因数分解量子アルゴリズムの全量子シミュレーション

山田 崇* 丹羽 純平* 由良 文孝† 今井 浩*,†

要旨: Shor の整数因数分解量子アルゴリズムを実現する量子回路のうち, Beauregard によって改良された回路と従来の回路双方を量子計算シミュレーションシステム (QCSS) 上で実装し, 分散メモリ型並列計算機上でシミュレーションをおこなう. Beauregard の量子回路は, 任意の L ビットの数の因数分解回路を $2L+3$ 量子ビットで構築でき, 量子回路全体をシミュレータ上に実装できる. この環境で 12 ビットまでの数の因数分解ができることを確認する. また, 並列化による時間短縮効果を検証する. 加えて, 実装した量子回路のデコヒーレンスエラー・操作エラーに対する耐性を調べる. さらには, 近似フーリエ変換 (AQFT) を Beauregard の量子回路に適用可能かどうか実験の結果から議論する.

A Large Scale Simulation of the Quantum Factorization Algorithm with Full Implementation

TAKASHI YAMADA*, JUMPEI NIWA*, FUMITAKA YURA† AND HIROSHI IMAI*,†

Abstract: We implement the quantum order-finding circuit for integer factorizing introduced by Beauregard and the existing ones on Quantum Computer Simulation System (QCSS), and do simulation on the scalable distributed-memory parallel computer. The quantum circuit by Beauregard uses $2L+3$ qubits to factorize an arbitrary L-bit number. By using this circuit we can implement the whole quantum circuit on the simulator. On this environment we confirm that we can factorize up to 12-bit numbers. In addition, we examine effect of parallelization and investigate the robustness of the circuits for decoherence and operational errors. Moreover, we discuss from the experimental results the applicability of approximate QFT (AQFT) to Beauregard's circuit.

1 Introduction

In 1994, P. W. Shor proposed a polynomial-time quantum algorithm for order-finding [11], which seemed to be intractable on classical computation, and showed the integer factorization quantum algorithm by using it. This was a breakthrough for quantum computation.

The major cost in the quantum order-finding circuit used in arbitrary L-bit number factorization, is not the QFT (quantum Fourier transform) but the modular exponentiation, which has $O(L^3)$ depth. This modular exponentiation circuit requires $2L+6$ additional working qubits [6]. In this approach, $5L+6$ qubits are totally required to construct the quantum order-finding circuit.

In order to reduce the number of qubits, there are several studies utilizing the information of the numbers to be factored. Both the circuit that Vandersypen et al. [12] have realized in the NMR quantum computer and the circuit that Obenland et al. [8] have implemented in the quantum computer simulator rely on the property of the number to be factored (for instance, 15). Hence, these circuits cannot be used for factorization of other numbers.

There is another simulation study to reduce the number of qubits [7]. Modular exponentiation is executed classically and only QFT is executed on the simulator. This approach can deal with much larger numbers. However, it is impossible to check whether the modular exponentiation part of the order-finding circuit is robust or not in the presence of errors.

What we need is the general circuit which does not rely on any properties of the number to be factored and requires as few qubits as possible. Beauregard improved the order-finding circuit [3]. The number of qubits required to construct the circuit is reduced to $2L+3$ to factorize an arbitrary L-bit number whereas the depth of the circuit is still $O(L^3)$.

The following three techniques were used:

1. the quantum adder circuit by QFT introduced by Draper [4] is applied.
2. "one control quantum bit trick" [9] is applied.
   To use this trick, the number of qubits for controlling is diminished to only one, whereas $2L$ controlling qubits are required in the existing circuit.
3. QFT is replaced with AQFT (approximate QFT) [2].

As a result, it becomes possible to implement the whole quantum order-finding circuit used in arbitrary number factorization. With current technologies, however, it is still hard to make actual quantum computers dealing with many qubits. The quantum computers realized today can treat at most 7 qubits [12].

Hence, simulation is required to investigate the behavior of the whole quantum order-finding circuit for large numbers (that is, for many qubits cases). Many qubits simulation requires more computational power than is usually available on not only sequential computers but also symmetric multi-processors. Scalability of processors and memories must be required for many qubits simulations. Therefore, we have proposed and developed the QCSS (Quantum Computation Simulation System) on the scalable distributed-memory parallel computers. The current implementation on SCoreIII [10] enables us to perform up to 34 qubits simulation and to factorize as many as 15-bit numbers ($2L+3 \leq 34 \Rightarrow L \leq 15$) if we consider nothing about execution time.

In this paper, we implement the Shor's factorization algorithm and the four order-finding circuits (by Beauregard

*東京大学大学院情報理工学系研究科コンピュータ科学専攻
Department of Computer Science, Graduate School of Information Science and Technology, University of Tokyo.
E-mail address: tyamada@is.s.u-tokyo.ac.jp
† 科学技術振興事業団今井量子計算機構プロジェクト
ERATO Project Quantum Computation and Information, JST.

[3], by Miquel et al. [6], and two more circuits for comparison) on QCSS. On this environment we measure simulation time and confirm that the effects of parallelization are enough. In addition, through the simulation, we can analyze performance and robustness of the quantum order-finding circuits in the presence of decoherence and operational errors by comparing the success probability among them. Moreover, we discuss from the experimental results the applicability of approximate QFT of the order-finding circuit by Beauregard.

The results will be useful when we build quantum computers with many qubits, since it is still difficult to theoretically analyze the effects of decoherence and operational errors in the real order-finding circuit. The significance of our work lies in the full simulation and the analysis of the whole quantum order-finding circuit for the factorization of an arbitrary numbers.

2 Quantum Computer Simulation System

For many qubits simulation, we use the scalable simulator, QCSS (Quantum Computer Simulation System) that runs efficiently on scalable distributed-memory computers. QCSS has been implemented in the two ways. We adopt the QCSS programmed by using MPI [1][i]. Therefore it runs on any platforms as long as MPI is implemented.

2.1 Register

A collection of n qubits is called a *register* of size n. The general qubit state $|\Phi\rangle$ of the n-qubit register is

$$|\Phi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle, \quad \text{where } \alpha_i \in \mathbf{C}, \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1, \quad (1)$$

that is, the state of a n-qubit register is represented by a unit vector in 2^n-dimensional Hilbert space $\mathcal{H}_{2^n}$.

In a classical computer, an array of 2^n complex numbers is required to have the quantum state. In order to store a complex number $\alpha = x + iy$, a pair of real numbers (x, y) is prepared. Each real number is represented by a double precision word. The size of it is 8 bytes (64 bits) on many systems.

Therefore, 2^{n+4} bytes memory is required to deal with the state of a n-qubit register in a classical computer. For example, a classical computer with 1 GB memory can represent a 26-qubit system.

In addition, if the number of qubits of a quantum register increments by one, the memory needed to represent the register is doubled. It is therefore difficult to simulate larger quantum systems since current PCs and WSs have at most several GB memory.

To overwhelm the limit, we choose the distributed-memory parallel computers as the simulation platform. The array of amplitudes is distributed to the *nodes* of the parallel computers.

Here let the number of the nodes be 2^p. Fig. 1 shows the representation of the quantum register on the distributed-memory paralell computers. Each node has 2^{n-p} continuous elements of the amplitude array.

[i] Another QCSS is designed for SMP (Symmetric Multi-Processors).

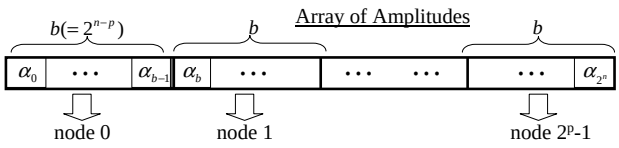

Figure 1: Representation of the n-qubit register.

2.2 Evolution

The time evolution of a n-qubit register is determined by a unitary operator in 2^n-dimensional Hilbert space $\mathcal{H}_{2^n}$. This unitary operator is a matrix of size $2^n \times 2^n$. In general $2^n \times 2^n$ space is required to represent the matrix.

However, operators which have simple structures are used when we design quantum circuits. That is, an evolution step is performed by applying a unitary operator of size 2×2 to a single qubit or by applying a controlled unitary operator such as a controlled-NOT gate. They require only 2×2 space to simulate such an evolution step.

Single qubit gate: Single qubit gate is represented as a transformation in which 2×2 unitary operator $U = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix}$ to the i-th qubit of the register of size n $(0 \le i < n)$.

Let the most significant bit (MSB) be the 0-th qubit. When the unitary operator U is applied to the i-th qubit, the overall unitary operator X applied to the n-qubit register state is a matrix of size of $2^n \times 2^n$ and described as follows:

$$X = \left(\bigotimes_{k=0}^{i-1} I \right) \otimes U \otimes \left(\bigotimes_{k=i+1}^{n-1} I \right)$$

$$= \begin{bmatrix} \overbrace{\begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix}}^{2^{n-i}} & & & O \\ & \ddots & & \\ & & \ddots & \\ O & & & \overbrace{\begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix}}^{2^{n-i}} \end{bmatrix}, \quad (2)$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{2^n}$$

where

$$U_{11} = \begin{bmatrix} u_{11} & & O \\ & \ddots & \\ O & & u_{11} \end{bmatrix} \quad U_{12} = \begin{bmatrix} u_{12} & & O \\ & \ddots & \\ O & & u_{12} \end{bmatrix}$$

$$U_{21} = \begin{bmatrix} u_{21} & & O \\ & \ddots & \\ O & & u_{21} \end{bmatrix} \quad U_{22} = \underbrace{\begin{bmatrix} u_{22} & & O \\ & \ddots & \\ O & & u_{22} \end{bmatrix}}_{2^{n-i-1}}. \quad (3)$$

X is constructed by 2^i submatrices aligned diagonally, and each submatrix is a $2^{n-i} \times 2^{n-i}$ matrix $\begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix}$.

Therefore, each column vector of X has always two non-zero elements.

When we apply the matrix X to the quantum state $|\Phi\rangle$, we require only the four complex numbers u_{11}, u_{12}, u_{21}, u_{22} and the two elements of the array of amplitudes. Since the array of amplitudes is distributed into 2^p nodes, the inter-node communication occurs depending on the size of the submatrix when the product $X|\Phi\rangle$ is calculated.

When the size of the submatrix is less than or equal to the number of elements of the amplitude array which each node possesses, that is, $2^{n-i} \leq 2^{n-p} \iff p \leq i$, the computation of the product $X|\Phi\rangle$ is performed locally (Figure 2).

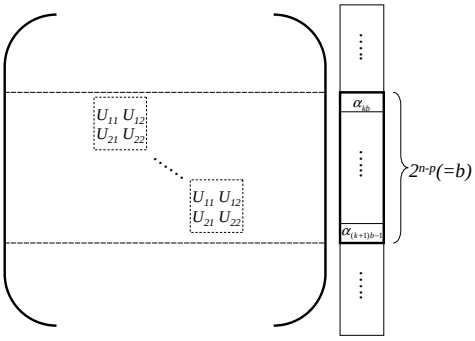

Figure 2: $X|\phi\rangle$ (the submatrix is small)

When the size of the submatrix is more than the number of elements of amplitude array, that is $p > i$, the amplitude array that another node has is required in order to compute $X|\Phi\rangle$, the next state. Therefore, inter-node communication occurs.

Let (i,i) and (i,j) be the two non-zero elements of i-th column of X. Then, clearly, (j,i) and (j,j) become the two non-zero elements of j-th column of X. Thus, if a node q requires the amplitude array that a node p has, then the node p requires the amplitude array that the node q has (Figure 3). Note that the node p communicates with only the node q and the node q communicates with only the node p.

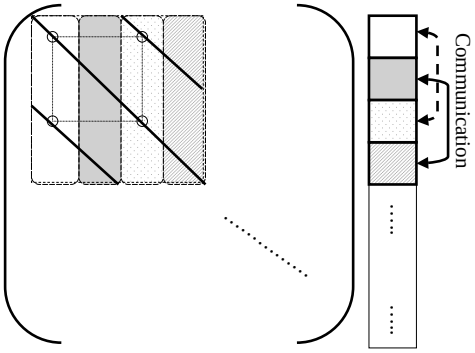

Figure 3: $X|\phi\rangle$ (the submatrix is large)

In order to reduce the simulation time, first, communication overheads must be reduced, and coarse-grained data transfer is required. As shown in Figure 4, data-transfer size is not the element-size (16 bytes) but the buffer-size. This time, we set the optimal buffer-size as 2048 bytes on the basis of the pilot study results.

Second, communications and computations must be overlapped. As shown in Figure 4, a node i sends its amplitude array data with buffer size and the node i performs computations using only its own amplitude array data. The node i waits until it receives the necessary data from a node j. Then, the node i performs computation using both its own data and the received data. The node j perform the same process as the node i performs.

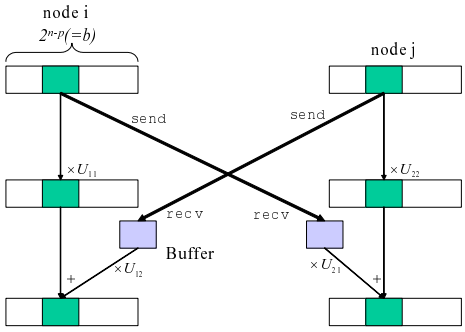

Figure 4: Data transfer diagram

Controlled qubit gate: Suppose that a unitary matrix $U = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix}$ is applied to the i-th qubit if and only if the c-th qubit is 1. Let CTX be the overall matrix of size of $2^n \times 2^n$.

First, we consider the matrix X mentioned above as if there are no controlled qubits. Then, for each j ($1 \leq j < 2^n - 1$), the j-th row of the matrix CTX ($CTX[j]$) is defined as follows:

$$CTX[j] = \begin{cases} X[j] & \text{if the } c\text{-th bit of } j \text{ is 1} \\ I[j] & \text{if the } c\text{-th bit of } j \text{ is 0} \end{cases}, \quad (4)$$

where I is the unit matrix.

In this case, we also do not have to generate CTX or X explicitly. We only have to have the 2×2 matrix U.

2.3 Measurement

The gate for measurement of i-th qubit of n-qubit register is designed as follows:

1. In each node i, two values o_i and z_i are calculated such that $o_i = \sum_{c \in S} |\alpha_c|^2$, $z_i = \sum_{c \notin S} |\alpha_c|^2$, where $S = \{s | \text{the } i\text{-th bit of } s \text{ is 1.}\}$.
2. The 0-th node collects o_i and z_i from each node i. Then it should have $O = \sum_{i=0}^{2^p-1} o_i$ and $Z = \sum_{i=0}^{2^p-1} z_i$, and the sum of O and Z should be 1.
3. The 0-th node generates a random number r ($0 \leq r < 1$). We consider that 0 is observed if $r < Z$ and that 1 is observed if $Z \leq r$.
4. The 0-th node sends the result of measurement to all nodes. Then each node generates the post-measurement state according to the result which it receives from 0-th node.

2.4 Error model

The simulator can represent *decoherence* and *operational* errors. By using it we can analyze robustness of some quantum circuits in the presence of decoherence and operational errors. Here we explain the error models that the simulator adopts.

Decoherence error: The simulator implements *depolarizing channel* as the decoherence error model.

In this channel, when the depth of the circuit increments by one, with probability $1-p$ each qubit is left alone, and with probability p the error occurs, and there are equal probabilities that σ_x, σ_y or σ_z (Pauli's X, Y, Z) are applied to each qubit independently. The error rate p is determined by users.

In general, on an actual quantum computer, gates are applied parallelly as much as possible. Then the depth of a circuit is not always equal to the number of the gates of the circuit. The simulator has a function which applies error gates. It is called only if the depth of the circuit increments.

Operational error: Generally, all of single qubit gates are described with *rotations* $U_R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$, and *phase shifts* $U_{P_1}(\varphi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix}$, $U_{P_2}(\varphi) = \begin{bmatrix} e^{i\varphi} & 0 \\ 0 & 1 \end{bmatrix}$. For example, NOT gate $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ is described as $U_R(\pi/2)U_{P_1}(\pi)$, and Hadamard gate $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ is described as $U_R(\pi/4)U_{P_1}(\pi)$.

The simulator represents inaccuracies by adding small deviations to the angles of rotation θ and φ. Each error angle is drawn from Gaussian distribution with the standard deviation σ, which is also determined by users.

3 Preliminaries for the Algorithm

In this section, we review briefly the quantum algorithm for factorization proposed by Shor. After that, we introduce the quantum circuits for order finding. This is an important part of this algorithm and there are several way for realization. We focus on the circuit by Beauregard and explain in detail the techniques which the circuit uses. Moreover, we consider approximate QFT. This approximation is helpful to reduce the number of quantum gates but also effects of decoherence.

3.1 Shor's Factorization Algorithm

Suppose a L-bit number N is an odd and non-primal number[ii].

1. [CLASSICAL] Choose a random integer number a so that $1 < a \leq N-1$.
2. [CLASSICAL] Using Euclid's algorithm, determine if $\gcd(a,\ N) > 1$, and if so, return the factor $\gcd(a,\ N)$.
3. [QUANTUM] Use the order-finding quantum algorithm to find the order r of a modulo N.
4. [CLASSICAL] If r is odd or r is even but $a^{r/2} \equiv -1 \pmod N$ or $a^r \not\equiv 1 \pmod N$, print "failed" and terminate this algorithm.
5. [CLASSICAL] Otherwise, compute $r_\pm = \gcd(a^{r/2} \pm 1,\ N)$. Test to see if these are non-trivial factors of N, and return the factor $r_\pm$ and terminate this algorithm if so.

The process to find a *order* of $a \bmod N$(step 3) is known to be computable in polynomial-time on a quantum computer.

3.2 Order Finding Circuit

We review the quantum circuits for order-finding we have implemented for the following experiments.

3.2.1 Traditional Circuit

Figure 5 shows the quantum order-finding circuit constructed by Miquel et al. [6]. In this figure, H is Hadamard gate, QFT^{-1} is inverse QFT gate, and m is measurement gate. The gate named U_a maps $|x\rangle$ to $|(ax) \bmod N\rangle$. It is used for *modular exponentiation* ($|x\rangle \rightarrow |x^a \bmod N\rangle$).

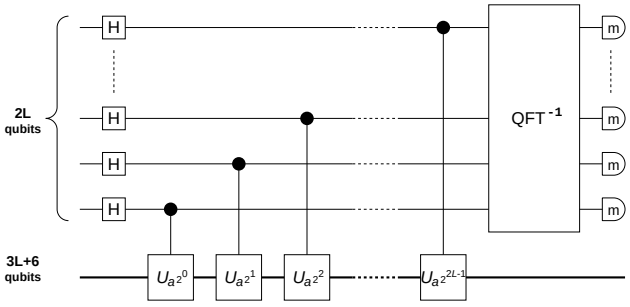

Figure 5: Traditional order-finding circuit.

U_a gate is constructed by combination of (controlled) addition gates, (controlled) subtraction gates, and so on. Because the design of them is a straightforward application of the classical carry gates and summation gates, many working qubits are required and $3L+6$ qubits are wasted for modular exponentiation [13].

Therefore, totally $5L+6$ qubits are required. We call this order-finding circuit the *traditional* circuit.

3.2.2 Improvements by Beauregard

Last year, Beauregard improved the order-finding circuit (see Figure 6 and [3]). We review the following three techniques that Beauregard adopted.

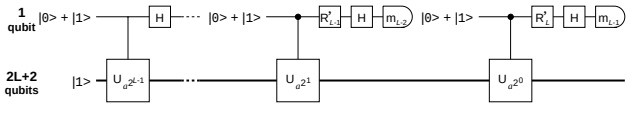

Figure 6: Improved order-finding circuit.

[ii] Even if N is even or primal, there is no problem. It is trivial whether N is even or odd, and we can determine in polynomial time whether N is primal or not.

Addition by QFT [4]: This uses QFT and reduces the number of qubits necessary for addition by removing temporary carry bits. This addition takes two values a and b, computes $F(a)$ (QFT of a) and then uses b to evolve $F(a)$ into $F(a+b)$. Inverse QFT may then be applied and the sum recovered.

One control quantum bit trick [9]: It is shown that the (inverse) Fourier transform preceding the final measurement can be calculated in a semiclassical way [5]. That is, instead of performing the entire transform and then making measurements on all control qubits afterwards, we can apply Hadamard transform to the first qubit and then measure it. The controlled phase shifts by this first qubits are then replaced by single qubit operations given the result of the measurement on the first.

QFT is replaced with AQFT [2]: In physical implementations, we cannot perform rotation gates below a certain tolerance. Therefore, AQFT is adopted. Details are given in the next subsection.

Eventually $2L+3$ qubits are needed to construct this circuit. We call it *improved* order-finding circuit.

3.3 Approximate QFT

The order-finding circuit by Beauregard (improved circuit) contains $O(L^2)$ QFT gates. In this section we try to approximate the QFT gates. This means that the quantum state after QFT is applied is approximated. However, we obtain the result by measurement. Note that this approximation affects not the result itself but the probability the correct one is obtained.

The QFT circuit (Figure 7) contains of L Hadamard gates and $L(L+1)/2$ controlled phase shift gates.

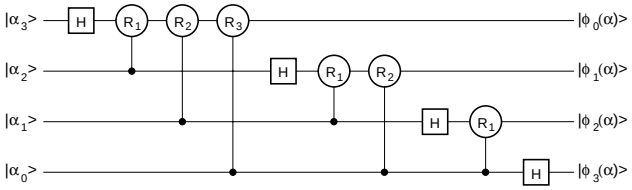

Figure 7: QFT circuit with 4 qubits.

Similarly, *the AQFT of degree* m (see [2]) is represented by Hadamard gates and the phase shift gates in which the target qubit and the controlling qubit are far apart (in the register) are neglected, that is, those operations R_k for which the phase shift $2\pi/2^k \leq 2\pi/2^m$ for some m such that $1 \leq m \leq L$ are dropped[iii]. In that case, we need L gates of Hadamard transform, and $(2L-m)(m-1)/2$ gates of phase shift, which is less than those on the QFT case since $m < L$.

Figure 8 shows the $m=2$ AQFT circuit.

4 Experiments

4.1 Environment

As an experimental environment, we adopted SCoreIII [10], a PC cluster system, built by Real World Computing Partnership (RWCP). By running the SCore cluster system software on it, SCoreIII realizes a parallel computer with high scalability, high cost-performance, and high usability.

SCoreIII has 512 computation hosts. Each node has two Pentium III processors (933 MHz) and 512 MB memory. The nodes are connected each other with Myrinet-2000 and ethernet.

By running QCSS on SCoreIII, we can deal with up to 34 qubits and factorize 15-bit numbers with Beauregard's order-finding circuit if we consider nothing about execution time.

[iii] If $m=1$, the circuit is equivalent to the Hadamard circuit, and if $m=L$, it is equivalent to the QFT circuit.

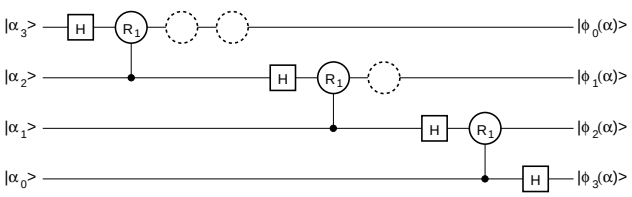

Figure 8: Approximate QFT (AQFT) circuit ($m=2$) with 4 qubits. The circles drawn with dotted lines mean dropped gates according to the definition.

4.2 Implementation

We have implemented on QCSS the two order-finding circuits, traditional and improved ones. Besides, two more circuits have been implemented for comparison.

Table 1 shows the features of these circuits.

Table 1: Features of the order-finding circuits.

	#qubits	one control qubit trick	addition by QFT
A (improved)	$2L+3$	W	W
B	$4L+2$	W/O	W
C	$3L+7$	W	W/O
D (traditional)	$5L+6$	W/O	W/O

4.3 Execution time for order-finding

First of all the experiments, we measured execution time for simulation.

With circuit A, some L-bit numbers which satisfy the condition as input are chosen ($4 \leq L \leq 15$) and factorized on QCSS with changing the number of working processors $P = 2^p$ ($0 \leq p \leq 8$). We assume that there are no decoherence and operational errors during computation.

Then, we have confirmed that factorization of up to 12-bit numbers succeeded on the environment. Figure 9 shows the result as a double-logarithm chart. The points connected with bold lines mean that an enough effect of parallelization is gained. The execution time becomes almost half if the number of processors doubles. The points connected with broken lines mean that we did not measure the time because we expect it will take much time for simulation but that we can forecast the time. We can see that it

will take 3.7 million seconds, about 43 days, to try a factorization of a 15-bit number.

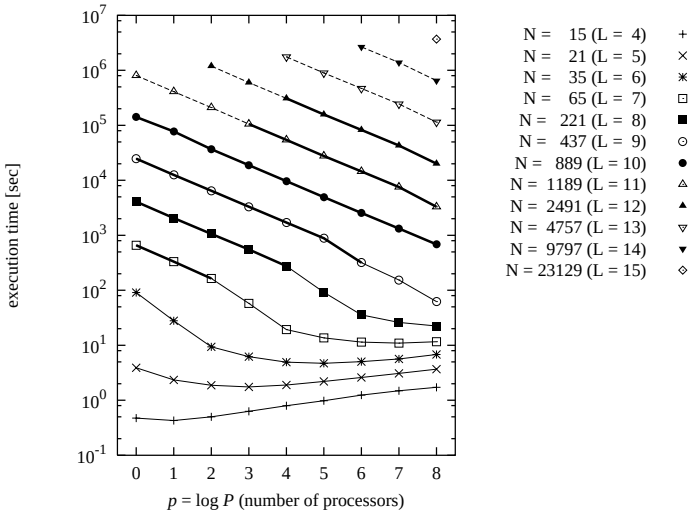

Figure 9: Relation between number of processors and execution time.

As for other circuits, we confirmed that simulation to factorize 21 (5 bits) finished within a minute with 128 processors by using circuit B and C, but that on the same environment it took 1003 seconds per trial to factorize 35 (6 bits). With circuit D it took 318 seconds per trial to factorize 15 (4 bits).

Note that it will take much more time for simulation in the presence of especially decoherence error as mentioned in the previous section.

4.4 Robustness for decoherence error

To investigate the robustness of the circuits for decoherence and operational errors, an appropriate number of trials are needed for analyses. Therefore it is better that time for simulation is short.

We factorized 21 by using circuit A, B and C on QCSS, with changing decoherence error rate d from 0 to 10^{-2}. With each d 200 trials were done. The number of qubits, the depth of each circuit, the product of this two values, and the number of Hadamard gates, (controlled) NOT gates, rotate gates are shown in Table 2. As circuit A and B adopt

Table 2: Quantum circuits to factorize 21.

	#qubits	depth	#qubits ×depth	# gates		
				Had.	(c-)NOT	Rot.
A	13	15384	2.0×10^5	2660	4510	9610
B	22	12399	2.7×10^5	2660	4525	9645
C	22	28771	6.3×10^5	20	30250	10
D	31	26084	8.1×10^5	18	26040	36

the adder by QFT, there are much more Hadamard gates and rotate gates than circuit C and D, whereas circuit C and D have much more NOT gates because the existing addition circuit is realized as a combination of (controlled) NOT gates.

How many times the error gates are applied totally depends on the size of the circuit (the product of the number of qubits and the depth) because they are applied independently to each qubit in our implementation. For example, in circuit A, if the decoherence error rate is 10^{-6}, some 0.2 error gates is applied to the circuit in each trial.

We calculated the success probability, the probability with which a trial for factorization succeeded by using the quantum part of the algorithm and the order-finding circuit (step 3 in section 3).

Table 3 shows the result.

Table 3: Success probability (%) in the presence of decoherence error.

	decoherence error rate d					
circuit	0	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}
A	12.0	8.5	3.5	3.0	2.5	4.0
B	11.5	11.5	3.0	2.5	1.5	1.0
C	17.0	5.5	3.0	1.5	1.0	0.5
D	N/A					

As for success probability of the factorization algorithm $\mathrm{Prob}_{\mathrm{succ}}(N)$, if we assume that there are no decoherence and operational errors, it is known that the lower bound is given as the following inequality:

$$\mathrm{Prob}_{\mathrm{succ}}(N) \geq \frac{\phi(N)}{N-1} \cdot \left(\frac{1}{2} \cdot \frac{4}{\pi^2} \cdot \frac{e^{-\gamma}}{\log\log N} \right),$$

where $\phi(N)$ is the Euler number of N, and γ is the Euler constant. If we substitute 21 to N, then the right hand of above inequality is about 11.6%.

From the result we find some tendencies: As for circuit A and B, the circuits are robust for decoherence error if d is less than 10^{-5}, whereas as for circuit C it seems not to be useful for factorization when there is decoherence.

However, it is strange that the success probabilities differs among the circuits when $d = 0$. These should converge to constant. There are too few trials to see the differences among the circuits from this experimental result. Then we are now planning that the probability we measure correct answers is computed accurately from the amplitude of the final state. We expect that the result will appear in the near future.

Next, in order to make the effects of error gates to the circuits clearer, we investigate the relation between the error gates applied and the success probability with circuit A and C. We did 2000 and 1000 trials with circuit A and C, respectively.

Table 4 shows the relation between the number of error gates applied to circuit A the success probability.

As for circuit C, Table 5 shows the relation between the number of error gates applied to the circuit and the success probability.

From the results we can see that on the both circuit A and C, the more error gates are applied, the less trials succeeds

Table 4: Relation between the number of error gates applied and success probability (circuit A).

#error	#succeed	#trial	#succeed/#trial
0	20	126	15.87%
1	29	301	9.63%
2	20	444	4.50%
3	12	349	3.44%
4	9	284	3.17%
5	2	186	1.08%
6	2	80	2.50%
7	1	47	2.13%

Table 5: Relation between the number of error gates applied and success probability (circuit C).

#error	#succeed	#trial	#succeed/#trial
0	18	106	16.98%
1	18	153	11.76%
2	16	172	9.30%
3	8	122	6.56%
4	5	46	10.87%
5	1	23	4.35%
6	0	13	0.00%
7	0	3	0.00%

and that the success probability seems to be diminished exponentially as the number of the error gates increases. The size of a circuit seems to be more serious than the other aspects of it (for instance, the ratio of Hadamard gates, NOT gates and rotate gates in the circuit). Perhaps the decoherence error model we adopt has lead such a result. More experiments and analyses are needed to examine the results.

4.5 Robustness for operational errors

We factorized 21 by using circuit A, B and C on QCSS, with changing standard deviation of operational error σ from 0 to 10^{-2}. With each σ 200 trials were done. Then we calculated the success probability.

Table 6 shows the result.

Table 6: Success probability (%) in the presence of operational errors.

	standard deviation σ					
circuit	0	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}
A	12.0	13.0	8.5	11.5	6.5	7.5
B	11.5	10.0	12.5	12.5	9.0	8.5
C	17.0	15.0	12.5	8.5	11.5	12.5
D	N/A					

As for the circuit A, if the standard deviation of operational errors σ is less than 10^{-3}, we find that the circuit is not affected by the operational error, whereas circuit C is robust for operational errors. We consider that this is because circuit A has much more rotate gates, which it is considered is fragile for operational errors, than circuit C.

However, circuit B seemed to be more robust for circuit A, though there are almost the same number of rotate gates in circuit B. In circuit A, "one controlling qubit trick" is used. The operational error may affect the only one control qubit directly and may make the result we can measure incorrect.

4.6 Effect of Approximate QFT (AQFT)

Beauregard's order-finding circuit consists of many QFT gates and quantum adder gates similar to QFT gates: If a L-bit number is factorized, $O(L^2)$ QFT gates and $O(L^2)$ quantum adder gates by QFT are required to construct the circuit.

We prepared the circuit in which the QFT gates and the quantum adder gates are replaced by the AQFT gates and the A-ADD gates, respectively. A-ADD gate is an approximated addition gate like AQFT.

There are two reasons why we use these approximated gates: First, the time taken to finish the algorithm on not only the simulator but only actual quantum computers in the near future becomes short. Second, the circuit becomes more robust for decoherence errors because the depth of the circuit is reduced in our implementation[iv].

We tried to factorize 187 (8 bits, 11×17) on circuit A with changing decoherence error rate d from 0 to 10^{-2} and the degree of AQFT m (see section 3.3) from 1 to 9. Here we suppose that operational errors do not occur. In this circuit, there are 544 QFT gates, 544 inverse QFT gates, 640 adder gates and 640 inverse adder gates.

Table 7 shows the properties of the circuit to factorize 187.

Table 7: Properties of circuit A ($N = 187$).

	#qubits	depth	#qubit×depth
A	19	70080	1.3×10^7

In each m and d, the trials were repeated 100 times. From the results obtained from the experiment, we computed the success probability.

Figure10 shows the success probability of the quantum part of the algorithm (step 3 in section 3). The horizontal axe and the vertical axe represent the degree of AQFT (m) and the number of success trials, respectively.

From the figure we can see that the success probability is almost zero if the decoherence error rate d is greater than 10^{-6}. This means that the quantum circuit is not helpful due to the decoherence errors.

[iv] In general, the depth does not change.

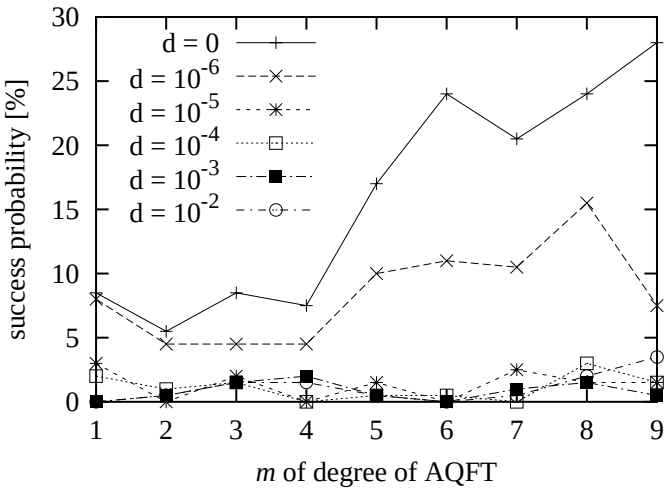

Figure 10: Success probability (%) of the circuit with AQFT and A-ADD.

When $d = 0$ and $d = 10^{-6}$, the factorization succeeds by the quantum circuit. There appears to be a gap of the success probability at the point between $m = 4$ and 5.

Barenco et al. have pointed out in Ref. [2] that if we assume that there is no decoherence, (the decoherence error rate d equals to zero), the ability of AQFT gates is almost the same as QFT gates as far as the degree of AQFT m satisfies the following lower bound condition,

$$m > \log L + 2, \tag{5}$$

where L is the size of the quantum register applied the AQFT, and that when $d > 0$ the optimum m is found near this lower bound.

As stated above, we estimate that if there is no decoherence and $m \geq 5$, the success probability is slightly less than that of the circuit without AQFT and A-ADD.

If we substitute $L = 9$ to Equation (5), we obtain the following inequality:

$$m > 5.17. \tag{6}$$

We can find that the simulation results corresponds to the theoretical condition as in Equation (6).

Therefore, we consider that AQFT is useful for the order-finding circuit. We expect that AQFT gates are applicable to the circuit if the number to factorize becomes larger as far as the condition (Equation (5)) is satisfied.

Furthermore, in the order-finding circuit, AQFT may be rather more useful than QFT in the presence of decoherence. This is because to use AQFT gates means that the number of gates affected by decoherence diminishes.

In fact, the experimental results indicate it. When the decoherence error rate is 10^{-6}, more number of trials in which the factorization finishes to succeed in the case of $m = 5$, 6, 7, 8 (using AQFT) than in the case of $m = 9$ (using QFT) as shown in Figure 10.

5 Conclusion

We have developed QCSS (Quantum Computer Simulation System) on the scalable distributed-memory computer. Under QCSS on SCoreIII, we have implemented the entire order-finding circuit proposed by Beauregard. We confirmed the effect of parallelization. Time to simulate the factorization almost halved if the number of nodes doubled.

By using the circuit, we have confirmed that Shor's factorization algorithm is really effective. Suppose that there are no errors, the success probability calculated theoretically is almost the same as that gained in the simulation.

We have analyzed robustness of the whole quantum order-finding circuits in the presence of decoherence and operational errors. If the decoherence rate d is greater than 10^{-5} or if the standard deviation of the operational errors σ is greater than 10^{-4}, it seems to be hard to use the order-finding circuit in practice. We also have found that AQFT is more useful than QFT for decoherence rate $10^{-6} \sim 10^{-5}$.

References

[1] ANL Mathematics and Computer Science. MPI—message passing interface standard. http://www-unix.mcs.anl.gov/mpi/.

[2] A. Barenco, A. Ekert, K.-A. Suominen, and P. Törmä. Approximate quantum Fourier transform and decoherence. *Phys. Rev. A*, 54:139–146, 1996.

[3] S. Beauregard. Circuit for Shor's algorithm using $2n + 3$ qubits. *Quantum Information and and Computation*, 3(2):175–185, 2003.

[4] T. G. Draper. Addition on a quantum computer, 2000. Los Alamos Physics Preprint Archive, quant-ph/0008033.

[5] R. B. Griffiths and C.-S. Niu. Semiclassical Fourier transform for quantum computation. *Phys. Rev. Lett.*, 76(17):3228–3231, 1996.

[6] C. Miquel, J. P. Paz, and R. Perazzo. Factoring in a dissipative quantum computer. Los Alamos Physics Preprint Archive, http://xxx.lanl.gov/abs/quant-ph/9601021, 1996.

[7] J. Niwa, K. Matsumoto, and H. Imai. General-purpose parallel simulator for quantum computing. Phys. Rev. A **62**, 062317, December 2002.

[8] K. Obenland and A. Despain. A parallel quantum computer simulator. In *High Performance Computing*, 1998.

[9] S. Parker and M. B. Plenio. Efficient factorization with a single pure qubit and $\log N$ mixed qubits. *Phys. Rev. Lett.*, 85:3049–3052, 2000. Also on Los Alamos Physics Preprint Archive, quant-ph/0001066.

[10] PC Cluster Consortium. The website of PC Cluster Consortium. http://www.pccluster.org.

[11] P. W. Shor. Polynomial-time algorithms for prime factorization and descrete logarithms on a quantum computer. *SIAM J. Comp.*, 26:1484–1509, 1997. Also on Los Alamos Physics Preprint Archive, quant-ph/9508027.

[12] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sterwood, and I. L. Chuang. Experimental realization of Shor's quantum factorization algorithm using nuclear magnetic resonance. *Nature*, (414):883–887, December 2000.

[13] V. Vedral, A. Barenco, and A. K. Ekert. Quantum networks for elementary arithmetic operations. *Physical Review A*, 54(1):147-153, 1996.