Experiments and Resource Analysis of Shor's Factorization Using a Quantum Simulator

Junpei Yamaguchi[1], Masafumi Yamazaki[1], Akihiro Tabuchi[1], Takumi Honda[1], Tetsuya Izu[1], and Noboru Kunihiro[2]

[1] Fujitsu Limited, Japan
{j-yamaguchi,izu}@fujitsu.com
[2] University of Tsukuba, Japan

Abstract. Shor's algorithm on actual quantum computers has succeeded only in factoring small composite numbers such as 15 and 21, and simplified quantum circuits to factor the specific integers are used in these experiments. In this paper, we factor 96 RSA-type composite numbers up to 9-bit using a quantum computer simulator. The largest composite number $N = 511$ was factored in approximately 2 hours on the simulator. In our experiments, we implement Shor's algorithm with basic circuit construction, which does not require complex tricks to reduce the number of qubits, and we give some improvements to reduce the number of gates, including MIX-ADD method. This is a flexible method for selecting the optimal ADD circuit which minimizes the number of gates from the existing ADD circuits for each of the many ADD circuits required in Shor's algorithm. Based on our experiments, we estimate the resources required to factor 2048-bit integers. We estimate that the Shor's basic circuit requires 2.19×10^{12} gates and 1.76×10^{12} depth when 10241 qubits are available, and 2.37×10^{14} gates and 2.00×10^{14} depth when 8194 qubits are available.

Keywords: Shor's algorithm · integer factorization · quantum computer · quantum computer simulator.

1 Introduction

The security of RSA, one of the standardized public key cryptosystems, is based on the difficulty of the integer factorization problem of large composite numbers. The current factorization record by a classical computer is the factorization of an 829-bit integer [6], so that RSA with larger than 2048-bit integer is considered to be secure for the time being. On the other hand, it is known that the integer factorization problem can be solved in polynomial time by Shor's algorithm by using an ideal quantum computer [18]. Some factorization experiments on quantum computers by using Shor's algorithm have been reported only for $N = 15$ and 21 [1,13,14,15,16,17] because of the difficulty of realizing ideal quantum computers – quantum computers free from the limitation of the number of quantum bits (qubits) and the noise on the qubits [3]. To make matters worse, these experiments used the simplified Shor's circuits in which qubits and gates are reduced as much as possible by using the properties of the integers to be factored and their factors to be found. Since such experiments do not lead to accurate quantum resource estimation, the implementation of Shor's algorithm which can factor general composite numbers and its resource evaluation based on factoring experiments are required.

Various quantum circuits of Shor's algorithm for general composite numbers have been proposed. Kunihiro summarized basic circuits [12], which use $2n$ controlling qubits as a 1st qubit sequence for an n-bit composite number. On the other hand, advanced circuits have also been proposed [4,8]. These circuits use a technique to reduce the qubits of the 1st sequence from $2n$ to 1, which requires a complex quantum operation, repeatedly performing observations and quantum gate operations depending on the observation results.

Despite some efforts to estimate circuit resources for factoring 2048-bit integers with noisy qubits [8,9], it is too difficult to give exact estimates for factoring such large integers.

There are two major problems to break the situation. The first problem is the lack of computational resources, specifically, the number of qubits available on quantum computers. Although IBM has recently developed a 433-qubit processor [10], because of the effects of the noise, it is still difficult to process Shor's algorithm even on such computers. The second problem to be solved is the lack of experimental results for Shor's algorithm. To estimate the circuit resources for factoring 2048-bit integers, more experimental results on the same computing environments are needed.

Contribution of This Paper

This paper has three contributions. First, we implemented the basic circuits of Shor's algorithm applicable to general composite numbers, and succeeded in factoring 96 RSA-type composite numbers up to 9-bit on a quantum computer simulator running on a supercomputer. The largest composite number $N = 511$ was factorized in 2 hours on the simulator. We used the simulator *mpiQulacs* developed by Fujitsu [11], a State Vector (SV) type simulator that records all qubit states in memory with no noise and allows to simulate an ideal quantum computation [11]. This paper focuses on the basic circuits because the current large scale quantum simulator cannot handle the advanced circuit due to its complexity. Our implementations are based on the well-known techniques [4,12], but we provide some bug-fixes, improvements (including the second contribution) and comparisons of required resource.

Second, we propose a flexible ADD method, MIX-ADD, to reduce the elementary gates and the depth of the basic circuit. The dominant circuit in Shor

[3] Very recently, Yan et al. proposed a new quantum factoring algorithm which requires a fewer number of qubits [21] and gave a new estimation for factoring 2048-bit integers. However, the validity of the algorithm and the correctness of the estimation are under the analysis.

is Mod-EXP which computes a modular exponentiation $f_{a,N}(x) = a^x \bmod N$. Mod-EXP can be constructed from ADD circuits, and there are three well-known ADD circuits: R-ADD, GT-ADD, and Q-ADD [12]. The basic circuit requires $5n + 1$ qubits for R-ADD, and $4n + 2$ qubits for GT/Q/MIX-ADD for n-bit composite numbers to be factored. MIX-ADD reduces the gates and the depth by selecting the optimal ADD circuit which minimizes the number of gates from R/GT/Q-ADD for each ADD circuit called multiple times in Mod-EXP. Our analysis shows that R/MIX/Q/GT-ADD require more gates in this order for larger n. MIX-ADD can factor larger composite numbers more efficiently in an environment where the number of available qubits is limited like the present.

Finally, we gave estimations of the number of gates and the depth for the Shor's basic circuits. We generated some quantum circuits for $n = 8, \ldots, 24$, and evaluated the resources of the circuit. Based on these data, we estimated the circuit resources required to factor 2048-bit integers. In our estimation, the basic circuit requires 10241 qubits, 2.19×10^{12} gates and 1.76×10^{12} depth for R-ADD, and 8194 qubits, 2.37×10^{14} gates and 2.00×10^{14} depth for MIX-ADD.

The rest of the paper is organized as follows: Section 2 describes the construction of Shor's quantum circuit, in particular the modular exponentiation circuit Mod-EXP using ADD circuits. Then, in Section 3, concrete constructions of Mod-EXP from R-ADD, GT-ADD, Q-ADD and MIX-ADD are explained. Section 4 summarizes factoring experiments by Shor's quantum circuit using the quantum computer simulator including the estimation for 2048-bit integers.

2 Quantum Circuit of Shor's algorithm

This section describes quantum circuits of Shor's algorithm for general composite numbers based on known techniques [4,12]. In this paper, we consider the following quantum gates as the elementary gates for evaluating the number of gates and the depth: 1-qubit gates including the Hadamard gate, NOT gate, the rotation gate and the phase-shift gate, Controlled NOT (C-NOT) gate, and Toffoli (C^2-NOT) gate.

2.1 Shor's algorithm and Factorization

Suppose we want to factor an n-bit composite number N. For an integer a coprime to N, the order of a with regard to N is defined as the smallest positive integer r such that $a^r \equiv 1 \bmod N$. In 1994, Shor proposed a quantum algorithm to compute the order r of a with regard to N in polynomial time [18]. The integer N can be factored by using Shor's algorithm in the following way:

i) Choose an integer a from $\{2, \ldots, N - 1\}$. If $\gcd(a, N) \neq 1$ then output $\gcd(a, N)$ and terminate (since a factor of N larger than 1 is found).
ii) Compute the order r from a and N by quantum order finding algorithm.
iii) If r is even, $a^{r/2} + 1 \not\equiv 0 \bmod N$ and $\gcd(a^{r/2} \pm 1, N) \neq 1$, output $\gcd(a^{r/2} \pm 1, N)$ and terminates. Otherwise, return step i).

Note that step i) and iii) can be proceeded by classical computers. On the other hand, step ii) can be computed by the quantum order finding algorithm on a quantum computer in the following way:

1. Generate an initial state $|\phi_0\rangle = |0\dots0\rangle\,|0\dots01\rangle$, where the 1st qubit sequence has m qubits, while the 2nd qubit sequence has n qubits.
2. Apply the Hadamard operation H_m to the 1st sequence:
$$|\phi_1\rangle = H_m(|\phi_0\rangle) = \frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x\rangle\,|0\dots01\rangle\,.$$
3. Apply the operation $U_{f_{a,N}}$ which corresponds to a modular exponentiation $f_{a,N}(x) = a^x \bmod N$, to the 2nd sequence:
$$|\phi_2\rangle = U_{f_{a,N}}(|\phi_1\rangle) = \frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x\rangle\,|f_{a,N}(x)\rangle\,.$$
4. Apply the Inverse Quantum Fourier Transform to the 1st sequence.
5. Observe the 1st sequence, an approximation of a multiple of $2^m/r$ is obtained.
6. Repeat Step 1–5 until r can be estimated.

The parameter m is determined from the approximation precision in Step 5, $m = 2n$ is used usually and in this paper.

2.2 Construction of Mod-EXP from ADD

Above steps except Step 3 can be easily realized by elementary gates. On the other hand, Step 3 requires complex circuits called Mod-EXP [12]. This subsection describes how to realize Mod-EXP from elementary gates. In fact, Mod-EXP can be constructed from ADD circuits, by transforming Mod-EXP to the following circuits step-by-step:

– Mod-EXP(a) : $|x\rangle\,|1\rangle \to |x\rangle\,|a^x \bmod N\rangle$
– Mod-MUL(d) : $|y\rangle \to |dy \bmod N\rangle$
– Mod-PS(d) : $|y\rangle\,|t\rangle \to |y\rangle\,|t + dy \bmod N\rangle$
– Mod-ADD(d) : $|y\rangle \to |y + d \bmod N\rangle$
– ADD(d) : $|y\rangle \to |y + d\rangle$

Construction of Mod-EXP from Mod-MUL For an exponent x represented in binary, namely, $x = \sum_{i=0}^{m-1} 2^i x_i$, a modular exponentiation Mod-EXP(a) is computed by a repetition of multiplying $d_i = a^{2^i} \bmod N$ and taking modulus by N, since $a^x \bmod N = a^{\sum_{i=0}^{m-1} 2^i x_i} \bmod N = \prod_{i=0}^{m-1} a^{2^i x_i} \bmod N$. In other words, Mod-EXP(a) can be computed by a repetition of the modular multiplication Mod-MUL(d_i) controlled by $|x_i\rangle$, so that Mod-EXP(a) requires m controlled-Mod-MULs, which is denoted by $C(x_i)$-Mod-MUL.

Construction of Mod-MUL from Mod-PS The modular multiplication Mod-MUL(d) for an integer $0 \leq d < N$ and an n-bit integer y can be computed by using modular product-sums Mod-PSs in the following way:

$$\begin{aligned}
|y\rangle\,|\underbrace{0\ldots0}_{n}\rangle &\overset{\text{Mod-PS}(d)}{\to} |y\rangle\,|0+dy \bmod N\rangle \overset{\text{SWAP}}{\to} |dy \bmod N\rangle\,|y\rangle \\
&\overset{\text{Mod-PS}(-d^{-1})}{\to} |dy \bmod N\rangle\,|y+(-d^{-1})(dy \bmod N) \bmod N\rangle \\
&= |dy \bmod N\rangle\,|0\rangle\,.
\end{aligned}$$

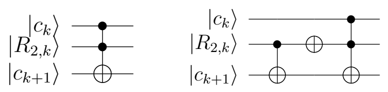
Since $d_i = a^{2^i} \bmod N$ and $\gcd(a, N) = 1$, there always exists the inverse $d^{-1} \bmod N$. Thus, Mod-MUL can be computed by two Mod-PSs and one n-qubit SWAP with n auxiliary qubits $|R_2\rangle = |0\ldots0\rangle$. Especially, $C(x_i)$-Mod-MUL requires two $C(x_i)$-Mod-PSs and one n-qubit C-SWAP. Moreover, an n-qubit C-SWAP can be realized by n 1-qubit C-SWAPs, and one 1-qubit C-SWAP can be realized by two C-NOTs and one Toffoli gate.

Construction of Mod-PS from Mod-ADD When the 2nd sequence is represented as $|y\rangle = |y_{n-1}\ldots y_0\rangle$, for an integer $0 \leq d < N$, we have $dy = \sum_{j=0}^{n-1} d2^j y_j$. Thus, a modular product-sum Mod-PS(d) on a bit sequence $|R_2\rangle$ can be computed by a repetition $R_2 \leftarrow R_2 + d2^j \bmod N$ if $y_j = 1$ for $j = 0, 1, \ldots, n-1$, which is equivalent to $C(y_j)$-Mod-ADD($d2^j \bmod N$). That is, Mod-PS can be realized by n 1-controlled Mod-ADDs, and $C(x_i)$-Mod-PS can be realized by n 2-controlled Mod-ADDs, namely, $C(x_i, y_j)$-Mod-ADDs.

Construction of Mod-ADD from ADD There are two constructions, Type 1 and Type 2 for realizing $C(x_i, y_j)$-Mod-ADD [12]. From the efficiency point of view, Type 2 is optimal for R-ADD and Q-ADD, while Type 1 for GT-ADD. Due to space limitation, we omit describing the details.

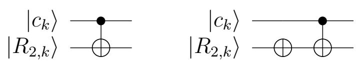
2.3 Construction of ADD

This subsection describes how to construct ADD circuits from the elementary gates in three ways: R-ADD, GT-ADD, and Q-ADD. Here, we consider the circuit to add an n-bit integer $p = p_{n-1}\ldots p_0$ to an n-qubit register $|R_2\rangle = |R_{2,n-1}\ldots R_{2,0}\rangle$. Considering the carry-over, the result is represented by $|R_1R_2\rangle$ with 1-qubit register $|R_1\rangle$. All ADD circuits use another 1-qubit register $|R_3\rangle$, and R-ADD uses further $(n-1)$-qubit sequence $|c\rangle$. In total, GT-ADD and Q-ADD require $m + n + 1 + n + 1 = m + 2n + 2 = 4n + 2$ qubits, while R-ADD requires $m + 2n + 2 + (n-1) = m + 3n + 1 = 5n + 1$ qubits. On the other hand, the number of elementary gates is estimated by $270n^3$ for R-ADD, $16/3n^5$ for GT-ADD, and $97n^4$ for Q-ADD [12].



(a) $p_k = 0$ (b) $p_k = 1$

Fig. 1: CARRY Circuit



(a) $p_k = 0$ (b) $p_k = 1$

Fig. 2: SUM Circuit

Construction of R-ADD R-ADD is a ripple carry adder [5,20], which computes $R_2 + p$ by using the following addition table:

$$\begin{array}{rccccc} & c_{n-1} & c_{n-2} & \cdots & c_1 & \\ & R_{2,n-1} & R_{2,n-2} & \cdots & R_{2,1} & R_{2,0} \\ +) & p_{n-1} & p_{n-2} & \cdots & p_1 & p_0 \\ \hline R_1 & R_{2,n-1} & R_{2,n-2} & \cdots & R_{2,1} & R_{2,0} \end{array}$$

Here, $c = c_{n-1} \dots c_1$ is an auxiliary $(n-1)$-bit register with initial value 0, which is used for storing carry-overs. R-ADD consists of three circuits, CARRY (for computing carry bits), SUM (for computing additions), and CARRY^{-1} (for resetting carry bits). As in Figure 2 of Vedral, Barenco and Ekert's paper [20], R-ADD firstly computes all carry-overs by using CARRY circuit described in Figure 1 for $k = 0, 1, \dots, n-1$ (set $c_n = R_1$ when $k = n-1$). When $p_{n-1} = 1$, apply the NOT gate to $R_{2,n-1}$. Finally, for $k = n-1, \dots, 0$, update $R_{2,k}$ by using SUM circuit described in Figure 2 and reset c_k by using CARRY^{-1} circuit, which is reverse circuit of CARRY. When $k = 0$, CARRY^{-1} is omitted. Thus, R-ADD can be constructed from Toffoli gates, C-NOT gates, and NOT gates.

Type 2 Mod-ADD requires 1-controlled R-ADD and 2-controlled R-ADD, which require not only C-NOT gate and Toffoli gate, but 3-controlled NOT and 4-controlled NOT gates. Barenco et al. showed two conversions from a C^k-NOT gate to Toffoli gates [3]. The first conversion converts a C^k-NOT gate to $2k-3$ Toffoli gates by using $k-2$ clean auxiliary qubits (qubits with their state known to be $|0\rangle$). The second converts a C^k-NOT gate to $4k-8$ Toffoli gates by using $k-2$ dirty (unclean) auxiliary qubits. Both auxiliary qubits return to their initial state after the usage. According to Kunihiro's paper [12], the first conversion is used for all C^k-NOT gates.

Construction of GT-ADD For $k = 0, 1, \dots, n-1$, GT-ADD adds p by repeatedly computing $R_2 \leftarrow R_2 + 2^k$ when $p_k = 1$. An addition by 2^k can be realized by C^ℓ-NOT gates ($1 \le \ell \le n-k$) and one NOT gate as in Figure 3. Type 1 Mod-ADD requires, in addition to GT-ADD, 1/2/3-controlled GT-ADD, which consists of NOT gates, C-NOT gates, Toffoli gates, and C$^\ell$-NOT gates ($3 \le \ell \le n+3$). Both conversions described in Section 2.3 can be used in GT-ADD, however, since it is difficult to allocate clean qubits, Kunihiro used the second conversion for all C$^\ell$-NOT gates [12].

Construction of Q-ADD Q-ADD is an adder using the Quantum Fourier Transform (QFT) [4,7]. For simplicity, we set $|R_{2,n}\rangle := |R_1\rangle$ and assume that

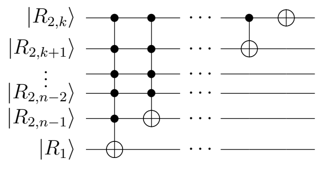


Fig. 3: Adder 2^k to $|R_2\rangle$

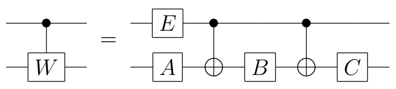


Fig. 4: Conversion of 1-controlled R_k

$|R_2\rangle$ has $n+1$ qubits in this subsection. Also set $p_n = 0$. Unlike R/GT-ADD, Q-ADD computes $|R_2\rangle \leftarrow |R_2 + p \bmod 2^{n+1}\rangle$. Denote the state after applying QFT to the register $|R_2\rangle$ (Figure 9 in [4]) as $\phi(|R_2\rangle)$. Then, Q-ADD computes in the following way: for $j = n, n-1, \dots, 0$, and for $k = 1, 2, \dots, j+1$, apply the Z-rotation gate $R_k = (1, 0; 0, e^{2\pi i/2^k})$ to $\phi(|R_{2,j}\rangle)$ when $p_{j-k+1} = 1$. Inverse QFT (QFT^{-1}) is required to obtain the result of the addition. Thus, Q-ADD can be realized by rotation gates except QFT/QFT^{-1}.

Type 2 Mod-Add requires 1/2-controlled Q-ADDs, that is, 1/2-controlled R_k gates are required. Here, 1-controlled R_k gate can be realized by 2 C-NOTs and 4 1-qubit gates, and 2-controlled R_k gate can be realized by 6 C-NOTs and 8 1-qubit gates [3].

Construction of 1/2-controlled R_k is as follows. Arbitrary unitary matrix W can be represented by

$$W = \Phi(\delta) Rz(\alpha) Ry(\theta) Rz(\beta) \tag{1}$$

for parameters $\alpha, \beta, \theta, \delta \in [0, 2\pi]$, where

$$\Phi(x) = \begin{pmatrix} e^{ix} & 0 \\ 0 & e^{ix} \end{pmatrix}, \ Ry(x) = \begin{pmatrix} \cos x/2 & \sin x/2 \\ -\sin x/2 & \cos x/2 \end{pmatrix}, \ Rz(x) = \begin{pmatrix} e^{ix/2} & 0 \\ 0 & e^{-ix/2} \end{pmatrix}.$$

Then 1-controlled W gate can be represented as in Figure 4, where

$$A = Rz(\alpha) Ry(\theta/2), B = Ry(-\theta/2) Rz(-(\alpha+\beta)/2),$$
$$C = Rz((\beta-\alpha)/2), E = Rz(-\delta)\Phi(\delta/2).$$

Thus, 1-controlled R_k can be realized by 2 C-NOTs and 4 1-qubit gates as in Figure 4 by determining parameters $\alpha, \beta, \theta, \delta$. Similarly, 2-controlled R_k gate can be realized by 6 C-NOTs and 8 1-qubit gates from Lemma 6.1 in [3].

2.4 Required Resources

This section summarizes the resources required in Shor's circuit to factor an n-bit integer.

Shor's circuit has three main circuits, Hadamard, Mod-EXP, and QFT^{-1}. Required number of gates for each of Hadamard and QFT^{-1} is $O(n^2)$, while Mod-EXP requires $G_{\text{ModEXP}}(\text{R-ADD}) = 270n^3$ with R-ADD, $G_{\text{ModEXP}}(\text{GT-ADD}) = 16/3n^5$ with GT-ADD, and $G_{\text{ModEXP}}(\text{Q-ADD}) = 97n^4$ with Q-ADD. Therefore, required number of gates for Shor's circuit can be identified by these numbers: $G_{\text{Shor}}(\text{R-ADD}) = 270n^3$, $G_{\text{Shor}}(\text{GT-ADD}) = 16/3n^5$, and $G_{\text{Shor}}(\text{Q-ADD}) = 97n^4$. Unfortunately, no estimation for the depth is known. Required numbers of qubits are $Q_{\text{Shor}}(\text{R-ADD}) = 5n + 1$ with R-ADD, and $Q_{\text{Shor}}(\text{GT-ADD}) = Q_{\text{Shor}}(\text{Q-ADD}) = 4n + 2$ with GT-ADD and Q-ADD.

3 Implementation of Shor's Quantum Circuit

This section describes how to implement Mod-EXP with each of R-ADD, GT-ADD, and Q-ADD, respectively, based on Kunihiro's paper [12]. We also show bug-fixes and improvements from them. Moreover, we propose Mod-EXP with MIX-ADD method. This requires $4n+2$ qubits same as the case of GT/Q-ADD, but consists of fewer gates compared with GT/Q-ADD.

3.1 Mod-EXP with R-ADD

We use Type 2 Mod-ADD in order to minimize the number of gates. We also apply the following bug-fixes and improvements.

Bug-fix on Converting C^3-NOT, C^4-NOT to Toffoli Gate The first conversion described in Section 2.3 is used in [12] for all C^k-NOT ($k = 3, 4$) gates in 1/2-controlled R-ADD, however, $k - 2$ clean qubits are not available in some cases. In such cases we propose to take the following procedures. When $k = 3$ and no clean qubit is available, then use the second conversion described in Section 2.3. When $k = 4$, use the second conversion if no qubit is available, and use the conversion described in Figure 6 if 1-qubit is available, which is given by greedy method described later. Compared to the first conversion, 1 Toffoli gate is increased when $k = 3$, and 3/1 Toffoli gates are increased when $k = 4$ with 0/1 clean qubit. Though this increases the number of gates in Mod-EXP, it does not affect the order since it is at most $O(n)$ (explained later).

Greedy Method Suppose $1 \leq c \leq k-3$ clean qubits and sufficient dirty qubits are available. Our greedy method converts a C^k-NOT to Toffoli gates as follows.

1. Generate a null circuit `circ`.
2. Let X be a set of indices of k control qubits.
3. Select two indices from X, and delete these indices from X.
4. Select one clean qubit with changing its status as 'dirty' in clean qubit management, and add its index to X.
5. Generate a Toffoli gate, controlled by selected indexed-qubits and targeted to the selected clean qubit.
6. Add the generated Toffoli gate to `circ`.
7. Repeat Step 2–6 c times.

8. Generate a C^{k-c}-NOT gate controlled by $(k-c)$ indices in X, and targeted to the same qubit as the original C^k-NOT gate, and convert to $4(k-c)-8$ Toffoli gates by using the 2nd conversion, and add to `circ`.
9. Add all Toffoli gates generated in Step 2–7 in the reversed order to `circ`.
10. Output `circ`.

The number of Toffoli gates generated by the greedy method is $c+4(k-c)-8+c = 4k-8-2c$. See Appendix 1 and 2 for examples of our greedy method.

Clean Qubits Management It is difficult to figure out which qubit is clean or not manually when C^k-NOT conversion is required. So we implemented the management function to automatically list the status of auxiliary qubits.

- When a quantum gate is added to the circuit, set the status of the target qubit of the gate as 'dirty' (not clean). If the gate makes the status clean (such as CARRY^{-1}), set 'clean'.
- Use 'clean' qubits in C^k-NOT conversion.

This management minimizes the number of gates of Mod-EXP.

The Number of Gates after Bug-fix In Step 2 of Shor's algorithm, we apply the Hadamard gate to the m-qubit sequence. By changing this operation to applying the Hadamard gate to x_i just before each $C(x_i)$-Mod-MUL, $x_{i+1},\ldots,x_{m-1}$ can be used as clean qubits in $C(x_i)$-Mod-MUL. Thus, for $i = 0,\ldots,m-3$, x_{i+1}, x_{i+2} can be used as clean qubits and there is no increase on the number of gates because the first conversion can be applied same as in the Kunihiro's paper [12]. On the other hand, when $i = m-2, m-1$, available clean qubits are less than 2, and additional circuits are required.

The number of gates for Bug-fix Mod-EXP with R-ADD is given as follows. $C(x_i, y_j)$-Mod-ADD in Mod-EXP consists of 3 C^2-R-ADDs, 1 C-R-ADD, 1 C^2-NOT, 2 C-NOTs and 3 NOTs. In the case for $i = 0,\ldots,m-3$, C^4/C^3-NOT is converted to 5/3 Toffoli gates because two clean qubits are available. Hence, $C(x_i, y_j)$-Mod-ADD consists of $135/2n - 155/2$ elementary gates. For $i = m-2, m-1$, we count the number of gates to be added from this number. For $i = m-2$, there is no additional gates in the first and last C^2-R-ADD(d) in Type 1 $C(x_i, y_j)$-Mod-ADD, because two clean qubits (x_{m-1} and R_3) are available. On the other hand, additional gates are required in C^2-R-ADD($2^n - d$) due to lack of clean qubits. Specifically, two clean qubits x_{m-1} and c_{n-1} are available in C^2-CARRY for c_j and C^2-CARRY^{-1} for c_j for $j = 1,\ldots,n-2$, but just one in C^2-CARRY for c_{n-1}, c_n and C^2-CARRY^{-1} for c_{n-1}. Each C^4-NOT gate in these three CARRYs is converted to 6 Toffoli gates by the greedy method for $k = 4, c = 1$. This leads to the addition of 3 elementary gates compared to the case for $i = 0,\ldots,m-3$. Hence, $C(x_i, y_j)$-Mod-ADD consists of $135/2n - 149/2$ elementary gates. For $i = m-1$, additional gates are required as shown in Table 1, then $C(x_i, y_j)$-Mod-ADD consists of $135/2n - 55$ elementary gates. Therefore,

	C^2-ADD(d) at Step 1-1 and 7			C-ADD at Step 3			C^2-ADD($2^n - d$) at Step 5		
	required	available	#gates	required	available	#gates	required	available	#gates
CARRY c_{n-2}	2	R_3, c_{n-1}	+0	1	c_{n-1}	+0	2	c_{n-1}	+1
CARRY c_{n-1}	2	R_3	+1	1	–	+1	2	–	+7/2
CARRY R_1	2	R_3	+1	1	–	+1	2	–	+7/2
SUM $R_{2,n-1}$	2	R_3	+0	0	–	+0	1	–	+1
CARRY^{-1} c_{n-1}	2	R_3	+1	1	–	+1	2	–	+7/2
CARRY^{-1} c_{n-2}	2	R_3, c_{n-1}	+0	1	c_{n-1}	+0	2	c_{n-1}	+1

Table 1: The number of required clean qubits, available clean qubits and the number of additional gates in each controlled ADD in Mod-ADD Type-2 [12] with R-ADD for $i = m - 1$

Bug-fix Mod-EXP with R-ADD consists of

$$\begin{aligned} G_{\text{ModEXP}}(\text{R-ADD}) &= 2n(m-2)(135/2n - 155/2) + 2n(135/2n - 149/2) \\ &\quad + 2n(135/2n - 55) + 3mn \\ &= 270n^3 - 304n^2 + 51n \end{aligned}$$

elementary gates, where $3mn$ is the number of elementary gates for C-SWAPs in Mod-MUL. The gates increased by the lack of clean qubits is at most $O(n)$.

3.2 Mod-EXP with GT-ADD

For implementing Mod-EXP with GT-ADD, Type 1 Mod-ADD is used to minimize the number of gates. Kunihiro used the second conversion described in Section 2.3 for converting C^k-NOT gates (for $3 \le k \le n+3$) to Toffoli gates. This paper proposes to use clean qubits as much as possible by the greedy method to decrease the number of gates.

Greedy Method in Mod-EXP For all conversions from C^k-NOT gates ($3 \le k \le n+3$) to Toffoli gates appeared in Mod-EXP with GT-ADD, we use the 1st conversion described in Section 2.3 when more than or equal to $k-2$ clean qubits are available, the greedy method when 1 to $k-3$ clean qubits are available, and the 2nd conversion when no clean qubit is available. We also use the clean qubit management in the greedy method.

The Number of Gates with Greedy Method The number of gates for Mod-EXP with GT-ADD with the greedy method is given as follows. Type 1 $C(x_i, y_j)$-Mod-ADD consists of the following four gates. Each gate can be converted to elementary gates as shown in Case 1–4.

1. C^3-GT-ADD with $m-i-1$ cleans,
2. 2 C^3-GT-ADDs with $m-i$ cleans,
3. C^2-GT-ADD with $m-i-1$ cleans,
4. 2 C^3-NOTs and 4 C^2-NOTs.

Case 1. C^3-GT-ADD consists of $(n-k+4)/2$ C^k-NOTs ($4 \leq k \leq n+3$) and $n/2$ C^3-NOTs on average. In the case for $0 \leq i \leq n-2$, all C^k-NOTs can be converted to Toffoli gates by the 1st conversion because $n+1$ clean qubits are available. Hence, the number of gates is given as $n_1(i) = 1/2 \sum_{k=4}^{n+3}(n-k+4)(2k-3)+3/2n$. For $n-1 \leq i \leq m-2$, we convert C^k-NOT to Toffoli gates by the 1st conversion for $3 \leq k \leq m-i+1$ and the greedy method for $m-i+2 \leq k \leq n+3$. Hence, the number of gates is $n_1(i) = 1/2 \sum_{k=m-i+2}^{n+3}(n-k+4)(4k-8-2(m-i-1)) + 1/2 \sum_{k=4}^{m-i+1}(n-k+4)(2k-3)+3/2n$. For $i = m-1$, we use the 2nd conversion, then the number of gates is $n_1(i) = 1/2 \sum_{k=4}^{n+3}(n-k+4)(4k-8)+3/2n$.

Case 2. In the same way as Case 1, the number of gates is $n_2(i) = 1/2 \sum_{k=3}^{n+2}(n-k+3)(2k-3)+n/2$ for $0 \leq i \leq n$, $n_2(i) = 1/2 \sum_{k=m-i+3}^{n+2}(n-k+3)(4k-8-2(m-i)) + 1/2 \sum_{k=3}^{m-i+2}(n-k+3)(2k-3)+n/2$ for $n+1 \leq i \leq m-1$.

Case 3. In the same way as Case 1, the number of gates is $n_3(i) = 1/2 \sum_{k=3}^{n+2}(n-k+3)(2k-3)+n/2$ for $0 \leq i \leq n-1$, $n_3(i) = 1/2 \sum_{k=m-i+2}^{n+2}(n-k+3)(4k-8-2(m-i-1)) + 1/2 \sum_{k=3}^{m-i+1}(n-k+3)(2k-3)+n/2$ for $n \leq i \leq m-2$, and $n_3(i) = 1/2 \sum_{k=3}^{n+2}(n-k+3)(4k-8)+n/2$ for $i = m-1$.

Case 4. Each C^3-NOT can be converted to 3 Toffoli gates for $0 \leq i \leq m-2$, and 4 for $i = m-1$.

Mod-EXP with GT-ADD with the greedy method consists of

$$\begin{aligned} G_{\mathrm{ModEXP}}(\text{GT-ADD}) &= 2n\left\{\sum_{i=0}^{m-1}(n_1(i)+n_2(i)+n_3(i)+4)+6(2n-1)+8\right\}+3mn \\ &= 3n^5+15n^4+\frac{51}{2}n^3+\frac{103}{2}n^2+8n \end{aligned}$$

elementary gates. The greedy method reduces the fifth-order coefficient from 16/3 to 3.

3.3 Mod-EXP with Q-ADD

This subsection describes how to implement Mod-EXP with Q-ADD.

Bug-fix in Q-ADD Since Q-ADD requires to apply QFT to the registers $|R_1R_2\rangle$, QFT just before $C(x_0)$-Mod-MUL in Q-ADD (Figure 2 in [4]), and QFT^{-1} just before C-SWAP and QFT just before C-SWAP in $C(x_i)$-Mod-MUL should be added. Thus the number of gates are increased to $4n+2$ QFTs for Mod-EXP from the original [12]. Furthermore, the original number of gates did not consider C-SWAP, so that mn Toffoli gates and $2mn$ C-NOTs should be added. However, since these increase is at most $O(n^3)$, it does not effect on the total number of Mod-EXP at all.

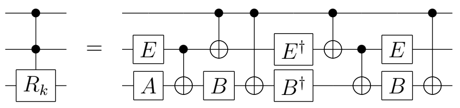


Fig. 5: Conversion of 2-controlled Rotation Gate

Change of Mod-ADD When Type 2 Mod-ADD is used for Q-ADD, 4 QFTs and 4 QFT^{-1}s are required, and the number of gates of Mod-EXP will be increased (the order is same, but the coefficient becomes larger). So, we propose to use Beauregard's Mod-ADD which requires 2 QFTs and 2 QFT^{-1}s [4].

Gate Reduction of Controlled Rotation Gate Conversion When 1/2-controlled R_k gates are converted to elementary gates, one 1-qubit gate can be reduced by setting parameters properly. In fact, set $\alpha = \beta = -\pi/2^k,\ \theta = 0,\ \delta = \pi/2^k$ in (1) for $W = R_k$, then C becomes an identity matrix and can be omitted. Similarly, setting $\alpha = \beta = -\pi/2^{k-1},\ \theta = 0,\ \delta = \pi/2^{k-1}$ for 2-controlled R_k gates reduces one 1-qubit gate as in Figure 5, where † denotes an inversion.

The Number of Gates after Bug-fix Mod-EXP consists of $2mn$ Beauregard's Mod-ADDs, $2m+2$ QFTs (or QFT^{-1}) and mn C-SWAPs. Mod-ADD also consists of as follows.

- 3 C^2-Q-ADDs, each of which consists of $(n+2-k)/2$ C^2-R_k for $1 \le k \le n+1$,
- C-Q-ADD, which consists of $(n+2-k)/2$ of C-R_k for $1 \le k \le n+1$,
- Q-ADD, which consists of $(n+2-k)/2$ of R_k for $1 \le k \le k$,
- 4 QFTs, each QFT consists of n+1 H gates and n+2–k C-R_k for $2 \le k \le n+1$,
- 2 C-NOTs and 2 NOTs.

Hence, Mod-ADD consists of $G_{\text{ModADD}}(\text{Q-ADD}) = 21n^2/4 + 47n/4 + 21/2$ elementary gates because C^2/C-R_k can be converted to 13/5 elementary gates. And QFT consists of $G_{\text{QFT}}(n+1) = 5/2n^2 + 7n/2 + 1$ elementary gates. Therefore, Mod-EXP with Q-ADD consists of

$$\begin{aligned} G_{\text{ModEXP}}(\text{Q-ADD}) &= 2mn \times G_{\text{ModADD}}(\text{Q-ADD}) + (2m+2) \times G_{\text{QFT}}(n+1) + 3mn \\ &= 85n^4 + 201n^3 + 147n^2 + 11n + 2 \end{aligned}$$

elementary gates. The fourth-order coefficient is reduced from 97 to 85 by the gate reduction.

3.4 Mod-EXP with MIX-ADD

This subsection proposes a MIX-ADD method, which uses different ADD methods in Mod-EXP depending on the number of available clean qubits to minimize the number of elementary gates.

Definition of MIX-ADD The original Mod-EXP uses just one ADD circuit such as R/GT/Q-ADD, but in the case of $4n+2$ qubits circuit, the number of gates for Mod-EXP can be reduced by selecting the optimal ADD for each ADD in Mod-EXP. We call this construction Mod-EXP with MIX-ADD. Considering the order of the number of gates for each ADD, R-ADD is top priority, next Q-ADD, then GT-ADD. However, R-ADD is available only if $n-1$ clean auxiliary qubits are available as carry qubits. In $C(x_i)$-Mod-MUL in Mod-EXP, we can use R-ADD for $0 \leq i \leq n$ because $m-i+1$ clean qubits $(x_{i+1}, \ldots, x_{m-1})$ are available. On the other hand, we use Q-ADD for $n+1 \leq i \leq m-1$ to minimize the number of gates. In applying Q-ADD from the middle of Mod-EXP, QFT is added in the following three points. The first is after $C(x_{n-2})$-Mod-MUL, the second is QFT^{-1} before C-SWAP and QFT after C-SWAP in Mod-MUL for $n+1 \leq i \leq m-1$, and the third is after $C(x_{m-1})$-Mod-MUL.

The Number of Gates The number of gates for Mod-EXP with MIX-ADD is computed in the same way as in Section 3.1 and Section 3.3. Therefore, we have

$$\begin{aligned} G_{\mathrm{ModEXP}}(\text{MIX-ADD}) &= 2n(n-1)(135/2n-155/2) + 2n(135/2n-149/2) \\ &\quad + 2n(135/2n-55) + 2n(n-1) \times G_{\mathrm{ModADD}}(\text{Q-ADD}) \\ &\quad + 2n \times G_{\mathrm{QFT}}(n+1) + 3mn \\ &= \frac{85}{2}n^4 + 193n^3 - \frac{83}{2}n^2 - 163n, \end{aligned}$$

which is about half the number of gates for Mod-EXP with Q-ADD.

4 Experimental Results

This section reports our factorization results based on our implementation described in Section 3 by using the quantum computer simulator *mpiQulacs* [11], a distributed version of the quantum simulator *Qulacs* [19]. We used an A64FX-based cluster system similar to *Todoroki* [11] with 512 nodes, which enables 39-qubit operations. A64FX is an ARM-based CPU that is also equipped in the world's top Fugaku supercomputer.

The experiments were conducted by the following steps:

1. For an n-bit RSA-type composite number (a product of two different odd primes) N, choose a which induces the factorization (for efficiency reason).
2. Generate the quantum circuit for factoring N by Shor's algorithm. Here we have four choices for ADD circuit.
3. Input the quantum circuit to the simulator.
4. Observe the 1st bit sequence 10,000 times to estimate the order r.
5. Output $\gcd(a^{r/2} \pm 1, N)$.

Note that, since the observation in Step 4 does not destroy the quantum state, it is sufficient to run each quantum circuit once in the experiments.

			R-ADD				GT-ADD				Q-ADD				MIX-ADD			
N	n	a	Q	G	D	T	Q	G	D	T	Q	G	D	T	Q	G	D	T
15	4	2	21	12937	10507	2.4	18	12595	9838	0.91	18	38967	20208	3.5	18	22815	14273	1.6
21	5	2	26	26155	20779	89.9	22	25325	18824	5.2	22	78334	40409	18	22	47273	28866	10.4
33	6	5	31	46935	36870	–	26	44461	31436	92	26	145620	76578	404	26	87251	53343	228
35	6	2	31	47662	37775	–	26	55387	38869	115	26	155329	79693	426	26	93174	55541	241
39	6	2	31	47843	38214	–	26	61941	43483	129	26	160315	81152	441	26	95233	56408	246
51	6	2	31	46991	37413	–	26	55755	39348	116	26	152468	78285	421	26	90995	54677	237
55	6	2	31	47845	38513	–	26	61899	43507	129	26	160613	80877	441	26	95368	56384	246
57	6	5	31	47555	38028	–	26	51360	36346	107	26	154085	78686	431	26	91616	55062	238
65	7	3	36	76341	59902	–	30	82676	56199	2430	30	251424	132329	10545	30	150521	90940	5915
69	7	2	36	78035	61939	–	30	98774	66690	2866	30	271832	138888	11329	30	162705	95730	6362
77	7	2	36	77066	61391	–	30	104285	70616	3033	30	267042	135177	11125	30	159450	93522	6275
85	7	2	36	75704	60041	–	30	99407	67570	2906	30	256625	132179	10719	30	153316	91241	6011
87	7	2	36	78196	62751	–	30	120027	80999	3485	30	284083	142164	11792	30	167554	97300	6524
91	7	2	36	77819	62369	–	30	116234	78729	3369	30	279204	141000	11594	30	165151	96642	6435
93	7	2	36	77659	62319	–	30	108070	73227	3150	30	276912	140313	11516	30	163710	96243	6380
95	7	2	36	78550	63480	–	30	125960	85061	3664	30	289797	144364	12098	30	169991	98446	6610
111	7	2	36	78692	63633	–	30	124959	84533	3646	30	289793	144261	12020	30	170163	98552	6648
115	7	2	36	78591	63151	–	30	109922	74503	3188	30	282238	141557	11809	30	168210	97753	6568
119	7	2	36	78563	63477	–	30	122960	83264	3577	30	287020	142555	11978	30	170386	98332	6620
123	7	2	36	78691	63672	–	30	118337	80519	3452	30	286730	143475	11899	30	170798	99083	6643

Table 2: Factorization of N up to 7-bit (with 1-node).

4.1 Naive Circuit

Firstly, we factored small RSA-type composite numbers up to 7-bit with 1-node by using Shor's quantum circuits generated by our implementation. Table 2 shows the required resources and timings for factorization, where Q, G, D, T denote the number of required qubits, the number of elementary gates, the depth of Shor's circuit, and the timing data in seconds. Since we used 1-node only, 30 qubits are available for factorization. Thus, circuits with R-ADD for 6-bit and 7-bit integers cannot be proceeded (denoted by '–' in the table).

As in the table, required resources depend on the parameters N and n, but on n mainly. The ratio D/G seems to be a constant depending on the features of R-ADD, GT-ADD, Q-ADD, and MIX-ADD. Since Q-ADD has many 1-qubit operations and is easy to parallelize, so that the ratio D/G is smaller (0.50-0.53 for Q-ADD and 0.57-0.63 for MIX-ADD) compared to other ADDs (0.79 to 0.81 for R-ADD, 0.68-0.79 for GT-ADD). Though G and D are expected in the following order, $O(n^3)$ for R-ADD, $O(n^4)$ for Q-ADD and MIX-ADD, and $O(n^5)$ for GT-ADD, the results differ from expected ones. The reason is that the composite numbers are so small that other terms rather than the dominant term affect. The difference may be smaller for larger parameters.

4.2 Optimized Circuit

Then, we factor 8-bit and 9-bit integers with 512-nodes. GT-ADD is used for the experiment because it requires less number of qubits and gates compared to other ADDs in the case of these small integers. In order to decrease the number of gates and the depth as much as possible, we used `optimize_light` option of

N	n	a	Q	G	D	T	N	n	a	Q	G	D	T	N	n	a	Q	G	D	T
129	8	7	34	152780	100141	256	259	9	2	38	288684	183065	6143	395	9	2	38	319088	203494	7307
133	8	2	34	169108	111205	247	265	9	6	38	272685	173346	5620	403	9	2	38	307506	195485	6271
141	8	2	34	183453	120170	287	267	9	2	38	309270	196137	6572	407	9	2	38	338095	214301	7907
143	8	2	34	207514	135907	311	287	9	2	38	359003	228259	7511	411	9	2	38	335319	214006	7404
145	8	6	34	158918	105271	262	291	9	2	38	308155	195603	6542	413	9	2	38	327370	208569	6648
155	8	2	34	198473	130150	311	295	9	2	38	334848	212590	6370	415	9	2	38	359587	228199	7723
159	8	2	34	217743	142924	335	299	9	2	38	321523	204402	7094	417	9	5	38	267426	171328	5940
161	8	3	34	155238	103030	238	301	9	2	38	317493	202575	6461	427	9	2	38	324243	207582	6862
177	8	5	34	168876	111997	259	303	9	2	38	353151	224856	7559	437	9	2	38	314856	200771	5925
183	8	2	34	207468	136410	297	305	9	3	38	285798	182560	6350	445	9	2	38	339458	216426	6572
185	8	3	34	180752	119593	282	309	9	2	38	309354	196737	6358	447	9	2	38	373035	237421	7448
187	8	2	34	208281	137192	328	319	9	2	38	367923	233944	7419	451	9	2	38	306484	195876	5999
201	8	7	34	170050	112064	244	321	9	7	38	260877	166496	5899	453	9	2	38	286538	183164	6146
203	8	2	34	193163	126762	285	323	9	2	38	304490	193554	5956	469	9	2	38	303229	193946	6246
205	8	3	34	178117	117326	276	327	9	2	38	322336	204745	6115	471	9	2	38	343707	219148	7473
209	8	3	34	165014	109327	243	329	9	3	38	285506	182113	6099	473	9	3	38	303975	194528	6933
213	8	2	34	184210	121450	272	335	9	2	38	349246	222013	8104	481	9	3	38	281077	180267	6815
215	8	2	34	204621	134697	327	339	9	2	38	317273	201779	7109	485	9	2	38	305606	195586	6502
217	8	5	34	178741	118044	255	341	9	2	38	291468	186213	6363	489	9	7	38	302012	193333	7218
219	8	2	34	204160	134522	299	355	9	2	38	310783	197410	7491	493	9	2	38	329162	210756	6188
221	8	2	34	200121	131790	283	365	9	2	38	322926	206125	6346	497	9	3	38	296472	189877	5750
235	8	2	34	198443	130597	285	371	9	2	38	324641	206674	6287	501	9	2	38	322414	207063	6335
237	8	2	34	193348	127347	286	377	9	3	38	316691	202612	6676	505	9	6	38	313370	200596	6811
247	8	2	34	208086	136900	289	381	9	2	38	321134	204686	5860	511	9	3	38	395310	252188	8226
249	8	11	34	186487	123502	292	391	9	2	38	326281	207709	6697							
253	8	2	34	202159	133987	306	393	9	5	38	281956	179878	6014							

Table 3: Factorization of N up to 9-bit with GT-ADD (with 512-nodes).

Qulacs which unifies successive 1-qubit gates to one gate. However, the effect was very limited: it reduce the number of gates by only 1 percent.

Since factorization of 9-bit integers require 38-qubits, and 256-nodes are sufficient for the computation, other 256-nodes can be used for the speed-up. To do so, we used the `fused_swap_option` option of *mpiQulacs* which enables to distribute tasks to identified nodes for efficient computation.

Table 3 summarizes the factorization results. As in the table, we have succeeded factoring all RSA-type integers up to 9-bit. The largest integer we factored here was $N = 511$, which requires 8226 seconds (2.3 hours). On the other hand, `optimize_light` option works very well for Q-ADD, since Q-ADD uses a lot of successive 1-qubit gates. In fact, the optimized quantum circuit for factoring $N = 511$ with Q-ADD requires 225523 gates and 187618 depth, and it factors $N = 511$ in 7050 seconds (1.96 hours) in the experiment.

4.3 Resource Estimation of Basic Circuit

Finally, we estimated the quantum circuit resources for factoring 1024-bit and 2048-bit integers. For each $8 \leq n \leq 24$, we generated 10 composite numbers N randomly (170 composite numbers in total). Then, we generated the quantum circuit for each N with the `optimize_light` option, and evaluated the number of elementary gates and the depth. Here, we used R-ADD since resources become smaller than others for larger N's. Next, we computed the average of resources for each n. See Appendix 3 for the detailed values from this experiment.

	$n = 1024$			$n = 2048$		
	qubits	gates	depth	qubits	gates	depth
Kunihiro [12]	3074	2.90×10^{11}	–	6146	2.32×10^{12}	–
R-ADD	5121	2.74×10^{11}	2.20×10^{11}	10241	2.19×10^{12}	1.76×10^{12}
GT-ADD	4098	3.33×10^{15}	1.02×10^{15}	8194	1.07×10^{17}	3.23×10^{16}
Q-ADD	4098	2.87×10^{13}	2.43×10^{13}	8194	4.58×10^{14}	3.88×10^{14}
MIX-ADD	4098	1.49×10^{13}	1.26×10^{13}	8194	2.37×10^{14}	2.00×10^{14}

Table 4: Circuit estimation for factoring 1024/2048-bit integers

From average values for $8 \leq n \leq 24$, we obtain approximation polynomials

$$G_{\text{R-ADD}}(n) = 254.84981n^3 - 338.63513n^2 - 177.31878n + 3112.36316,$$
$$D_{\text{R-ADD}}(n) = 204.72160n^3 - 265.74807n^2 - 515.61678n + 5232.47162,$$

using least squares method with assuming that $G(n) = O(n^3)$ and $D(n) = O(n^3)$. Then, by substituting $n = 1024$ and $n = 2048$ to these polynomials, we obtain approximations as in Table 4. Compared to the estimation by Kunihiro, our estimation decreases by about 5.6% for the number of gates. We do not discuss the feasibility of such a huge quantum computer, however, if the quantum circuit for factoring a 2048-bit integer is proceeded by an ideal quantum computer which can proceed the operation in the same speed as Google's Sycamore [2], that took 200 seconds to sample 10^6 times with a circuit with depth 40, factoring requires about 101.70 days, which seems infeasible by the current quantum technology.

As in the R-ADD case, we obtain the approximation polynomials for GT-ADD, Q-ADD and MIX-ADD

$$G_{\text{GT-ADD}}(n) = 2.931n^5 + 20.169n^4, \quad D_{\text{GT-ADD}}(n) = 0.883n^5 + 21.875n^4,$$
$$G_{\text{Q-ADD}}(n) = 25.983n^4 + 59.060n^3, \quad D_{\text{Q-ADD}}(n) = 21.993n^4 + 44.503n^3,$$
$$G_{\text{MIX-ADD}}(n) = 13.378n^4 + 136.287n^3, \quad D_{\text{MIX-ADD}}(n) = 11.309n^4 + 107.630n^3,$$

with assuming that $G(n) = O(n^k)$ and $D(n) = O(n^k)$ for $k = 5, 4, 4$, respectively. Since k is large, we compute the approximation polynomials only in the upper two degrees. We obtain approximations for $n = 1024$ and 2048 as in Table 4. Factoring a 2048-bit composite number requires about 5107, 61.4 and 31.7 years (GT, Q and MIX-ADD, respectively). MIX-ADD requires less time than GT/Q-ADD, but more time than R-ADD. However, MIX-ADD is useful in environments where the number of available qubits is limited since MIX-ADD requires fewer qubits than R-ADD.

5 Concluding Remarks

In this paper, we have proposed the MIX-ADD method that can flexibly select the optimal ADD circuit for each of the ADD circuits in the Mod-EXP.

This method reduces the number of elementary gates and the depth in Shor's quantum circuit while maintaining a lower qubit requirement compared to R-ADD. Next, we have implemented Shor's algorithm for factoring general composite numbers using 4 different ADD (R-ADD, GT-ADD, Q-ADD and MIX-ADD), and successfully factored 96 RSA-type composite numbers up to 9-bit using the quantum computer simulator developed by Fujitsu. Finally, we have estimated the number of gates and depth required of Shor's quantum circuit for larger composite numbers by actually generating quantum circuits, and gave the estimation for 1024 and 2048-bit integers.

A new finding obtained from our experiments is that the required resources related to Shor's algorithm can be evaluated based on actual implementation rather than theoretical analysis, at least for small parameters, by using the quantum simulator. The effectiveness of improvements can be assessed through actual implementation and experiments on quantum simulators.

Our implementations are based on the basic construction of Shor's quantum circuit. Future work will involve experiments and resource estimation using advanced circuits that apply complex techniques to reduce the number of qubits, as well as under noisy conditions.

Appendix 1. Examples of Greedy Method

Figure 6 shows an example of our greedy method for $k = 4, c = 1$, and Figure 7 for $k = 5, c = 1, 2$. The number of Toffoli gates is 6 for $k = 4, c = 1$, 8 for $k = 5, c = 2$, and 10 for $k = 5, c = 1$, which matches $4k - 8 - 2c$.

Appendix 2. Effectiveness of Greedy Method

In order to show the superiority of our greedy method, we factored RSA-type composite numbers up to 7-bit with 1-node, without and with the greedy method for GT-ADD. Results are summarized in Table 5, where results in the 'Greedy' column coincide with the results shown at 'GT-ADD' column in Table 2. As shown in the table, the greedy method reduces the number of gates to about 66–71%, and the depth to about 45–56%. Since the generated Toffoli gates by the greedy method can be parallelized easily, the effect on the depth is much larger than that on the number of gates. Our analysis in Section 3.2 shows that the greedy method reduces the number of gates to about 56.25% (calculated as $3/(16/3) \times 100$) when n is sufficiently large.

Appendix 3. Data for circuit estimation in Section 4.3

Figure 8 shows the average values and the approximation polynomials described in Section 4.3. Table 6 summarizes the average values, lowest values, and highest values for the R-ADD case. There is virtually no difference between them.

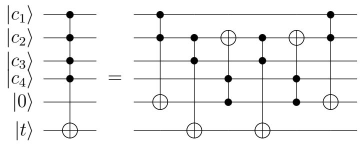
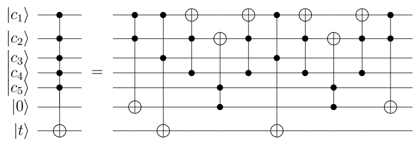
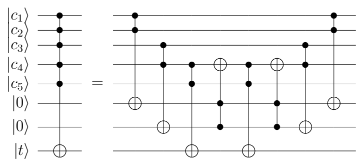


Fig. 6: Conversion from a C^4-NOT to C^2-NOTs with 1 clean qubit



(a) With 1 clean qubit



(b) With 2 clean qubits

Fig. 7: Conversion from a C^5-NOT to C^2-NOTs

Acknowledgments

We would like to thank Shintaro Sato, Hirotaka Oshima, Kazunori Maruyama, Masayoshi Hashima and Kohta Nakashima from Fujitsu for their efforts in using the quantum computer simulator. The sixth author conducted the research supported by JST CREST JPMJCR2113 and JPSP KAKENHI Grant Number JP21H03440.

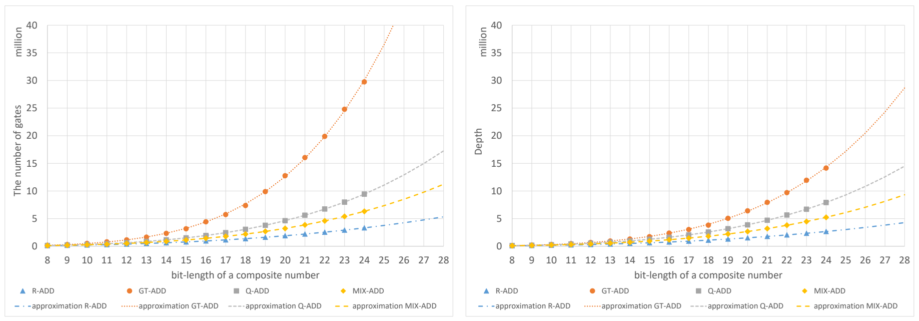
References

1. Amico, M., Saleem, Z.H., Kumph, M.: Experimental study of Shor's factoring algorithm using the IBM Q Experience. Physical Review A **100**(1), 012305 (2019). https://doi.org/10.1103/PhysRevA.100.012305
2. Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J.C., Barends, R., et al.: Quantum supremacy using a programmable superconducting processor. Nature **574**(7779), 505–510 (2019). https://doi.org/10.1038/s41586-019-1666-5
3. Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., et al.: Elementary gates for quantum computation. Physical review A **52**(5), 3457 (1995). https://doi.org/10.1103/PhysRevA.52.3457
4. Beauregard, S.: Circuit for shor's algorithm using 2n+3 qubits. arXiv preprint quant-ph/0205095 (2002). https://doi.org/10.48550/arXiv.quant-ph/0205095
5. Beckman, D., Chari, A.N., Devabhaktuni, S., Preskill, J.: Efficient networks for quantum factoring. Physical Review A **54**(2), 1034 (1996). https://doi.org/10.1103/PhysRevA.54.1034
6. Boudot, F., Gaudry, P., Guillevic, A., Heninger, N., Thomé, E., Zimmermann, P.: Factorization of RSA-250. https://web.archive.org/web/20200228234716/https://lists.gforge.inria.fr/\pipermail/cado-nfs-discuss/2020-February/001166.html (2020)
7. Draper, T.G.: Addition on a quantum computer. arXiv preprint quant-ph/0008033 (2000). https://doi.org/10.48550/arXiv.quant-ph/0008033

GT-ADD				No Greedy			Greedy			Ratio		
N	n	a	Q	G_0	D_0	T_0	G_1	D_1	T_1	G_1/G_0	D_1/D_0	T_1/T_0
15	4	2	18	17881	17763	1.5	12595	9838	0.91	0.71	0.56	0.61
21	5	2	22	37044	36867	10.1	25325	18824	5.2	0.69	0.52	0.52
33	6	5	26	66679	66433	227	44461	31436	92	0.67	0.48	0.41
35	6	2	26	83216	82966	282	55387	38869	115	0.67	0.47	0.41
39	6	2	26	93136	92886	315	61941	43483	129	0.67	0.47	0.41
51	6	2	26	83790	83541	285	55755	39348	116	0.67	0.48	0.41
55	6	2	26	93156	92906	315	61899	43507	129	0.67	0.47	0.41
57	6	5	26	77400	77151	262	51360	36346	107	0.67	0.48	0.41
65	7	3	30	126462	126133	6814	82676	56199	2430	0.66	0.45	0.36
69	7	2	30	151490	151157	8121	98774	66690	2866	0.66	0.45	0.36
77	7	2	30	159842	159509	8546	104285	70616	3033	0.66	0.45	0.36
85	7	2	30	152208	151875	8165	99407	67570	2906	0.66	0.45	0.36
87	7	2	30	183909	183575	9864	120027	80999	3485	0.66	0.45	0.36
91	7	2	30	178045	177711	9537	116234	78729	3369	0.66	0.45	0.36
93	7	2	30	165750	165417	8857	108070	73227	3150	0.66	0.45	0.36
95	7	2	30	193219	192885	10358	125960	85061	3664	0.66	0.45	0.36
111	7	2	30	191313	190979	10257	124959	84533	3646	0.66	0.45	0.36
115	7	2	30	168479	168145	9048	109922	74503	3188	0.66	0.45	0.36
119	7	2	30	188369	188035	10112	122960	83264	3577	0.66	0.45	0.36
123	7	2	30	181029	180695	9692	118337	80519	3452	0.66	0.45	0.36

Table 5: Factorization of N with GT-ADD without and with the greedy method

8. Gidney, C., Ekerå, M.: How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. Quantum **5**, 433 (2021). https://doi.org/10.22331/q-2021-04-15-433
9. Gouzien, E., Sangouard, N.: Factoring 2048-bit RSA integers in 177 days with 13 436 qubits and a multimode memory. Physical Review Letters **127**(14), 140503 (2021). https://doi.org/10.1103/PhysRevLett.127.140503
10. IBM: 433–qubits quantum processor, Osprey. https://research.ibm.com/blog/next-wave-quantum-centric-supercomputing
11. Imamura, S., Yamazaki, M., Honda, T., Kasagi, A., Tabuchi, A., Nakao, H., et al.: mpiqulacs: A distributed quantum computer simulator for A64FX-based cluster systems. arXiv preprint arXiv:2203.16044 (2022). https://doi.org/10.48550/arXiv.2203.16044
12. Kunihiro, N.: Exact analyses of computational time for factoring in quantum computers. IEICE transactions on fundamentals of electronics, communications and computer sciences **88**(1), 105–111 (2005). https://doi.org/10.1093/ietfec/e88-a.1.105
13. Lanyon, B.P., Weinhold, T.J., Langford, N.K., Barbieri, M., James, D.F., Gilchrist, A., et al.: Experimental demonstration of a compiled version of Shor's algorithm with quantum entanglement. Physical Review Letters **99**(25), 250505 (2007). https://doi.org/10.1103/PhysRevLett.99.250505
14. Lu, C.Y., Browne, D.E., Yang, T., Pan, J.W.: Demonstration of a compiled version of Shor's quantum factoring algorithm using photonic qubits. Physical Review Letters **99**(25), 250504 (2007). https://doi.org/10.1103/PhysRevLett.99.250504
15. Lucero, E., Barends, R., Chen, Y., Kelly, J., Mariantoni, M., Megrant, A., et al.: Computing prime factors with a Josephson phase qubit quantum processor. Nature Physics **8**(10), 719–723 (2012). https://doi.org/10.1038/nphys2385



(a) the number of gates (b) depth

Fig. 8: Average values of the number of gates and the depth of Shor's circuit for n-bit integers. The dashed lines represent approximation polynomials.

	Average		Lowest		Highest	
n	gates	depth	gates	depth	gates	depth
8	109654	87762.8	107372	85241	111100	89648
9	159835.8	128291.1	158641	126288	162018	130662
10	223161.5	179300.9	218662	173506	225187	182354
11	299715.2	240530.5	297074	238494	302068	243438
12	393551.7	315677.3	387423	309689	397390	320203
13	503860	403660.2	497511	396630	508398	409619
14	633082.9	507259.3	627831	500757	641738	517706
15	783469.7	627229.6	779900	622474	788900	634906
16	958542.6	769055.5	953155	761892	967134	781641
17	1152195	922632.6	1146738	915406	1157066	929076
18	1373845.4	1100267	1366213	1087255	1384917	1115100
19	1628078.6	1307037.4	1618600	1293395	1644736	1331018
20	1901953.5	1525742.6	1886368	1503536	1909138	1535936
21	2213048.9	1776710.5	2203974	1764214	2222441	1789865
22	2549491.8	2045255.8	2532631	2023007	2562329	2062360
23	2919664.5	2342540.3	2907098	2320299	2936593	2366786
24	3326305.5	2669232.6	3295857	2629337	3349921	2701801

Table 6: Resources of optimized Shor's circuit with R-ADD

16. Martin-Lopez, E., Laing, A., Lawson, T., Alvarez, R., Zhou, X.Q., O'brien, J.L.: Experimental realization of Shor's quantum factoring algorithm using qubit recycling. Nature photonics **6**(11), 773–776 (2012). https://doi.org/10.1038/nphoton.2012.259

17. Politi, A., Matthews, J.C., O'brien, J.L.: Shor's quantum factoring algorithm on a photonic chip. Science **325**(5945), 1221–1221 (2009). https://doi.org/10.1126/science.1173731

18. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS. pp. 124–134. IEEE Computer Society Press (Nov 1994). https://doi.org/10.1109/SFCS.1994.365700

19. Suzuki, Y., Kawase, Y., Masumura, Y., Hiraga, Y., Nakadai, M., Chen, J., et al.: Qulacs: a fast and versatile quantum circuit simulator for research purpose. Quantum **5**, 559 (2021). https://doi.org/10.22331/q-2021-10-06-559

20. Vedral, V., Barenco, A., Ekert, A.: Quantum networks for elementary arithmetic operations. Physical Review A **54**(1), 147 (1996). https://doi.org/10.1103/PhysRevA.54.147
21. Yan, B., Tan, Z., Wei, S., Jiang, H., Wang, W., Wang, H., et al.: Factoring integers with sublinear resources on a superconducting quantum processor. arXiv preprint arXiv:2212.12372 (2022). https://doi.org/10.48550/arXiv.2212.12372