is an output value, and the fourth register is $b-1$ ancilla qubits for each output qubit. The approximate QFT is calculated as follows.

1) An H gate is applied to qubit k in the third register.
2) The value of this qubit k is copied on corresponding $b-1$ ancilla qubits in the fourth register. The state $1/\sqrt{2}(|0\dots0\rangle_b + |1\dots1\rangle_b)$ is then obtained. This state is defined as $|\psi'\rangle_b$. Moreover, $b-1$ copies of $|x\rangle_n$ are generated by copying the value of each qubit $|x_j\rangle$ on corresponding $b-1$ ancilla qubits in the second register.
3) Controlled phase gates are applied. In more detail, a controlled P_{j+1} gate is applied whose control qubit is one of $|x_{n-1-k-j}\rangle$'s and target qubit is $|\psi'_j\rangle$.
4) The value in the second and fourth registers is then erased.

We then obtain $|x\rangle_n |0\rangle_{n(b-1)} (\mathrm{QFT}\,|x\rangle_n) |0\rangle_{n(b-1)}$. We erase the value of the first register by the inverse method from the third register to the first register.

The previous method requires $2n \log(1/\varepsilon_q)$ qubits, $2n \log(1/\varepsilon_q)$ controlled P_j gates, and two controlled P_j-depth. As discussed later in Section III-A, the desired ε_q is $\Theta(n^{-2})$. Thus, we require $4n \log n$ qubits to run these gates simultaneously, which is larger compared with Nam et al. [26] approximate QFT. However, this method calculates all P_j gates simultaneously and requires $O(\log n)$ CNOT depth and two controlled P_j-depth. Therefore, it requires a smaller depth compared with Nam et al. [26] approximate QFT.

D. PREVIOUS MULTIPLIER [18]

In this section, we explain Rines and Chuang [18] controlled modular multiplier using Montgomery reduction. The controlled modular multiplier is the calculation given as

$$|c\rangle |y\rangle_n \rightarrow \begin{cases} |0\rangle |y\rangle_n & (c=0) \\ |1\rangle |Xy \bmod N\rangle_n & (c=1) \end{cases} \tag{5}$$

where X is a classical n-bit number. This controlled modular multiplier consists of two modular multipliers without control. First, the controlled modular multiplier is decomposed into modular multipliers without control in Section II-D1. Realizing modular multipliers without control is discussed in Section II-D2. Finally, the calculation cost of a modular multiplier is explained in Section II-D3.

1) DECOMPOSITION OF CONTROLLED MODULAR MULTIPLIER

In this decomposition, $|c\rangle\,|y\rangle_n\,|0\rangle_n$ is prepared. The first register $|c\rangle$ is a control qubit. The controlled modular multiplier is realized as follows.

1) The second and third registers are swapped when the first register $|c\rangle$ is $|0\rangle$. Then, $\begin{cases}|0\rangle\,|0\rangle_n\,|y\rangle_n & (c=0)\\ |1\rangle\,|y\rangle_n\,|0\rangle_n & (c=1)\end{cases}$.
2) Next, X times the second register is added into the third register. Then, $\begin{cases}|0\rangle\,|0\rangle_n\,|y\rangle_n & (c=0)\\ |1\rangle\,|y\rangle_n\,|Xy\rangle_n & (c=1)\end{cases}$.
3) The second and third registers are swapped when the first register $|c\rangle$ is $|1\rangle$. Then, $\begin{cases}|0\rangle\,|0\rangle_n\,|y\rangle_n & (c=0)\\ |1\rangle\,|Xy\rangle_n\,|y\rangle_n & (c=1)\end{cases}$.
4) Then, $-X^{-1} \bmod N$ times the second register is added into the third register. Actually, we calculate the inverse of a calculation adding $X^{-1} \bmod N$ times of the second register into the third register. Then, $\begin{cases}|0\rangle\,|0\rangle_n\,|y\rangle_n & (c=0)\\ |1\rangle\,|Xy\rangle_n\,|0\rangle_n & (c=1)\end{cases}$.
5) The second and third registers are swapped when the first register $|c\rangle$ is $|0\rangle$. Then, $\begin{cases}|0\rangle\,|y\rangle_n\,|0\rangle_n & (c=0)\\ |1\rangle\,|Xy\rangle_n\,|0\rangle_n & (c=1)\end{cases}$.

In the previous calculation, operations 2) and 4) are modular multipliers without control, and these operations require $\Theta(n^2 \log n)$ T-count and $\Theta(n \log n)$ T-depth, respectively. Different from operations 2) and 4), operations 1), 3), and 5) require $O(n)$ T-count and $O(n)$ T-depth, respectively. Therefore operations 1), 3), and 5) are negligible compared to operations 2) and 4). In conclusion, a controlled modular multiplier consists of almost two modular multipliers without control.

2) CONSTRUCTION OF MODULAR MULTIPLIER

First, it is necessary to explain Montgomery reduction. Montgomery reduction realizes efficient multiplication. Then, the multiplication of two n-bit integers x and y must be explained. Let R be a constant, and x and y are transformed as $xR \bmod N$, $yR \bmod N$. We then calculate the multiplication by

$$(xR)\cdot(yR)\cdot R^{-1} = xyR \tag{6}$$

on modulo N. This calculation is efficient when we set R as the power of two. Let m be $\lceil \log n \rceil$, and let R be 2^m for simplicity.

We now explain how to realize a modular multiplier. This method requires $m+1$ more ancilla qubits. We then calculate a modular multiplier as

$$\begin{aligned}&|yR \bmod N\rangle_n\,|0\rangle_{m+1}\,|0\rangle_n\\ &\rightarrow |yR \bmod N\rangle_n\,|XyR \bmod N\rangle_n\,|0\rangle_{m+1}.\end{aligned} \tag{7}$$

We then calculate a modular multiplier as follows.

1) *Multiplication:* We calculate a multiplication as

$$|yR \bmod N\rangle_n\,|0\rangle_{m+1}\,|0\rangle_n = |yR \bmod N\rangle_n\,|0\rangle_{n+m+1} \tag{8}$$

$$\rightarrow |yR \bmod N\rangle_n\,|0\rangle^{\Phi}_{n+m+1} \tag{9}$$

$$\rightarrow |yR \bmod N\rangle_n\,|t\rangle^{\Phi}_{n+m+1} \tag{10}$$

where $t = (XR \bmod N)(yR \bmod N)$. Naively, the following are performed in this order.

a) A QFT on the second register, state (8) → (9).
b) Fourier-basis multiplication, which is represented as "Mult," state (9) → (10).

However, to reduce the T-depth, these calculations are performed simultaneously. The previous calculation consists of H gates and controlled P_j gates. In P_j gates, a control qubit and a target qubit are interchangeable. Therefore, all the controlled P_j gates whose control qubit is in the second register of the state (8) are performed simultaneously.

2) *Reduction:* We apply Montgomery reduction on the second register of the state (10) and only focus on this register. We then conduct the following.

a) *Estimation:* We multiply $2^{-m} \bmod N$ in this procedure. In more detail, we perform the following from $j = 1$ to m.
   i) The lower order $j-1$ qubits are disregarded.
   ii) The H gate is applied on the LSB.
   iii) The LSB is set as a control qubit. Then, $(N-1)/2$ is subtracted from the remaining qubits with this control qubit.

   A calculation in each j corresponds to division by two. The calculation result is

$$|(t-uN)/2^m\rangle^{\Phi}_{n+1}\,|u\rangle_m. \tag{11}$$

   The first register of state (11) takes from $-(N-1)$ to $N-1$, when the MSB of this register is regarded as the sign bit. This value is corrected from 0 to $N-1$ in the *Extraction* and *Correction* procedures.

b) *Extraction:* The sign qubit on the first register of state (11) is extracted, and only this register is focused on. QFT^{-1} is applied, and the MSB is extracted as the sign qubit. Moreover, QFT is applied to the lower-order n qubits. This procedure changes state (11) as

$$|s\rangle\,|(t-uN)/2^m\rangle^{\Phi}_n\,|u\rangle_m \tag{12}$$

   where $|s\rangle$ is the sign qubit. In Rines and Chuang [18] construction, QFT^{-1}, the first operation, is combined with the previous *Estimation* procedure, and QFT, the second operation, is combined with the later *Correction* procedure.

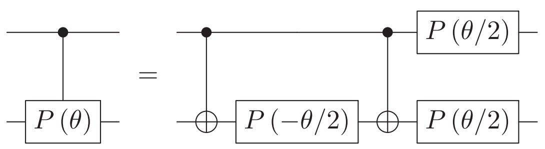

FIGURE 3. Quantum circuit of controlled P_j gates.

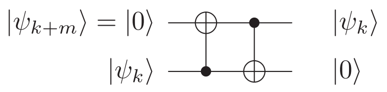

FIGURE 4. SWAP gate.

c) *Correction:* The exact $t2^{-m} \bmod N$ is calculated in this procedure. Then, the focus is on all registers of state (12). The first register is set as a control qubit and N is added to the second register using this control qubit. Then, the CNOT gate is applied, whose control qubit is the LSB of the second register and target qubit is the first register. Moreover, the first register is added as the MSB of the third register. The result state is then

$$|t2^{-m} \bmod N\rangle_n^{\Phi} \, |tN^{-1} \bmod 2R\rangle_{m+1} . \tag{13}$$

3) *Uncomputation:* By the previous procedure, we obtained the following state:

$$|yR \bmod N\rangle_n \, |XyR \bmod N\rangle_n^{\Phi} \, |tN^{-1} \bmod 2R\rangle_{m+1} \tag{14}$$

because $t = (XR \bmod N)(yR \bmod N)$. The focus is on all registers of state (14). First, a QFT is applied to the third register. We then simultaneously conduct the following as *Multiplication*.

a) An inverse QFT is applied to the second register.
b) Then, $(XR \bmod N)N^{-1} \bmod 2R$ times of the first register is subtracted from the third register, which is represented as "Inv mult."

Finally, the third register is reset into $|0\rangle$ by applying H gates to these qubits.

3) COMPUTATIONAL COST OF A MODULAR MULTIPLIER

In the previous construction, controlled $P(\theta)$ gates, which require approximately one $P(-\theta/2)$ gate [18], are only non-Clifford gates. Controlled $P(\theta)$ gates naively require one $P(-\theta/2)$ gate and two $P(\theta/2)$ gates, as shown in Fig. 3. We only employ one $P(-\theta/2)$ gate between two CNOT gates. The other two $P(\theta/2)$ gates can be combined with other phase gates applied on the same qubit if there is no H gate between them. We can calculate correctly under this summary because the value of a qubit does not change because of the phase gates. Therefore, the computational cost of two $P(\theta/2)$ gates is small compared to $P(-\theta/2)$ in Fig. 3.

According to the earlier discussion, controlled phase gates require almost one phase gate. A phase gate requires $3\log(1/\varepsilon_a)$ T gates, where ε_a is the precision of phase gates [20]. In more detail, ε_a is the upper bound of $\|U - P(\theta)\|$, where U is an approximate matrix of phase gate. Therefore, controlled phase gates require almost $3\log(1/\varepsilon_a)$ T gates in Rines and Chuang [18] construction. By using this precision ε_a, T-count is $9n^2\log(1/\varepsilon_a)$ and T-depth is $12n\log(1/\varepsilon_a)$, as shown in Table 2.

TABLE 2. Computational Cost of Montgomery Multiplication of n-Bit Number With QFT

Procedure	T-count	T-depth
Multiplication	$4.5n^2\log(1/\varepsilon_a)$	$3n\log(1/\varepsilon_a)$
Reduction	$3n^2\log(1/\varepsilon_a)$	$6n\log(1/\varepsilon_a)$
Uncomputation	$1.5n^2\log(1/\varepsilon_a)$	$3n\log(1/\varepsilon_a)$
Total	$9n^2\log(1/\varepsilon_a)$	$12n\log(1/\varepsilon_a)$

III. IMPROVEMENT OF MONTGOMERY MULTIPLICATION BY APPLYING APPROXIMATE QFTS

In this section, we explain how to improve the Montgomery multiplication proposed by Rines and Chuang [18]. Their quantum circuit is improved, where the dominant term is $2n$, almost the same number of qubits with their construction. Tables 3 and 4 show the T-count and T-depth of the proposed Montgomery multiplication, respectively. The dominant term is mainly discussed in this section. We now explain how to realize implementation realizing the computational cost as shown in Tables 3 and 4.

Before explaining the proposed method, how to set thresholds ε_q and ε_a is discussed in Section III-A. Previous research used the thresholds ε_q [22], [26] and ε_a [18]. These thresholds are discussed more rigorously to apply both approximations to the Montgomery multiplication.

In the remaining sections, the proposed method is described. In these discussions, m is $\lceil \log n \rceil$ and R is 2^m for simplicity, as in Section II-D2.

First, modification for Rines and Chuang [18] controlled modular multiplication is proposed because their method cannot calculate correctly when using Montgomery reduction. A method for calculating the controlled modular multiplication correctly on Montgomery reduction is provided in Section III-B.

Next, the computational cost is evaluated by the naively application of approximate QFTs in Section III-C. Because approximate QFTs are applied, the T-count is only one-third that of Rines and Chuang [18] original implementation.

Finally, the T-depth of the previous quantum circuit is minimized in Section III-D. The method minimizes the T-depth by a new approximate QFT proposed in Section III-D1. Moreover, the proposed method minimizes the T-depth by parallelizing controlled phase gates by using ancilla qubits, as discussed in Section III-D2. The proposed Montgomery multiplication then requires only one-fourth of the T-depth of Rines and Chuang [18] original implementation in Section III-D3.

TABLE 3. ***T*-Count of Montgomery Multiplication of *n*-Bit Number Using Approximate QFT**

		Rines and Chuang [18]		Naive (Section III-C)		Proposed (Section III-D)	
Procedure	Detail	Raw	Substituted	Raw	Substituted	Raw	Substituted
Multiplication	n-QFT	$\mathbf{1.5n^2 \log(1/\varepsilon_a)}$	$\mathbf{4.5n^2 \log n}$	$3n \log(1/\varepsilon_q) \log(1/\varepsilon_a)$	$18n (\log n)^2$	$32n \log(1/\varepsilon_q)$	$64n \log n$
	Mult	$\mathbf{3n^2 \log(1/\varepsilon_a)}$	$\mathbf{9n^2 \log n}$	$\mathbf{3n^2 \log(1/\varepsilon_a)}$	$\mathbf{9n^2 \log n}$	$\mathbf{3n^2 \log(1/\varepsilon_a)}$	$\mathbf{9n^2 \log n}$
Reduction	Estimation	$3n \log n \log(1/\varepsilon_a)$	$9n (\log n)^2$	$3n \log n \log(1/\varepsilon_a)$	$9n (\log n)^2$	$3n \log n \log(1/\varepsilon_a)$	$9n (\log n)^2$
	Inv n-QFT	$\mathbf{1.5n^2 \log(1/\varepsilon_a)}$	$\mathbf{4.5n^2 \log n}$	$3n \log(1/\varepsilon_q) \log(1/\varepsilon_a)$	$18n (\log n)^2$	$32n \log(1/\varepsilon_q)$	$64n \log n$
	n-QFT	$\mathbf{1.5n^2 \log(1/\varepsilon_a)}$	$\mathbf{4.5n^2 \log n}$	$3n \log(1/\varepsilon_q) \log(1/\varepsilon_a)$	$18n (\log n)^2$	$32n \log(1/\varepsilon_q)$	$64n \log n$
	Correction	$3n \log(1/\varepsilon_a)$	$9n \log n$	$3n \log(1/\varepsilon_a)$	$9n \log n$	$3n \log(1/\varepsilon_a)$	$9n \log n$
Uncomputation	$\log n$-QFT	$1.5 (\log n)^2 \log(1/\varepsilon_a)$	$4.5 (\log n)^3$	$1.5 (\log n)^2 \log(1/\varepsilon_a)$	$4.5 (\log n)^3$	$1.5 (\log n)^2 \log(1/\varepsilon_a)$	$4.5 (\log n)^3$
	Inv n-QFT	$\mathbf{1.5n^2 \log(1/\varepsilon_a)}$	$\mathbf{4.5n^2 \log n}$	$3n \log(1/\varepsilon_q) \log(1/\varepsilon_a)$	$18n (\log n)^2$	$32n \log(1/\varepsilon_q)$	$64n \log n$
	Inv mult	$3n \log n \log(1/\varepsilon_a)$	$9n (\log n)^2$	$3n \log n \log(1/\varepsilon_a)$	$9n (\log n)^2$	$3n \log n \log(1/\varepsilon_a)$	$9n (\log n)^2$
	Reset	0	0	0	0	0	0
Total		$9n^2 \log(1/\varepsilon_a)$	$27n^2 \log n$	$3n^2 \log(1/\varepsilon_a)$	$9n^2 \log n$	$3n^2 \log(1/\varepsilon_a)$	$9n^2 \log n$

This table shows the dominant term. In this table, x-QFT means the QFT whose dimension is $\Theta(x)$, and Inv x-QFT is the inverse QFT whose dimension is $\Theta(x)$. "Extraction" procedure in *reduction* has already been represented as two QFTs. Moreover, the parameters ε_q and ε_a are used. The T-count is also shown by substituting n^{-2} for ε_q and n^{-3} for ε_a in "substituted" while "raw" shows T-count using ε_q and ε_a.
Boldface entities show the major dominant terms.

TABLE 4. ***T*-Depth of Montgomery Multiplication of *n*-Bit Number Using Approximate QFT**

		Rines and Chuang [18]		Naive (Section III-C)		Proposed (Section III-D)	
Procedure	Detail	Raw	Substituted	Raw	Substituted	Raw	Substituted
Multiplication	n-QFT	$\mathbf{3n \log(1/\varepsilon_a)}$	$\mathbf{9n \log n}$	$\mathbf{3n \log(1/\varepsilon_a)}$	$\mathbf{9n \log n}$	$8n \log\log(1/\varepsilon_q)$	$8n \log\log n$
	Mult					$\mathbf{3n \log(1/\varepsilon_a)}$	$\mathbf{9n \log n}$
Reduction	Estimation	$\mathbf{3n \log(1/\varepsilon_a)}$	$\mathbf{9n \log n}$	$\mathbf{3n \log(1/\varepsilon_a)}$	$\mathbf{9n \log n}$	$3n \log(1/\varepsilon_a)/k$	$9n \log n/k$
	Inv n-QFT					$8n \log\log(1/\varepsilon_q)$	$8n \log\log n$
	n-QFT	$\mathbf{3n \log(1/\varepsilon_a)}$	$\mathbf{9n \log n}$	$\mathbf{3n \log(1/\varepsilon_a)}$	$\mathbf{9n \log n}$	$8n \log\log(1/\varepsilon_q)$	$8n \log\log n$
	Correction					$3n \log(1/\varepsilon_a)/k \log n$	$9n/k$
Uncomputation	$\log n$-QFT	$3 \log n \log(1/\varepsilon_a)$	$9 (\log n)^2$	$3 \log n \log(1/\varepsilon_a)$	$9 (\log n)^2$	$3 \log n \log(1/\varepsilon_a)$	$9 (\log n)^2$
	Inv n-QFT	$\mathbf{3n \log(1/\varepsilon_a)}$	$\mathbf{9n \log n}$	$\mathbf{3n \log(1/\varepsilon_a)}$	$\mathbf{9n \log n}$	$8n \log\log(1/\varepsilon_q)$	$8n \log\log n$
	Inv mult					$3n \log(1/\varepsilon_a)/k$	$9n \log n/k$
	Reset	0	0	0	0	0	0
Total		$12n \log(1/\varepsilon_a)$	$36n \log n$	$12n \log(1/\varepsilon_a)$	$36n \log n$	$3n \log(1/\varepsilon_a)$	$9n \log n$

This table shows the dominant term. In this table, x-QFT means the QFT whose dimension is $\Theta(x)$, and Inv x-QFT is the inverse QFT whose dimension is $\Theta(x)$. "Extraction" procedure in *reduction* has already been represented as two QFTs. Moreover, the parameters ε_q and ε_a are used. The parameter k corresponds to the number of ancilla qubits, namely we add $k \log n$ Qubits, where $k = \omega(1)$. The T-depth is also shown by substituting n^{-2} for ε_q and n^{-3} for ε_a in "substituted" while "Raw" shows T-depth using ε_q and ε_a.
Boldface entities show the major dominant terms.

A. EVALUATING APPROXIMATION ERROR FROM THRESHOLDS

In this section, thresholds ε_q and ε_a in Shor's algorithm are discussed. Especially, the approximation error occurring from these thresholds is considered. First, each approximation error is analyzed in Section III-A1. We can then address these errors easily. Next, the approximation error on Shor's algorithm from thresholds ε_q and ε_a is evaluated in Section III-A2. It is then shown that Shor's algorithm requires $\varepsilon_q = \Theta(n^{-2})$ and $\varepsilon_a = \Theta(n^{-3})$ for outputting the correct answer with a high probability.

1) APPROXIMATION ERROR IN EACH GATE

This section addresses thresholds ε_q and ε_a. The thresholds ε_q and ε_a deal with different types of error. The threshold ε_q addresses only the phase error. The threshold ε_a comprises flip error in addition to phase error, and these errors are dependent on previous research [29]. These errors are then decomposed into flip error and phase error to address them independently. Especially, it is shown that $O(\varepsilon_a^2)$ flip error and $O(\varepsilon_a)$ phase error occur in a phase gate.

We now review the threshold ε_a [29]. This threshold ε_a is used for approximating a phase gate $P(\theta)$. Let U be

$$U = \begin{bmatrix} \alpha + \beta \mathrm{i} & t \\ \bar{t} & \alpha - \beta \mathrm{i} \end{bmatrix} \tag{15}$$

which is an approximate matrix of $P(\theta)$ gate. The matrix of the $P(\theta)$ gate is written as

$$P(\theta) = \begin{bmatrix} x + y\mathrm{i} & 0 \\ 0 & x - y\mathrm{i} \end{bmatrix}. \tag{16}$$

This representation is consistent with Section II-A without a global phase. Let $\boldsymbol{u}$ be $[\alpha, \beta]^{\mathrm{T}}$ and $\boldsymbol{z}$ be $[x, y]^{\mathrm{T}}$. Moreover, let ϕ be an angle between two vectors $\boldsymbol{u}$ and $\boldsymbol{z}$. This angle ϕ corresponds to phase error. Then, threshold ε_a satisfies the following equation:

$$\|U - P(\theta)\|^2 \leq \varepsilon_a^2 \Leftrightarrow \|\boldsymbol{u}\| \, \|\boldsymbol{z}\| \cos\phi \geq 1 - \frac{\varepsilon_a^2}{2} \tag{17}$$

which is shown by Selinger [29].

We now decompose approximation errors into flip error and phase error to address them independently. First, the flip error by matrix U is examined. In (17)

$$\|\boldsymbol{u}\| \geq 1 - \frac{\varepsilon_a^2}{2} \tag{18}$$

because $\|\boldsymbol{z}\| = 1$ and $\cos\phi \leq 1$. Therefore, the flip error is less than $\varepsilon_a^2/2$.

Next, the phase error ϕ by matrix U is investigated. From (17)

$$\cos\phi \geq 1 - \frac{\varepsilon_a^2}{2} \tag{19}$$

because $\|\boldsymbol{u}\| \leq 1$ and $\|z\| = 1$. Then, ϕ satisfying (19) is calculated. It is assumed that $\phi \ll 1$. First the upper bound of $\cos\phi$ is calculated using

$$f(\phi) \equiv 1 - \frac{\phi^2}{2} + \frac{\phi^4}{24} - \cos\phi \geq 0. \tag{20}$$

Equation (20) is true when $\phi \geq 0$, because $f(0) = 0, f^{(1)}(0) = 0, f^{(2)}(0) = 0, f^{(3)}(0) = 0,$ and $f^{(4)}(\phi) \geq 0$. From (20)

$$\cos\phi \leq 1 - \frac{\phi^2}{2} + \frac{\phi^4}{24} \leq 1 - \frac{11}{24}\phi^2 \tag{21}$$

where $\phi \ll 1$. Therefore, from (19) and (21)

$$1 - \frac{11}{24}\phi^2 \geq 1 - \frac{\varepsilon_a^2}{2} \Leftrightarrow -\sqrt{\frac{12}{11}}\varepsilon_a \leq \phi \leq \sqrt{\frac{12}{11}}\varepsilon_a. \tag{22}$$

Thus, phase error ϕ is $O(\varepsilon_a)$.

2) APPROXIMATION ERROR IN SHOR'S ALGORITHM

Next, the approximation error is evaluated from the thresholds ε_q and ε_a for Shor's algorithm. Especially, it is shown that Shor's algorithm requires $\varepsilon_q = \Theta(n^{-2})$ and $\varepsilon_a = \Theta(n^{-3})$ to output the correct answer with a high probability.

A quantum circuit of the Montgomery multiplication adopts thresholds ε_q and ε_a on controlled phase gates. The thresholds ε_q and ε_a have phase error and flip error.

The analysis of flip error is simple, focusing only on the correct state without flip. Let M_f be the number of all controlled phase gates. The probability of no flip error is then

$$\left(1 - O\left(\varepsilon_a^2\right)\right)^{M_f} \tag{23}$$

because the probability of no flip error in each controlled phase gate is $1 - O(\varepsilon_a^2)$.

Next, the phase error is examined. Phase error is calculated on the following procedure iterated in Shor's algorithm:

1) approximate QFT;
2) actual calculation on Fourier basis;
3) approximate inverse QFT.

It is assumed that the dimension of the approximate QFT is n. Moreover, let M_p be the maximal number of controlled P_j gates on each qubit.

First, an approximate QFT is considered. In this calculation, phase error occurs from thresholds ε_q and ε_a. The maximal value of phase error from the threshold ε_q in each qubit is

$$2\pi\mathrm{i} \sum_{j=\lceil \log(1/\varepsilon_q)\rceil}^{\infty} \frac{1}{2^{j+1}} \approx 2\pi\mathrm{i}\varepsilon_q. \tag{24}$$

Moreover, the maximal value of phase error occurs from the threshold ε_a is $\mathrm{i}O(\varepsilon_a)$ in each controlled phase gate. There are at most $\lceil \log(1/\varepsilon_q)\rceil$ controlled phase gates in each qubit. Therefore, the maximal value of phase error from the threshold ε_a in each qubit is $\mathrm{i}O(\varepsilon_a)\log(1/\varepsilon_q)$ In conclusion, the phase error in each qubit is then $2\pi\mathrm{i}(\varepsilon_q + O(\varepsilon_a)\log(1/\varepsilon_q))$.

Next, we must consider the actual calculation of the Fourier basis. In the previous discussion, controlled P_j gates on QFTs are omitted. However, we cannot omit phase gates in this procedure because the phase is not always small. The error source in each qubit is the phase error from the threshold ε_a. Therefore, the phase error in each qubit is $M_p O(\varepsilon_a)$.

Finally, an inverse approximate QFT is considered. Similar to Step 1), the phase error in each qubit is $2\pi\mathrm{i}(\varepsilon_q + O(\varepsilon_a)\log(1/\varepsilon_q))$. By gathering the above phase errors, the total phase error in each qubit is

$$O\left(\varepsilon_q + \left(M_p + \log\left(1/\varepsilon_q\right)\right)\varepsilon_a\right). \tag{25}$$

We now discuss how phase error affects the calculation. Phase error is converted into flip error by an approximate inverse QFT. In the approximate inverse QFT, we transform the Fourier basis into the standard basis from the LSB to the MSB. Especially, we recover each qubit with phase error ε as follows where $y \in \{0, 1\}$

$$H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}\exp\left(2\pi\mathrm{i}\left(\frac{y}{2} + \varepsilon\right)\right)|1\rangle\right)$$
$$= H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{(-1)^y}{\sqrt{2}}\exp\left(2\pi\mathrm{i}\varepsilon\right)|1\rangle\right) \tag{26}$$
$$= \frac{1 + (-1)^y\exp\left(2\pi\mathrm{i}\varepsilon\right)}{2}|0\rangle + \frac{1 - (-1)^y\exp\left(2\pi\mathrm{i}\varepsilon\right)}{2}|1\rangle \tag{27}$$
$$= \frac{1 + \exp\left(2\pi\mathrm{i}\varepsilon\right)}{2}|\mathrm{Correct}\rangle + \frac{1 - \exp\left(2\pi\mathrm{i}\varepsilon\right)}{2}|\mathrm{Incorrect}\rangle. \tag{28}$$

The probability $p_I(\varepsilon)$ with the $|\mathrm{Incorrect}\rangle$ state is then

$$p_I(\varepsilon) = \sqrt{\left(\frac{1 - \cos\pi\varepsilon}{2}\right)^2 + \left(-\frac{\sin\pi\varepsilon}{2}\right)^2} \tag{29}$$
$$= \sqrt{\frac{1 - \cos\pi\varepsilon}{2}} \tag{30}$$
$$= O(\varepsilon) \tag{31}$$

because $p_I(0) = 0$ and $p_I'(\varepsilon) = \pi\sqrt{(1 + \cos\varepsilon)/8}$. Therefore, the flip error occurring from phase error is almost the same as the error (25), and the probability of no flip error is then

$$\left(1 - O\left(\varepsilon_q + \left(M_p + \log\left(1/\varepsilon_q\right)\right)\varepsilon_a\right)\right)^n. \tag{32}$$

We now summarize probabilities (23) and (32) and calculate the probability of obtaining the desired quantum states in Shor's algorithm. In Shor's algorithm, the number of controlled P_j gates is $O(n^3)$ [18], which means $M_f = O(n^3)$. Modular exponentiation uses $O(n)$ controlled modular multiplier [1], [30], with $O(1)$ QFTs and $O(n)$ controlled P_j gates in each qubit. Especially, one controlled modular multiplier outputs the desired state with the probability (32) where

$M_p = O(n)$. Therefore, the probability of obtaining the desired quantum states in Shor's algorithm is

$$\left(1 - O\left(\varepsilon_a^2\right)\right)^{M_f} \left(1 - O\left(\varepsilon_q + \left(M_p + \log\left(1/\varepsilon_q\right)\right)\varepsilon_a\right)\right)^{O(n^2)} \sim 1 - O\left(n^3\varepsilon_a^2 + n^2\varepsilon_q + n^3\varepsilon_a + n^2\log\left(1/\varepsilon_q\right)\varepsilon_a\right). \tag{33}$$

This probability (33) is $\Omega(1)$ when $\varepsilon_q = O(n^{-2})$ and $\varepsilon_a = O(n^{-3})$.

In the previous discussion, the focus is on modular exponentiation. However, Shor's algorithm obtains the output by inverse QFT. Therefore, we should consider the effect of approximation on this inverse QFT. Let r be the order of y of a modulo N, namely r is the minimal value satisfying $y^r = 1 \bmod N$ where $r \in \mathbb{N}$.

The exact Shor's algorithm [1] is calculated as

$$|0\rangle_{2n}|1\rangle_n \to \sum_{j=0}^{2^{2n}-1} |j\rangle_{2n}|1\rangle_n \tag{34}$$

$$\to \sum_{j=0}^{2^{2n}-1} |j\rangle_{2n}|y^j \bmod N\rangle_n \tag{35}$$

$$\to \sum_{j=0}^{2^{2n}-1}\sum_{l=0}^{2^{2n}-1} \exp\left(-2\pi \mathrm{i}\frac{l}{2^{2n}}j\right)|l\rangle_{2n}|y^j\rangle_n \tag{36}$$

$$= \sum_{k\in\{y^j|0\le j\le r-1\}}\sum_{l=0}^{2^{2n}-1} p_k|l\rangle_{2n}|k\rangle_n \tag{37}$$

where

$$p_k = \sum_{\substack{j\in\{y|0\le y\le 2^{2n}-1,\\ y^j=k \bmod N\}}} \exp\left(-2\pi \mathrm{i}\frac{l}{2^{2n}}j\right) \tag{38}$$

and the first register $|l\rangle_{2n}$ is measured. An attempt is made to obtain the state (35) by controlled multiplications. In more detail, H gates are applied on all qubits in the first register of state (34), and modular exponentiation is calculated [(34) $\to$ (35)]. Next, inverse QFT is considered, namely (35) $\to$ (36). Shor's algorithm obtains l near $2^{2n}/r$ with high probability, which takes a large p_k.

When the approximate calculation is adopted, state (35) is

$$\sum_{j=0}^{2^{2n}-1}\sum_{k=0}^{2^n-1} p_{j,k}|j\rangle_{2n}|k\rangle_n \tag{39}$$

where $p_{j,k}$ takes a high value when $y^j \bmod N = k$ and a low value otherwise. In the previous discussion, only the probability of taking the pair (j,k) satisfying $y^j \bmod N = k$ is evaluated. To clarify the effect of approximation, we must evaluate the probability in the other pairs. However, this probability is difficult to analyze, because it involves exponential branches. To analyze the probability of obtaining correct output, Assumption 1 is used.

Assumption 1:

$$p_{j,k} = \begin{cases} C, & y^j \bmod N = k \\ \dfrac{1-\dfrac{2^{3n}}{r}C}{2^{3n}-\dfrac{2^{3n}}{r}}, & \text{otherwise} \end{cases} \tag{40}$$

where $C = \Theta(1)$.

Assumption 1 means that all desired pairs take a high probability, and the other pairs take the same small probability. The number of desired pairs is $2^{3n}/r$, namely, there are $2^{2n/r}$ js in each k. Moreover, it is assumed that the other pairs take the same probability because phases on transformations from (26) to (28) are almost random in the later calculation. It is necessary to evaluate how much Assumption 1 realizes the actual calculation in the future work. Here, the effect of approximation based on Assumption 1 is discussed.

By using Assumption 1, Shor's algorithm transforms the state (39) as

$$\sum_{j=0}^{2^{2n}-1}\sum_{k=0}^{2^n-1} p_{j,k}|j\rangle_{2n}|k\rangle_n$$

$$\to \sum_{j=0}^{2^{2n}-1}\sum_{k=0}^{2^n-1}\sum_{l=0}^{2^{2n}-1} p_{j,k}\exp\left(-2\pi \mathrm{i}\frac{l}{2^{2n}}j\right)|l\rangle_{2n}|k\rangle_n \tag{41}$$

$$= \sum_{k=0}^{2^n-1}\sum_{l=0}^{2^{2n}-1}\left(\sum_{j=0}^{2^{2n}-1} p_{j,k}\exp\left(-2\pi \mathrm{i}\frac{l}{2^{2n}}j\right)\right)|l\rangle_{2n}|k\rangle_n. \tag{42}$$

Let p_k be $\sum_{j=0}^{2^{2n}-1} p_{j,k}\exp(-2\pi \mathrm{i}\frac{l}{2^{2n}}j)$, which corresponds to (38). In the state (42), only ks in the set $\{y^j|0\le j\le r-1\}$ remain for the following reasons.

1) When k does not satisfy $y^j \bmod N = k$ with any j, $p_k = 0$ because $p_{j,k}$ is a constant value from Assumption 1.
2) When k has j satisfying $y^j \bmod N = k$

$$p_k \approx C\sum_{\substack{j\in\{y|0\le y\le 2^{2n}-1,\\ y^j=k \bmod N\}}} \exp\left(-2\pi \mathrm{i}\frac{l}{2^{2n}}j\right). \tag{43}$$

Approximation (43) satisfies because $p_{j,k}$ is a constant value when $y^j \bmod N \ne k$ from Assumption 1, and their phase uniformly distributes from 0 to 2π.

Therefore, when we apply approximation, we obtained the state (37) under p_k satisfying (43). Equations (38) and (43) only differ in the constant term C. Thus, the exact and approximate Shor's algorithms have almost the same output—they obtain l near $2^{2n}/r$ with high probability. Similar reasoning is applicable to Ekerå and Håstad [30] construction—the states are canceled when they are not calculated in the exact

Algorithm 3: *m*-Bit Left Shift.

```
Input: #(Qubits) $n$, #(Shift) $m$, Input state $|\psi\rangle_{n+m+1}$,
Output: Output state $|\psi'\rangle_{n+m+1}$
1: for $j = 0$ to $\lceil n/m \rceil - 1$ do
     top $= n - mj$
2:     for $k =$ top $- 1$ down to max(top $- n$, 0) do
3:         Swap the qubits $|\psi\rangle_k$ and $|\psi\rangle_{k+m}$ as shown in
           Fig. 4.
4:     end for
5: end for
6: return $|\psi'\rangle_{n+m+1}$
```

algorithm. Therefore, Shor's algorithm outputs the correct answer with a high probability when $\varepsilon_q = \Theta(n^{-2})$ and $\varepsilon_a = \Theta(n^{-3})$. These ε_q and ε_a can vary in each procedure. This optimization may reduce further computational costs, but the constants ε_q and ε_a are focused on in this study.

B. MODIFICATION OF THE PREVIOUS CONTROLLED MODULAR MULTIPLICATION

In this section, a small modification on Rines and Chuang [18] controlled modular multiplication is explained. They proposed the modular multiplication given as the transformation (7). However, we cannot directly apply this modular multiplication to the controlled modular multiplication given in Step 2 in Section II-D1. In more detail, the correct calculation is

$$\begin{cases} |0\rangle\,|0\rangle_n\,|yR \bmod N\rangle_n \\ |1\rangle\,|yR \bmod N\rangle_n\,|0\rangle_n \end{cases}$$
$$\rightarrow \begin{cases} |0\rangle\,|0\rangle_n\,|yR \bmod N\rangle_n \\ |1\rangle\,|yR \bmod N\rangle_n\,|XyR \bmod N\rangle_n \end{cases} \tag{44}$$

but we actually obtain

$$\begin{cases} |0\rangle\,|0\rangle_n\,|y \bmod N\rangle_n \\ |1\rangle\,|yR \bmod N\rangle_n\,|XyR \bmod N\rangle_n\,. \end{cases} \tag{45}$$

This difference occurs because previous researchers did not consider Montgomery reduction on $|0\rangle\,|0\rangle_n\,|yR \bmod N\rangle_n$.

To correct this miscalculation, we add m-bit left shift (multiplication by $R = 2^m$) before applying the modular multiplication. The previous calculation is realized by Algorithm 3. In Algorithm 3, Steps 2–4 can be parallelized. Therefore, the number of gates is $O(n)$ and the depth is $O(n/\log n)$ in Algorithm 3, which is negligible compared with the other calculation.

C. NAIVE APPLICATION OF APPROXIMATE QFTS

This section evaluates the Montgomery multiplication using approximate QFTs, namely phase gates whose phase is less than ε_q are omitted. We use the label of qubits in QFT similar to that in Section II-C. In the Montgomery multiplication, two types of QFT exist: $\log n$-bit QFT and n-bit QFT. In the former $\log n$-bit QFT, we use all gates. Therefore, we only focus on the approximation of n-bit QFT. By this approximation, the following costs decrease:

1) an n-bit QFT in *Multiplication*;
2) two n-bit QFTs in *Reduction*;
3) an n-bit inverse QFT in *Uncomputation*.

First, the T-count is evaluated. As Algorithm 2 shows, n-qubit approximate QFT applies at most $\log(1/\varepsilon_q)$ P_k gates in Step 3. This procedure means that an approximate QFT applies $\log(1/\varepsilon_q)$ controlled P_k gates to each qubit. As noted in Section II-D3, each controlled P_k gate requires $3\log(1/\varepsilon_a)$ T gates [20]. Therefore, an approximate QFT requires $3n\log(1/\varepsilon_q)\log(1/\varepsilon_a)$ T gates. Thus, using approximate QFTs, T-count of the Montgomery multiplication is only one-third of Rines and Chuang [18] original implementation.

Next, the T-depth is evaluated. In the previous construction, all controlled P_j gates whose control qubit is in the second register of the state (8) are used simultaneously. When we apply approximate QFTs, the number of phase gates with the same control qubit decreases. However, the T-depth in each control qubit does not change. Therefore, adopting naive approximate QFTs maintains the T-depth.

D. OPTIMIZATION OF MONTGOMERY MULTIPLICATION

This section minimizes the T-depth of the Montgomery multiplication. The T-depth of the Montgomery multiplication is minimized using the following:

1) a new approximate QFT with less T-depth (see Section III-D1);
2) a method for parallelizing controlled phase gates in the "Inv mult" procedure by using ancilla qubits (see Section III-D2).

We then evaluate the computational cost by using the techniques in Section III-D3.

1) PROPOSED APPROXIMATE QFT

We now propose a new approximate QFT with less T-depth. This approximate QFT has less T-depth than that of Nam et al. [26]. Their method minimizes the number of T gates by replacing phase gates into Gidney [28] adder. However, the same calculation can be realized by the other adders because this calculation only adds two numbers without using any specific property in Gidney [28] adder. The proposed method adopts Draper et al. [16] adder using Gidney [28] relative Toffoli gates, which was discussed by Thapliyal et al. [31] and reviewed by Oonishi et al. [23]. The proposed method is shown in Algorithm 4. By using more qubits and T gates, the above adder reduces T-depth. Draper et al. [16] adder requires $O(\log n)$ T-depth while Gidney [28] adder requires $O(n)$ T-depth. Table 5 shows the computational cost of the proposed approximate QFT.

Here, the computational cost of the proposed approximate QFT is discussed. Let b be $\lceil \log(1/\varepsilon_q) \rceil$, similar to Section II-C.

TABLE 5. Computational Cost of Approximate QFT Using Adder With Threshold ε_q

	Previous [26]	Ours
Adder	[28]	[16]
#(Qubits)	$n + 3\log(1/\varepsilon_q)$	$n + 4.5\log(1/\varepsilon_q)$
T-count	$8n\log(1/\varepsilon_q)$	$32n\log(1/\varepsilon_q)$
T-depth	$2n\log(1/\varepsilon_q)$	$8n\log\log(1/\varepsilon_q)$

```
Algorithm 4: Proposed Approximate QFT.
Input: #(Qubits) $n$, Input state $|\psi\rangle_n$, $4.5\lceil\log(1/\varepsilon_q)\rceil$
  ancilla qubits, Threshold $\varepsilon_q$
Output: Output state AQFT $|\psi\rangle_n$
 1: $b = \lceil\log(1/\varepsilon_q)\rceil$
 2: Prepare an auxiliary quantum state
    $|\phi\rangle_b = \frac{1}{\sqrt{2^b}}\sum_{l=0}^{2^b-1}\exp\left(-2\pi i\frac{l}{2^b}\right)|l\rangle_b.$
 3: for $j = 1$ to $n$ do
 4:    Apply the $H$ gate on the qubit $j$.
 5:    for $k = 1$ to $\min(\lceil\log(1/\varepsilon_q)\rceil - 1, n - j)$ do
 6:       Calculate the AND operation with the qubit $j$
          and the qubit $j + k$ using the relative Toffoli
          gate. The result is represented as $|p_{j,k}\rangle$.
 7:    end for
 8:    Let $|p\rangle_{b-1}$ be $|p_{j,1}\dots p_{j,b-1}\rangle$.
 9:    $|p\rangle_{b-1}$ is added into $|\phi\rangle_b$ by Draper et al. [16]
       adder using Gidney [28] relative Toffoli gates.
10:    $|p\rangle_{b-1}$ is reset by measurement.
11: end for
12: return AQFT $|\psi\rangle_n$
```

First, regarding the number of qubits, the previous method uses $b-1$ qubits for storing $|p_{j,k}\rangle$, an auxiliary quantum state $|\phi\rangle_b$, and $b-1$ ancilla qubits for storing carries in the adder. Draper et al. [23] adder uses $2.5b$ ancilla qubits in addition to two b-qubit numbers. Therefore, the proposed method requires $4.5b$ ancilla qubits.

Next, regarding the T-count, the previous method requires T gates in the calculation of each control qubit as follows:

1) $4b$ T gates in the calculation of $|p_{j,k}\rangle$;
2) $4b$ T gates in the adder.

Draper et al. [23] adder uses $28b$ T gates in the addition of two b-qubit numbers. Therefore, the proposed method requires $32b$ T gates.

Finally, the T-depths of previous and proposed methods are discussed. In the previous study, the T-depth was not investigated. Here, the T-depth of the previous method is evaluated. The previous method consists of storing $|p_{j,k}\rangle$, addition, and resetting $|p_{j,k}\rangle$. This addition requires $2b$ T-depth from each control qubit. Now, the remaining procedure, storing and resetting $|p_{j,k}\rangle$, is discussed. Fig. 5 shows a method for storing $|p_{j,k}\rangle$, and Fig. 6 shows a method for resetting $|p_{j,k}\rangle$. For storing $|p_{j,k}\rangle$, b circuits of Fig. 5 are used. The first qubit $|x_{n-j}\rangle$ of Fig. 5 is common in these b circuits, and the other two qubits differ between these b circuits. Therefore, we can parallelize the gates excluding the CNOT gate surrounded by a dotted line, and the T-depth is $O(1)$ on these gates. Moreover, the b CNOT gates surrounded by a dotted line, whose depth is b, are used. This depth is negligible compared with the other depth because a CNOT gate is a Clifford gate. Similar reasoning is applicable to resetting $|p_{j,k}\rangle$ because a controlled-Z gate is a Clifford gate. In conclusion, Nam et al. [26] QFT requires $2b$ T-depth.

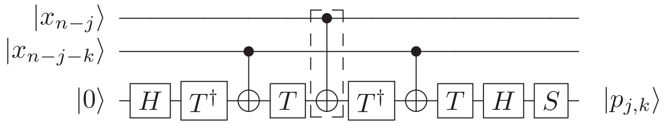

FIGURE 5. AND operation in calculating $|p_{j,k}\rangle$ [26]: We can parallelize the gates excluding the CNOT gate surrounded by a dotted line.

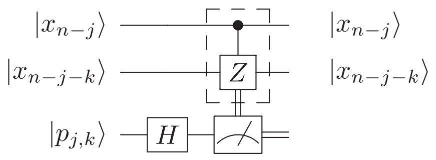

FIGURE 6. Resetting $|p_{j,k}\rangle$ [26]: We can parallelize the gates excluding the controlled-Z gate surrounded by a dotted line.

The proposed method consists of storing $|p_{j,k}\rangle$, addition, and resetting $|p_{j,k}\rangle$, similar to the previous result. This addition requires $8\log b$ T-depth from each control qubit [31]. The remaining procedure is storing and resetting $|p_{j,k}\rangle$. For storing $|p_{j,k}\rangle$, we run b CNOT gates, surrounded by a dotted line shown in Fig. 5. Naively, this procedure requires b CNOT-depth, and this depth is nonnegligible compared with $8\log b$ T-depth in addition. However, we reduce this depth by using b ancilla qubits prepared for the adder. Especially, we can store $|p_{j,k}\rangle$ as follows.

1) We copy the value of qubit $|x_{n-j}\rangle$ into b ancilla qubits. This operation requires $\log b$ CNOT-depth (Clifford), which is negligible compared with $8\log b$ T-depth (non-Clifford) in addition.
2) We store $|p_{j,k}\rangle$ by parallelized computation. This operation requires $O(1)$-depth, which is negligible compared to depth in addition.
3) We reset the value of b ancilla qubits by the inverse operation of Step 1).

Therefore, this depth is negligible compared with depth in addition. Similar reasoning is applicable to resetting $|p_{j,k}\rangle$ because a controlled-Z gate is a Clifford gate, as shown in Fig. 6. In conclusion, the proposed QFT requires $8\log b$ T-depth.

2) PARALLELIZING CONTROLLED PHASE GATES

This section proposes a method for parallelizing controlled phase gates by using ancilla qubits, which is similar to

Cleve and Watrous [27] study. The T-depth is reduced in the following procedures:

1) "Estimation" in the *Reduction* procedure;
2) "Correction" in the *Reduction* procedure;
3) "Inv mult" in the *Uncomputation* procedure.

In the computation, the P_j-depth is

$$\frac{\#\left(P_j \text{ gates}\right)}{\min(\#(\text{control qubits}), \#(\text{target qubits}))}. \tag{46}$$

Therefore, by using $k \log n$ ancilla qubits, the T-depth decreases as in Table 4. In more detail, we generate

$$\frac{k \log n}{\min(\#(\text{control qubits}), \#(\text{target qubits}))} \tag{47}$$

copies of target or control qubits with fewer qubits. Controlled phase gates are then calculated as many as possible.

3) COMPUTATIONAL COST OF PROPOSED MONTGOMERY MULTIPLICATION

In Sections III-D1 and III-D2, the T-depth of the Montgomery multiplication is minimized. In this construction, QFTs are run independently from the other procedure, and "mult" in the *Multiplication* procedure becomes the dominant term of T-depth. This T-depth is only one-fourth of Rines and Chuang [18] original implementation. The computational cost of the proposed method is given in Tables 3 and 4.

IV. APPLICATION OF PROPOSED MONTGOMERY MULTIPLICATION TO SHOR'S ALGORITHM

In this section, the proposed Montgomery multiplication is applied to Shor's algorithm. First, the proposed Montgomery multiplication is applied directly to Shor's algorithm in Section IV-A. Then, the T-depth is minimized in Section IV-B. Moreover, a quantum circuit with the smallest KQ is proposed in Section IV-C using Oonishi et al. [23] method.

Before evaluating the computational cost in each setting, we should review the total cost of Shor's algorithm. Shor's algorithm consists of modular exponentiations and an inverse QFT. Modular exponentiations, having a higher computational cost than an inverse QFT, require $3n$ times the Montgomery multiplications [30] as follows.

1) Modular exponentiations require $1.5n$ times the controlled Montgomery multiplications [30].
2) The controlled Montgomery multiplication requires two times the Montgomery multiplications [18].

In the previous construction, only one qubit is required for the control qubit for controlled Montgomery multiplications by qubit recycling [32]. We now discuss the computational cost of Shor's algorithm.

A. DIRECT APPLICATION TO PROPOSED MONTGOMERY MULTIPLICATION

This section evaluates the direct application of the proposed Montgomery multiplication. Table 4 and the discussion in Section III-D reveal that the proposed Montgomery multiplication requires $9n^2 \log n$ T gates and $9n \log n$ T-depth, where $k = \omega(1)$. Moreover, the proposed Montgomery multiplication requires $2n + k \log n$ qubits. This $k \log n$ term is negligible compared to $2n$ when $k = o(n/\log n)$, and there are k satisfying $k = \omega(1)$ and $k = o(n/\log n)$. Therefore, the proposed Montgomery exponentiations require $27n^3 \log n$ T gates and $27n^2 \log n$ T-depth with $2n$ qubits.

B. SHOR'S ALGORITHM MINIMIZING T-DEPTH

This section discusses Shor's algorithm with a low T-depth. First, the T-depth of each Montgomery multiplication is minimized in Section IV-B1. The T-depth of Shor's algorithm is then minimized by parallelizing Montgomery multiplications in Section IV-B2.

1) MINIMIZING T-DEPTH IN EACH MONTGOMERY MULTIPLICATION

The T-depth can be minimized by the following techniques:

1) Cleve and Watrous [27] approximate QFT given in Section II-C4;
2) parallelizing controlled phase gates given in Section III-D2.

Table 6 shows the computational cost by adopting the aforementioned techniques.

First, we minimize T-depth in approximate QFT. This minimization is applicable to the following procedures:

1) an n-bit QFT in *Multiplication*;
2) two n-bit QFTs in *Reduction*;
3) an n-bit inverse QFT in *Uncomputation*;
4) A $\log n$-bit QFT in *Uncomputation*.

In Steps 1–3, Cleve and Watrous [27] n-bit approximate QFT is adopted. As noted in Section II-C4, their approximate QFT requires the following computational cost:

1) $2n(1/\varepsilon_q)$ qubits;
2) $2n(1/\varepsilon_q)$ controlled P_j gates;
3) $O(\log n)$ CNOT depth and two controlled P_j-depth.

In the aforementioned computational cost, $\varepsilon_q = \Theta\left(n^{-2}\right)$. Moreover, one P_j gate requires $3 \log(\varepsilon_a)$ T gates and T-depth, namely $9 \log n$ T gates and T-depth, because $\varepsilon_a = \Theta\left(n^{-3}\right)$. Therefore, n-bit approximate QFT requires $4n \log n$ qubits, $36n(\log n)^2$ T gates, and $18 \log n$ T-depth. Next, Step 4, $\log n$-bit QFT without approximation in *Uncomputation*, is considered. The $\log n$-bit QFT requires $(\log n)^2$ qubits, $(\log n)^2/2$ controlled P_j gates, and two controlled P_j-depth. Therefore, $\log n$-bit QFT requires $(\log n)^2$ qubits, $9(\log n)^3/2$ T gates, and $18 \log n$ T-depth.

TABLE 6. Computational Cost of T-Depth Minimized Montgomery Multiplication of n-Bit Number, Which Requires $2n^2$ Ancilla Qubits

		T-count		T-depth	
Procedure	Detail	Raw	Substituted	Raw	Substituted
Multiplication	n-QFT	$6n\log(1/\varepsilon_q)\log(1/\varepsilon_a)$	$36n(\log n)^2$	$6\log(1/\varepsilon_a)$	$18\log n$
	Mult	$3n^2\log(1/\varepsilon_a)$	$9n^2\log n$	$3\log(1/\varepsilon_a)$	$9\log n$
Reduction	Estimation	$3n\log n\log\log(1/\varepsilon_a)$	$9n(\log n)^2$	$3\log(1/\varepsilon_a)$	$9\log n$
	Inv n-QFT	$6n\log(1/\varepsilon_q)\log(1/\varepsilon_a)$	$36n(\log n)^2$	$6\log(1/\varepsilon_a)$	$18\log n$
	n-QFT	$6n\log(1/\varepsilon_q)\log(1/\varepsilon_a)$	$36n(\log n)^2$	$6\log(1/\varepsilon_a)$	$18\log n$
	Correction	$3n\log\log(1/\varepsilon_a)$	$9n\log n$	$3\log(1/\varepsilon_a)$	$9\log n$
Uncomputation	$\log n$-QFT	$3(\log n)^2\log(1/\varepsilon_a)/2$	$9(\log n)^3/2$	$6\log(1/\varepsilon_a)$	$18\log n$
	Inv n-QFT	$6n\log(1/\varepsilon_q)\log(1/\varepsilon_a)$	$36n(\log n)^2$		
	Inv mult	$3n\log n\log\log(1/\varepsilon_a)$	$9n(\log n)^2$	$3\log(1/\varepsilon_a)$	$9\log n$
	Reset	0	0	0	0
Total		$3n^2\log(1/\varepsilon_a)$	$9n^2\log n$	$36\log(1/\varepsilon_a)$	$108\log n$

This table shows the dominant term. In this table, x-QFT means the QFT whose dimension is $\Theta(x)$, and Inv x-QFT is the inverse QFT whose dimension is $\Theta(x)$. The "Extraction" procedure in *Reduction* has already been represented as two QFTs. Moreover, the parameters ε_q and ε_a are used. The T-count and T-depth are also shown by substituting n^{-2} for ε_q and n^{-3} for ε_a in "Substituted" while "Raw" shows T-Count and T-Depth using ε_q and ε_a.

Algorithm 5: Parallelized Modular Exponentiation.

```
Input: #(Multiplier) $M, N, y_0, y_1, \ldots, y_{M-1}$
Output: $y_0 y_1 \ldots y_{M-1} \bmod N$
 1: for $k = 0$ to $M-1$ do
 2:     $y_{0,k} = y_k$
 3: end for
 4: for $j = 0$ to $\lceil \log M \rceil - 1$ do
 5:     for $k = 0$ to $\lfloor M/2^{j+1} \rfloor - 1$ do
 6:         $y_{j+1,k} = y_{j,2k} y_{j,2k+1} \bmod N$
 7:     end for
 8:     if $\lfloor (M-1)/2^j \rfloor \bmod 2 = 0$ then
 9:         $y_{j+1,\lfloor M/2^{j+1} \rfloor} = y_{j,2\lfloor M/2^{j+1} \rfloor}$
10:     end if
11: end for
12: return $y_{\lceil \log M \rceil, 0}$
```

Next, the other calculations are considered. We can parallelize controlled phase gates by using ancilla qubits. All gates in these calculations are controlled phase gates. If there are L controlled phase gates, these controlled phase gates are used in L/K P_j-depth by preparing $2K$ qubits (K control qubits and K target qubits). In the Montgomery multiplication, the *Multiplication* procedure employs n^2 controlled phase gates. Therefore, by preparing $2n^2$ qubits, all the calculation steps require one controlled P_j-depth: the $9\log n$ T-depth.

The total computational cost of Shor's algorithm is now evaluated. The previous discussion reveals that the number of qubits is $2n^2$ qubits. Moreover, the Montgomery multiplication requires $9n^2\log n$ T-count and $108\log n$ T-depth, as shown in Table 6. Therefore, Shor's algorithm requires $27n^3\log n$ T-count and $324n\log n$ T-depth.

2) MINIMIZING T-DEPTH BY PARALLELIZING MONTGOMERY MULTIPLICATION

Section IV-B1 considers the computational cost by assuming sequential Montgomery multiplications. However, we can parallelize these Montgomery multiplications [27], [33]. Cleve and Watrous [27] method realized modular exponentiations with $O(\log n)$-depth controlled multiplications. In more detail, we realize M Montgomery multiplications using number $y_0, y_1, \ldots, y_{M-1}$ as Algorithm 5. In Line 6, the multiplication is carried out as

$$|xR \bmod N\rangle_n |yR \bmod N\rangle_n |0\rangle_n |0\rangle_m$$
$$\to |xR \bmod N\rangle_n |yR \bmod N\rangle_n |xyR \bmod N\rangle_n |g\rangle_m \quad (48)$$

where $m = \lceil \log n \rceil$ and $|g\rangle$ is a garbage register. This garbage state is erased only when $y_0 y_1 \ldots y_{M-1} R \bmod N$ is obtained. The other garbage states are retained because all values except for the obtained value $y_0 y_1 \ldots y_M R \bmod N$ are erased by reverse computation. This implementation then requires the following procedure:

1) *Multiplication*;
2) *Reduction*;
3) "Inv n-QFT" in *Uncomputation*.

Moreover, to realize the previous computation, we cannot adopt qubit recycling [32] on a control qubit for controlled Montgomery multiplications. Therefore, we must prepare control qubits for controlled Montgomery multiplications.

We now discuss how to realize parallelizing Montgomery multiplications. Parallelizing Montgomery multiplications changes "Mult" in *Multiplication*. In this procedure, phase gates with two control qubits were employed using the following qubits in Montgomery multiplication (48):

1) control qubit from the first register;
2) control qubit from the second register;
3) target qubit from the third register.

These combinations of these three qubits were run, and the number of combinations is n^3 in "Mult" in a *Multiplication* procedure. Therefore, n^3 phase gates with two control qubits are used in one controlled Montgomery multiplication. To run these gates simultaneously, $2n^3$ ancilla qubits are required as follows:

1) n copies of each AND results on n^2 pairs of two control qubits;
2) n^2 copies of n target qubits.

Moreover, n^2 copies of n target qubits are realized by $\log(n^2)$ CNOT-depth. To realize the remaining states, a similar method to that of storing $|p_{j,k}\rangle$ in Section III-D1 is adopted. In more detail, we adopt AND gates given in Fig. 5, run one-qubit gates on the third qubit simultaneously, and run CNOT gates with $O(\log n)$ depth with $O(n^2)$ ancilla qubits.

Next, the computational cost of the above implementation is evaluated. To calculate the computational cost, the values of ε_q and ε_a must be clarified. In a controlled Montgomery multiplication, the number of controlled phase gates corresponding is n^2 in each qubit. Therefore, $M_p = n^2$. Moreover, this implementation requires $1.5n^4$ qubits, because of the following.

1) In each multiplication, $2n^3$ qubits are prepared.
2) At most, $0.75n$ multiplications are run simultaneously.

This implementation runs $1.5n$ Montgomery multiplications with overhead from running as (48). In each Montgomery multiplication, $O(n^2)$ T gates are required for copying AND results of control qubits, but this is negligible compared with n^3 controlled phase gates. Therefore, the number of controlled phase gates is $3n^4$, and this means $M_f = 3n^4$. By substituting these values M_p and M_f on (33), the probability of obtaining the desired quantum states is

$$1 - O\left(n^4\varepsilon_a^2 + n^2\varepsilon_q + n^4\varepsilon_a + n^2\log\left(1/\varepsilon_q\right)\varepsilon_a\right). \quad (49)$$

Thus, when $\varepsilon_q = \Theta\left(n^{-2}\right)$ and $\varepsilon_a = \Theta\left(n^{-4}\right)$, the probability of obtaining the desired quantum states is $\Omega(1)$. Based on the previous discussion, the T-count is

$$1.5n \times 2 \times n^3 \times 3\log\left(1/\varepsilon_a\right) = 36n^4\log n. \quad (50)$$

Each term, from left to right, represents the following.

1) $1.5n$: The number of Montgomery multiplications.
2) 2: Forward calculation and reverse computation for erasing ancilla qubits.
3) n^3: Dominant term of the number of P_j gates in each Montgomery multiplication.
4) $3\log(1/\varepsilon_a)$: The number of T gates for each P_j gate.

Moreover, Montgomery multiplications in this implementation require $33\log(1/\varepsilon_a)$ T-depth, which is obtained from Table 6 excluding "Inv mult." Therefore, the T-depth is

$$\log n \times 2 \times 33\log\left(1/\varepsilon_a\right) = 264\left(\log n\right)^2. \quad (51)$$

C. SHOR'S ALGORITHM MINIMIZING KQ

Next, the KQ optimized quantum circuit is evaluated. In the earlier sections, the T-depth was minimized. These constructions realize smaller a T-depth by using more ancilla qubits. These constructions run many T gates simultaneously. Therefore, circuits with a smaller T-depth require more ancilla qubits for magic states of the T gate [34], [35]. To realize an efficient circuit, we should consider the above tradeoff between the T-depth and the number of ancilla qubits. KQ [24] is an index considering both circuit depth and the number of ancilla qubits. Thus, an efficient construction based on KQ is proposed.

TABLE 7. Comparison of KQ$_T$ on Each Proposed Construction: This Table Only Shows the Dominant Term

Section	#(Qubits)	T-depth	KQ$_T$
Section IV-A	$n\left(c_g+3\right)$	$27n^2\log n$	$27\left(c_g+3\right)n^3\log n$
Section IV-B1	$n^2\left(c_g+3\right)$	$324n\log n$	$324\left(c_g+3\right)n^3\log n$
Section IV-B2	$0.75n^4\left(c_g+3\right)$	$264\left(\log n\right)^2$	$198\left(c_g+3\right)n^4\left(\log n\right)^2$

Before this quantum circuit is evaluated, Oonishi et al. [23] method for optimizing KQ is introduced. Draper et al. [16] decreased KQ on a controlled modular adder using Draper et al. [16] adder. Their method decreases KQ by considering the distillation of T gates [19]. Let n_T be the maximal number of T gates running simultaneously, and let $c_g + 1$ be the number of required qubits for a T gate. Note that c_g is the number of qubits for a distillation circuit, and $+1$ is used for an S gate correcting phase. Based on these parameters, Oonishi et al. [23] method calculates KQ$_T$, namely, #(Qubits) $\times$ (T-depth). Their method then minimizes the KQ$_T$. Similar to their method, KQ$_T$ is now calculated and minimized.

We now calculate n_T realizing the smallest KQ$_T$. Based on the previous discussion, Table 7 shows KQ$_T$ in each proposed construction. As Table 7 shows, KQ$_T$ increases as the T-depth decreases. This relationship is attributed to the fact that the T-depth decreases only a part of circuit when the number of qubits increases. The implementations other than the implementation in Section IV-B2 are now discussed because the implementation in Section IV-B2 has a larger KQ$_T$ than the other implementations in Table 7. In these implementations, n_T is at most n^2, and $n \le n_T \le n^2$ is considered. It is assumed that we use $2n_T$ ancilla qubits for copying control and target qubits, and KQ$_T$ is given as follows.

1) *$n\log n \le n_T \le n^2$:* In these n_T, the number of qubits is $2n + (c_g+3)n_T$ and T-depth is

$$\frac{27n^3\log n}{n_T} + 297n\log n. \quad (52)$$

In (52), the first term, namely $27n^3\log n/n_T$, is T-depth of "mult" in *Multiplication*. From Table 6, the T-depth without multiplication is $108\log n - 9\log n = 99\log n$ in each Montgomery multiplication. The Montgomery multiplication is run $3n$ times, and T-depth of "mult" except for *Multiplication* is $297n\log n$, which is the second term. Therefore, T-depth is given as (52) and

$$\mathrm{KQ}_T = \left(2n + \left(c_g+3\right)n_T\right)\left(\frac{27n^3\log n}{n_T} + 297n\log n\right) \quad (53)$$

$$= \left(297\left(c_g+3\right)n\log n\right)n_T + \frac{54n^4\log n}{n_T} + C(n) \quad (54)$$

where $C(n)$ is a function independent from n_T. The value in (54) is minimized when

$$n_T = \sqrt{\frac{2n^3}{11\left(c_g+3\right)}}. \tag{55}$$

KQ_T is then

$$\mathrm{KQ}_T = 27\left(c_g+3\right)n^3\log n + 54\sqrt{22\left(c_g+3\right)}n^{2.5}\log n + 594n^2\log n \tag{56}$$

$$= 27\left(c_g+3\right)n^3\log n + O\left(n^{2.5}\log n\right). \tag{57}$$

2) $n_T \le n\log n$: The focus is now on the KQ_T of "Mult" in *Multiplication*. In this part, KQ_T is

$$\left(2n+\left(c_g+3\right)n_T\right)\frac{27n^3\log n}{n_T}$$

$$= 27\left(c_g+3\right)n^3\log n + \frac{54n^4\log n}{n_T} \tag{58}$$

$$= 27\left(c_g+3\right)n^3\log n + \Omega\left(n^3\right). \tag{59}$$

Therefore, KQ_T is larger than the value in (57) when $n_T \le n\log n$.

In conclusion, KQ_T takes the smallest value when $n_T = \sqrt{\frac{2n^3}{11(c_g+3)}}$, and the dominant term of KQ_T is $27(c_g+3)n^3\log n$.

V. CONCLUSION AND FUTURE WORKS

We examined Shor's algorithm using the Fourier basis by improving Rines and Chuang [18] implementation of the Montgomery multiplication. The contributions of this study are as follows.

1) Montgomery multiplication [18] was improved (see Section III).
2) The proposed Montgomery multiplication was applied to Shor's algorithm (see Section IV)

First, Rines and Chuang [18] implementation of the Montgomery multiplication was improved, as discussed in Section III. Their quantum circuit was improved so that the dominant term is $2n$, almost the same number of qubits as in Rines and Chuang [18] construction. Their Montgomery multiplication for Shor's algorithm was modified because the original implementation did not consider the situation without changing the value. Moreover, a method was proposed for reducing the computational cost by two approximate QFTs (a naive method and the proposed method) based on the rigorous analysis on approximation errors. By applying the naive approximate QFT, the implementation requires only one-third the number of T gates of Rines and Chuang [18] original implementation. Moreover, the implementation requires only one-fourth of the T-depth of Rines and Chuang [18] original implementation when the proposed approximate QFT is applied.

Next, the proposed Montgomery multiplication was applied to Shor's algorithm in Section IV. First, the proposed Montgomery multiplication is applied directly. Next, a small T-depth circuit was proposed by adopting Cleve and Watrous [27] method. Moreover, as in Oonishi et al. [23] method, a T-scheduling method for minimizing KQ [24], the product of the number of qubits and depth, was proposed. The construction was then given with the smallest KQ, and we obtained the smallest KQ when we run $\Theta\left(n^{1.5}\right)$ T gates simultaneously.

We now discuss future works. First, constants ε_q and ε_a were adopted in this article. However, nonconstant ε_q and ε_a values can lead to more-efficient construction. Therefore, the appropriate values of ε_q and ε_a in each procedure should be considered. Moreover, Assumption 1 must be evaluated in more detail.

Next, the quantum computer architecture should be investigated. In this study, it was assumed that all qubits are fully connected, but actual quantum computers may not be fully connected. Therefore, the computational cost on specific structures of quantum computers as in previous research [36], [37] should be considered.

Moreover, the appropriate costs for phase gates should be examined. Ross and Selinger [20] method for decomposing a phase gate into Clifford+T gates was adopted. The optimal decomposition for the proposed method requires further investigation. In addition, the optimal distillation and error correction for the proposed method should be determined.

Finally, this study only focused on the Fourier-basis Shor's algorithm, but there are many different constructions. Therefore, it is necessary to clarify the best one based on the result proposed in this article.

DISCLOSURE OF CONFLICTS OF INTEREST

We have a conflict of interest with Rodney Van Meter who is Editor-in-Chief, because we conducted joint research until 2022.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Foundations Comput. Sci.*, 1994, pp. 124–134, doi: 10.1109/SFCS.1994.365700.

[2] A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse, and J. M. Pollard, "The number field sieve," in *Proc. 22nd Annu. ACM Symp. Theory Comput.*, 1990, pp. 564–572. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/100216.100295

[3] 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.

[4] N. Koblitz, "Elliptic curve cryptosystems," *Math. Computation*, vol. 48, no. 177, pp. 203–209, 1987, doi: 10.2307/2007884.

[5] V. S. Miller, *Use of Elliptic Curves in Cryptography*. Berlin, Germany: Springer, 1986, doi: 10.1007/3-540-39799-X_31.

[6] 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.

[7] "What reaching 20 qubits means for quantum computing," 2023. Accessed: May 23, 2023. [Online]. Available: https://www.honeywell.com/us/en/news/2022/06/what-reaching-20-qubits-means-for-quantum-computing
[8] "IBM quantum experience," 2023. Accessed: May 23, 2023. [Online]. Available: https://quantum-computing.ibm.com/
[9] "IonQ | trapped ion quantum computing," 2023. Accessed: May 23, 2023. [Online]. Available: https://ionq.com/
[10] "Quantum computing | Rigetti computing," 2023. Accessed: May 23, 2023. [Online]. Available: https://www.rigetti.com/
[11] "Xanadu | Borealis," 2023. Accessed: May 23, 2023. [Online]. Available: https://xanadu.ai/products/borealis/
[12] "Charting the course to 100,000 qubits | IBM research blog," 2023. Accessed: May 23, 2023. [Online]. Available: https://research.ibm.com/blog/100k-qubit-supercomputer
[13] L. Xiao, D. Qiu, L. Luo, and P. Mateus, "Distributed Shor's algorithm," *Quantum Inf. Computation*, vol. 23, no. 1/2, pp. 27–44, 2023, doi: 10.48550/arXiv.2207.05976.
[14] 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*, doi: 10.48550/arXiv.quant-ph/0410184.
[15] T. G. Draper, "Addition on a quantum computer," 2000, *arXiv:quant-ph/0008033*, doi: 10.48550/arXiv.quant-ph/0008033.
[16] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "Alogarithmic-depth quantum carry-lookahead adder," *Quantum Inf. Computation*, vol. 6, no. 4, pp. 351–369, 2006, doi: 10.48550/arXiv.quant-ph/0406142.
[17] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," *Quantum*, vol. 5, p. 433, 2021, doi: 10.22331/q-2021-04-15-433.
[18] R. Rines and I. Chuang, "High performance quantum modular multipliers," 2018, *arXiv:1801.01081*, doi: 10.48550/arXiv.1801.01081.
[19] C. Gidney and A. G. Fowler, "Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2\,|T\rangle$ transformation," *Quantum*, vol. 3, p. 135, 2019, doi: 10.22331/q-2019-04-30-135.
[20] N. J. Ross and P. Selinger, "Optimal ancilla-free CLIFFORD+ T approximation of z-rotations," *Quantum Inf. Computation*, vol. 15, no. 11/12, pp. 932–950, 2015, doi: 10.48550/arXiv.1403.2975.
[21] A. Barenco, A. Ekert, K.-A. Suominen, and P. TÖrmä, "Approximate quantum Fourier transform and decoherence," *Phys. Rev. A*, vol. 54, no. 1, 1996, Art. no. 139, doi: 10.1103/PhysRevA.54.139.
[22] D. Coppersmith, "An approximate Fourier transform useful in quantum factoring," 2002, *arXiv:quant-ph/0201067*, doi: 10.48550/arXiv.quant-ph/0201067.
[23] K. Oonishi, T. Tanaka, S. Uno, T. Satoh, R. V. Meter, and N. Kunihiro, "Efficient construction of a control modular adder on a carry-lookahead adder using relative-phase Toffoli gates," *IEEE Trans. Quantum Eng.*, vol. 3, 2021, Art. no. 3100518, doi: 10.1109/TQE.2021.3136195.
[24] 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.
[25] 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.
[26] Y. Nam, Y. Su, and D. Maslov, "Approximate quantum Fourier transform with $O(n\log(n))$ T gates," *NPJ Quantum Inf.*, vol. 6, no. 1, 2020, Art. no. 26, doi: 10.1038/s41534-020-0257-5.
[27] R. Cleve and J. Watrous, "Fast parallel circuits for the quantum Fourier transform," in *Proc. 41st Annu. Symp. Foundations Comput. Sci.*, 2000, pp. 526–536, doi: 10.1109/SFCS.2000.892140.
[28] C. Gidney, "Halving the cost of quantum addition," *Quantum*, vol. 2, p. 74, 2018, doi: 10.22331/q-2018-06-18-74.
[29] P. Selinger, "Efficient Clifford+ T approximation of single-qubit operators," *Quantum Inf. Computation*, vol. 15, no. 1/2, pp. 159–180, 2015, doi: 10.26421/qic15.1-2-10.
[30] M. Ekerå and J. Håstad, "Quantum algorithms for computing short discrete logarithms and factoring RSA integers," in *Proc. Int. Workshop Post-Quantum Cryptogr.*, 2017, pp. 347–363, doi: 10.1007/978-3-319-59879-6_20.
[31] H. Thapliyal, E. Muñoz-Coreas, and V. Khalus, "Quantum circuit designs of carry lookahead adder optimized for T-count T-depth and qubits," *Sustain. Comput.: Inform. Syst.*, vol. 29, 2021, Art. no. 100457, doi: 10.1016/j.suscom.2020.100457.
[32] 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," *Nature Photon.*, vol. 6, no. 11, pp. 773–776, 2012, doi: 10.1038/nphoton.2012.259.
[33] R. V. 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.
[34] A. M. Steane, "Space, time, parallelism and noise requirements for reliable quantum computing," *Fortschritte der Physik: Prog. 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.
[35] R. V. 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.
[36] B.-S. Choi and R. V. Meter, "A $\theta(\sqrt{n})$-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.
[37] A. Fowler, S. Devitt, and L. Hollenberg, "Implementation of Shor's algorithm on a linear nearest neighbour qubit array," *Quantum Inf. Computation*, vol. 4, no. 4, pp. 237–251, 2004, doi: 10.48550/arXiv.quant-ph/0402196.

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.

He has been a Researcher with Mitsubishi Electric Corporation, Kamakura, Japan, since 2021. His research interests include cryptography, quantum computation, and artificial intelligence.

Noboru Kunihiro (Member, IEEE) 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.