Quantum Information and Computation, Vol. 6, No. 2 (2006) 184–192

A QUANTUM CIRCUIT FOR SHOR'S FACTORING ALGORITHM USING 2n + 2 QUBITS

YASUHIRO TAKAHASHI
NTT Communication Science Laboratories, NTT Corporation
Atsugi, Kanagawa 243-0198, Japan

NOBORU KUNIHIRO
The University of Electro-Communications
Chofu, Tokyo 182-8585, Japan

Received August 24, 2005
Revised January 3, 2006

We construct a quantum circuit for Shor's factoring algorithm that uses $2n + 2$ qubits, where n is the length of the number to be factored. The depth and size of the circuit are $O(n^3)$ and $O(n^3 \log n)$, respectively. The number of qubits used in the circuit is less than that in any other quantum circuit ever constructed for Shor's factoring algorithm. Moreover, the size of the circuit is about half the size of Beauregard's quantum circuit for Shor's factoring algorithm, which uses $2n + 3$ qubits.

Keywords: quantum circuits, elementary arithmetic operations, factoring
Communicated by: R Jozsa & M Mosca

1 Introduction

In 1994, Shor proposed an efficient quantum algorithm for factoring [1]. The main part of the algorithm is a quantum algorithm for order-finding and it uses the quantum Fourier transform (QFT) and modular exponentiation. These operations have been analyzed in depth and efficient quantum circuits for order-finding have been constructed [2, 3, 4, 5, 6, 7, 8, 9]. Since efficient quantum circuits for order-finding would be useful for performing Shor's algorithm on a quantum computer, there is great interest in constructing such circuits. In particular, there is interest in reducing the number of qubits in circuits for order-finding since it seems extremely difficult to realize quantum computers with many qubits.

Many studies of quantum circuits for order-finding have focused on reducing the number of qubits [2, 3, 6, 8, 9]. The best known circuit is the one constructed by Beauregard [2], which uses $2n + 3$ qubits, where n is the length of the number to be factored. When the approximate QFT [10] is used in the circuit, the depth and size of the circuit are $O(n^3)$ and $O(n^3 \log n)$, respectively. To reduce the number of qubits, the circuit uses the quantum addition proposed by Draper [5] and several techniques, such as the hardwiring of classical values and the sequential computation of the QFT [9, 11].

In this paper, we construct a quantum circuit for order-finding that uses $2n + 2$ qubits. When the approximate QFT is used in the circuit, the depth and size of the circuit are $O(n^3)$ and $O(n^3 \log n)$, respectively. The number of qubits used in the circuit is less than that in any other quantum circuit ever constructed for order-finding. Moreover, the size of the circuit is about half the size of Beauregard's circuit. The key component of the circuit is a new quantum circuit for the comparison of two numbers that uses "uninitialized" ancillary qubits. That is, the circuit uses ancillary qubits that are not initialized to $|0\rangle$. In the following, we use the

standard notation for quantum states and use the standard diagrams for quantum circuits [12].

2 Construction of the Circuit for Order-Finding

2.1 Decomposition of Modular Exponentiation

Let N be an n-bit number to be factored and a be an n-bit number less than N such that a is coprime to N. The quantum algorithm for order-finding finds the order of a modulo N, where the order of a modulo N is the least number r such that $a^r = 1 \bmod N$. The algorithm consists of the QFT and modular exponentiation. Thus, the usual circuit for order-finding is constructed using the circuit for the QFT and the circuit for the modular exponentiation operation that maps $|x\rangle|1\rangle$ to $|x\rangle|a^x \bmod N\rangle$, where x is a $2n$-bit number.

The construction method of our circuit for order-finding is similar to that of Beauregard's one [2]. We use the sequential computation of the QFT to reduce the number of qubits. In doing so, it suffices to construct a quantum circuit for the controlled modular multiplication operation $\mathrm{MM}(a^{2^i})$ for $i = 2n-1, \ldots, 0$ (in place of the modular exponentiation operation), where $\mathrm{MM}(a)$ is defined as

$$\mathrm{MM}(a)|c\rangle|x\rangle = \begin{cases} |c\rangle|ax \bmod N\rangle & \text{if } c = 1, \\ |c\rangle|x\rangle & \text{otherwise,} \end{cases}$$

where c is a one-bit number and x is an n-bit number. The circuit for order-finding uses $\mathrm{MM}(a^{2^i})$ for $i = 2n-1, \ldots, 0$ sequentially.

$\mathrm{MM}(a)$ is decomposed into the controlled modular product-sum operations $\mathrm{MPS}(a)$ and $\mathrm{MPS}(a^{-1})$, where $\mathrm{MPS}(a)$ is defined as

$$\mathrm{MPS}(a)|c\rangle|x\rangle|b\rangle = \begin{cases} |c\rangle|x\rangle|ax + b \bmod N\rangle & \text{if } c = 1, \\ |c\rangle|x\rangle|b\rangle & \text{otherwise,} \end{cases}$$

where c is a one-bit number and x and b are two n-bit numbers. Actually, $\mathrm{MM}(a)$ is computed using $\mathrm{MPS}(a)$, $\mathrm{MPS}(a^{-1})$, and one swap operation as follows (We assume that $c = 1$ and omit the register containing c):

$$\begin{aligned} |x\rangle|0\rangle &\rightarrow |x\rangle|ax \bmod N\rangle \\ &\rightarrow |ax \bmod N\rangle|x\rangle \\ &\rightarrow |ax \bmod N\rangle|x - a^{-1}ax \bmod N\rangle = |ax \bmod N\rangle|0\rangle, \end{aligned}$$

where the first operation is $\mathrm{MPS}(a)$, the second is the swap operation, and the third is $\mathrm{MPS}(a^{-1})^{-1}$. Note that a^{-1}, which is the inverse of a modulo N, exists since a is coprime to N, and a^{-1} is computed using Euclid's algorithm efficiently.

$\mathrm{MPS}(a)$ is decomposed into the doubly controlled modular addition operations $\mathrm{MA}(2^i a)$ for $i = 0, \ldots, n-1$, where $\mathrm{MA}(a)$ is defined as

$$\mathrm{MA}(a)|c_1\rangle|c_2\rangle|b\rangle = \begin{cases} |c_1\rangle|c_2\rangle|a + b \bmod N\rangle & \text{if } c_1 = c_2 = 1, \\ |c_1\rangle|c_2\rangle|b\rangle & \text{otherwise,} \end{cases}$$

where c_1 and c_2 are two one-bit numbers and b is an n-bit number. Actually, $\mathrm{MPS}(a)$ is computed using the relationship

$$ax + b \bmod N = (2^{n-1}ax_{n-1} + (\cdots(2^1 ax_1 + (2^0 ax_0 + b \bmod N) \bmod N)\cdots) \bmod N),$$

where $x_{n-1}\cdots x_0$ is the binary representation for x. More precisely, we repeatedly apply $\mathrm{MA}(2^i a)$ to the register containing b for $i = 0, \ldots, n-1$, where the first control bit of $\mathrm{MA}(2^i a)$ is used as the control bit of $\mathrm{MPS}(a)$ and the second control bit of $\mathrm{MA}(2^i a)$ is used to see whether $x_i = 1$ or not. Note that qubits for $x_j (j \neq i)$ are "idle"; that is, they are not used during $\mathrm{MA}(2^i a)$. As explained in the next subsection, our new idea is to use these "idle" qubits as "uninitialized" ancillary qubits.

$\mathrm{MA}(a)$ is decomposed into addition (subtraction) and comparison operations since the relationship

$$a + b \bmod N = \begin{cases} a + b - N & \text{if } a + b \geq N, \\ a + b & \text{otherwise,} \end{cases}$$

holds. Our circuit for order-finding is different in the construction of the circuit for $\mathrm{MA}(a)$ from Beauregard's one. In the next subsection, we briefly review Beauregard's circuit for $\mathrm{MA}(a)$ and explain our new one.

2.2 New Idea for Efficient Modular Addition Circuit

Beauregard's circuit for $\mathrm{MA}(a)$ is based on the following algorithm.

1. Add a to the initial content b. The resulting state is $|b + a\rangle$.
2. Subtract N from $b + a$. The resulting state is $|b + a - N\rangle$.
3. Write the high bit y of $b + a - N$ on one ancillary qubit to decide whether $b + a - N < 0$. The resulting state is $|b + a - N\rangle|y\rangle$, where y is 1 if $b + a - N < 0$ and 0 otherwise.
4. Add N to $b + a - N$ if y is 1. The resulting state is $|a + b \bmod N\rangle|y\rangle$.
5. Subtract a from $a + b \bmod N$. The resulting state is $|(a + b \bmod N) - a\rangle|y\rangle$.
6. Write the negation of the high bit z of $(a + b \bmod N) - a$ on the ancillary qubit to decide whether $(a + b \bmod N) - a < 0$. The resulting state is $|(a + b \bmod N) - a\rangle|y \oplus z \oplus 1\rangle$, where z is 1 if $(a + b \bmod N) - a < 0$ and 0 otherwise.
7. Add a to $(a + b \bmod N) - a$. The resulting state is $|a + b \bmod N\rangle|0\rangle$.

Note that $y \oplus z \oplus 1 = 0$ since the relationship

$$a + b \bmod N \geq a \Leftrightarrow a + b < N,$$

holds. Addition in the above algorithm is implemented by Draper's quantum addition circuit using QFTs [5]. Comparison in the above algorithm is essentially subtraction and also implemented by Draper's.

Our algorithm is as follows. The point is that we directly compare b and $N - a$ without computing $b + a - N$.

1. Compare the initial content b to $N - a$ and write the result y on one ancillary qubit. The resulting state is $|b\rangle|y\rangle$, where y is 1 if $b < N - a$ and 0 otherwise.
2. Add a to b if y is 1 and subtract $N - a$ from b if y is 0. The resulting state is $|a + b \bmod N\rangle|y\rangle$.

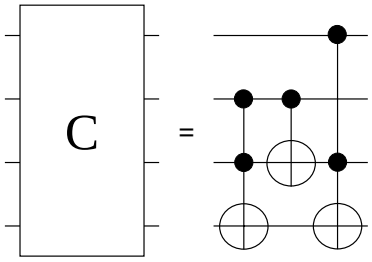

Fig. 1. The C gate. The gate consists of one CNOT gate and two Toffoli gates.

3. Compare $a+b \bmod N$ to a and write the negation of the result z on the ancillary qubit. The resulting state is $|a+b \bmod N\rangle|y \oplus z \oplus 1\rangle = |a+b \bmod N\rangle|0\rangle$ as above.

The important difference between Beauregard's and our algorithm is the number of qubits for the register containing $a+b \bmod N$ at the end of the computation. Beauregard's algorithm deals with intermediate results that may consist of $n+1$ qubits (e.g. $b+a-N$) and thus needs $n+1$ qubits for the register. On the other hand, our algorithm does not deal with such intermediate results and thus only needs n qubits for the register.

If we use circuits for addition, subtraction, and comparison that use no new ancillary qubits, the above difference implies that the number of qubits used in our circuit for order-finding is less than that in Beauregard's by one. We implement addition and subtraction in our algorithm by Draper's quantum addition circuit since the circuit does not need new ancillary qubits. Thus, the only problem is to construct such a circuit for comparison. More precisely, the problem is to construct a quantum circuit for the comparison operation COMP(a) with no new ancillary qubits, where COMP(a) is defined as

$$\mathrm{COMP}(a)|b\rangle|z\rangle = |b\rangle|z \oplus y\rangle,$$

where b is an n-bit number and z is a one-bit number and y is 1 if $a > b$ and 0 otherwise.

Our idea for constructing a quantum circuit for COMP(a) is to introduce a circuit for computing only the high bit of the sum that uses not new ancillary qubits but "uninitialized" ancillary qubits. Roughly speaking, the output y is the high bit of $a-b = a+(-b)$. Thus, we can construct a quantum circuit for COMP(a) by modifying the conventional ripple-carry adder in [8] if we use n new ancillary qubits. By modifying the adder slightly, this can be done if we use $n-1$ new ancillary qubits. The important point is that all we need to do is compute not the sum of two numbers but only the high bit of the sum of two numbers. Thus, we can use $n-1$ "uninitialized" ancillary qubits in place of $n-1$ new (initialized) ancillary qubits. That is, we can use ancillary qubits that are not set to $|0\rangle$ if the ancillary qubits are reset to their original values at the end of the computation. Fortunately, such $n-1$ qubits are available in the other (mostly idle) register as noted in the previous subsection. The exact construction of the circuit for COMP(a) that uses $n-1$ uninitialized ancillary qubits is shown in the next subsection.

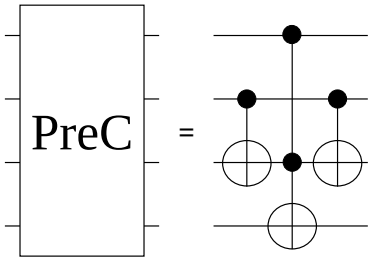

Fig. 2. The PreC gate. The gate consists of two CNOT gates and one Toffoli gate.

2.3 Circuit for Comparison

First, we construct a quantum circuit for computing only the high bit of the sum. Let a and b be two n-bit numbers less than N and $a_{n-1}\cdots a_0$ be the binary representation for a, where a_0 is the low-order bit. Similarly, let $b_{n-1}\cdots b_0$ be the binary representation for b. Let A_i and B_i be the memory locations where a_i and b_i are initially located ($0 \le i \le n-1$). Let R_i be the ancillary memory location where some value r_i is initially located ($1 \le i \le n-1$). Let Z be the output memory location where some value z is initially located. At the end of the computation, A_i will contain a_i, B_i will contain b_i, R_i will contain r_i, and Z will contain $z \oplus y$, where y is the high bit of $a+b$.

The main components of the circuit are the C gate and the PreC gate, which are depicted in Figs. 1 and 2, respectively. The C gate is defined in [8] and used to compute carry bits in the circuit for addition. When we input the four bits

$$|r_{i-1} \oplus c_i\rangle|a_i\rangle|b_i\rangle|r_i \oplus (a_i \oplus b_i)r_{i-1}\rangle$$

to the C gate, it is easy to check that the gate outputs

$$|r_{i-1} \oplus c_i\rangle|a_i\rangle|a_i \oplus b_i\rangle|r_i \oplus c_{i+1}\rangle,$$

where c_i is the i-th carry bit and $2 \le i \le n-1$. Note that c_{i+1} is computed using the relationship

$$c_{i+1} = a_i b_i \oplus b_i c_i \oplus c_i a_i,$$

where $c_0 = 0$ and $0 \le i \le n-1$. Since the ancillary memory locations in our circuit contain some values, in contrast to those in the circuit for addition in [8], we use the PreC gate before we use the C gate. When we input the four bits

$$|r_{i-1}\rangle|a_i\rangle|b_i\rangle|r_i\rangle$$

to the PreC gate, it is easy to check that the gate outputs

$$|r_{i-1}\rangle|a_i\rangle|b_i\rangle|r_i \oplus (a_i \oplus b_i)r_{i-1}\rangle,$$

where $2 \le i \le n-1$.

The circuit for computing only the high bit of the sum is defined as follows.

1. Apply a CNOT gate to a pair of memory locations R_{n-1} and Z.

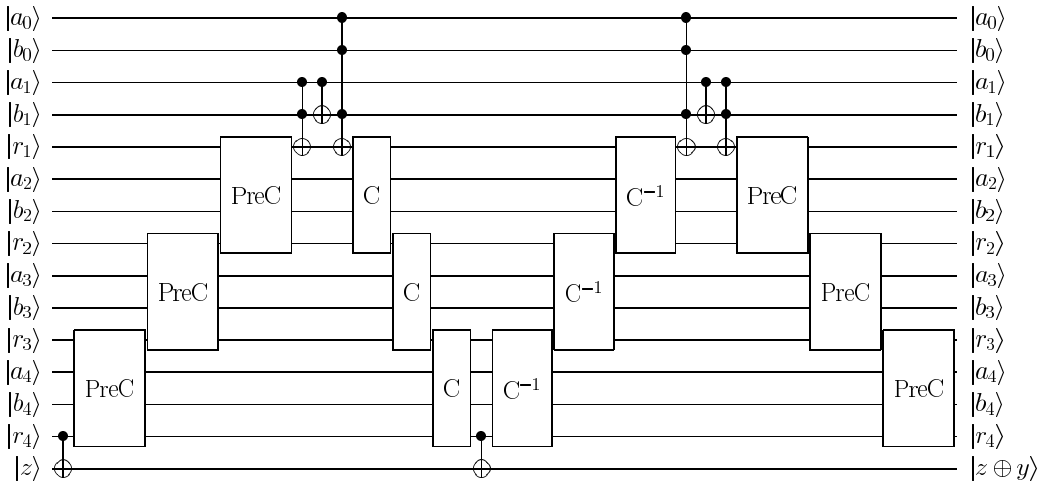

Fig. 3. The HIGHBIT gate for $n = 5$.

2. Apply a PreC gate to a tuple of memory locations R_{n-i-1}, A_{n-i}, B_{n-i}, and R_{n-i} for $i = 1, \ldots, n-2$.

3. Apply a Toffoli gate to a tuple of memory locations A_1, B_1, and R_1. Then, apply a CNOT gate to a pair of memory locations A_1 and B_1. Then, apply a C^3NOT gate to a tuple of memory locations A_0, B_0, B_1, and R_1, where a C^3NOT gate is a NOT gate with three control qubits.

4. Apply a C gate to a tuple of memory locations R_i, A_{i+1}, B_{i+1}, and R_{i+1} for $i = 1, \ldots, n-2$.

5. Apply a CNOT gate to a pair of memory locations R_{n-1} and Z.

6. Undo the previous steps.

Note that Step 3 can be eliminated by adding one PreC gate and one C gate if we can use n uninitialized ancillary qubits. However, we can use only $n-1$ uninitialized ancillary qubits and thus need the step.

The first CNOT gate writes $z \oplus r_{n-1}$ into Z. The PreC gates write $r_i \oplus (a_i \oplus b_i)r_{i-1}$ into R_i for $2 \leq i \leq n-1$. The Toffoli, CNOT, and C^3NOT gates write $a_1 \oplus b_1$ into B_1 and $r_1 \oplus c_2$ into R_1. The C gates write $a_i \oplus b_i$ into B_i and $r_i \oplus c_{i+1}$ into R_i for $2 \leq i \leq n-1$. Thus, $r_{n-1} \oplus c_n$ is written into R_{n-1}. The CNOT gate after the C gates writes

$$(z \oplus r_{n-1}) \oplus (r_{n-1} \oplus c_n) = z \oplus c_n$$

into Z. Here, c_n is the last carry bit and thus the high bit of $a + b$. Then, we apply these operations in reverse order to write the initial values into all memory locations except Z. The circuit is called the HIGHBIT gate and is depicted in Fig. 3 for $n = 5$.

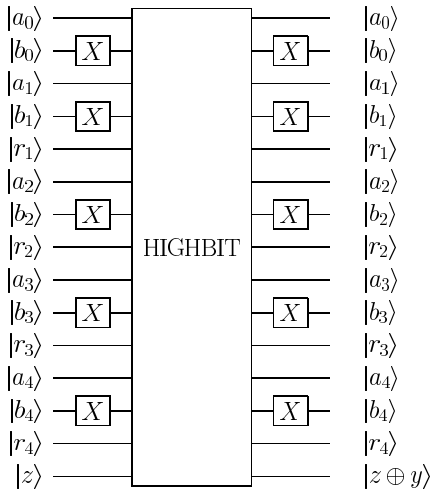

Fig. 4. The circuit for comparison for $n = 5$.

Representing the changes of the values of memory locations R_i caused by PreC and C gates in Fig. 3 would be helpful to readers. The changes of the values caused by the three PreC gates are as follows, where the first arrow corresponds to the first PreC gate, the second to the second PreC gate, and the third to the third PreC gate:

$$\begin{array}{cccccc} & r_1 & & r_2 & & r_3 & & r_4 \\ \rightarrow & r_1 & & r_2 & & r_3 & & r_4 \oplus (a_4 \oplus b_4) r_3 \\ \rightarrow & r_1 & & r_2 & & r_3 \oplus (a_3 \oplus b_3) r_2 & & r_4 \oplus (a_4 \oplus b_4) r_3 \\ \rightarrow & r_1 & & r_2 \oplus (a_2 \oplus b_2) r_1 & & r_3 \oplus (a_3 \oplus b_3) r_2 & & r_4 \oplus (a_4 \oplus b_4) r_3 \end{array}$$

The changes of the values caused by the three C gates are as follows, where the first arrow corresponds to the first C gate, the second to the second C gate, and the third to the third C gate:

$$\begin{array}{cccccc} & r_1 \oplus c_2 & & r_2 \oplus (a_2 \oplus b_2) r_1 & & r_3 \oplus (a_3 \oplus b_3) r_2 & & r_4 \oplus (a_4 \oplus b_4) r_3 \\ \rightarrow & r_1 \oplus c_2 & & r_2 \oplus c_3 & & r_3 \oplus (a_3 \oplus b_3) r_2 & & r_4 \oplus (a_4 \oplus b_4) r_3 \\ \rightarrow & r_1 \oplus c_2 & & r_2 \oplus c_3 & & r_3 \oplus c_4 & & r_4 \oplus (a_4 \oplus b_4) r_3 \\ \rightarrow & r_1 \oplus c_2 & & r_2 \oplus c_3 & & r_3 \oplus c_4 & & r_4 \oplus c_5 \end{array}$$

Then, we construct a quantum circuit for comparison of a and b that outputs 1 if $a > b$ and 0 otherwise. The circuit for comparison is defined as follows.

1. Apply a NOT gate to memory location B_i for $i = 0, \ldots, n-1$.
2. Apply a HIGHBIT gate to a tuple of all memory locations.
3. Apply a NOT gate to memory location B_i for $i = 0, \ldots, n-1$.

The HIGHBIT gate computes only the high bit of $b' + a$, where b' is the bitwise complement of b. It is shown that the high bit of $(b' + a)'$ is 1 if $b \geq a$ and 0 otherwise [13]. Thus, the

negation of the high bit of $(b' + a)'$, that is, the high bit of $b' + a$, is 1 if $a > b$ and 0 otherwise. Thus, the whole circuit writes $z \oplus y$ into Z, where y is 1 if $a > b$ and 0 otherwise. The circuit for comparison is depicted in Fig. 4 for $n = 5$. X represents a NOT gate.

When $a = a_{n-1} \cdots a_0$ is classical and we know what a is beforehand, we do not need to prepare memory location A_i where a_i is initially located for $0 \leq i \leq n-1$. This is because the memory locations contain only classical values and the classical values are used only for controlling some gates. The circuit for COMP(a) is constructed as the circuit for comparison without memory locations for a. Note that the circuit uses $2n$ qubits.

3 Complexity Analysis

We compute the exact number of qubits used in our circuit for order-finding and the orders of the depth and size of the circuit. The size of the circuit is defined as the total number of elementary gates, where the elementary gates are single-qubit gates and CNOT gates. The depth of the circuit is defined as follows. Input qubits are considered to have depth 0. For each gate G, the depth of G is equal to 1 plus the maximal depth of a gate that G depends on. The depth of the circuit is equal to the maximal depth of a gate in the circuit.

The circuit for COMP(a) uses $2n$ qubits as noted in the previous section. The depth and size of the circuit are $O(n)$. The circuit for MA(a) uses $2n+2$ qubits since it uses the circuit for COMP(a) with two control qubits. The depth and size of the circuit are $O(n)$ and $O(n^2)$, respectively, since it uses one QFT and one inverse of it for quantum addition. The circuit for MPS(a) uses $2n + 2$ qubits and the depth and size of the circuit are $O(n^2)$ and $O(n^3)$, respectively, since it uses MA($2^i a$) for $i = 0, \ldots, n-1$ sequentially. The circuit for MM(a) uses $2n+2$ qubits and the depth and the size of the circuit are $O(n^2)$ and $O(n^3)$, respectively. The circuit for order-finding uses $2n + 2$ qubits. The depth and size of the circuit are $O(n^3)$ and $O(n^4)$, respectively, since the circuit uses MM(a^{2^i}) for $i = 2n-1, \ldots, 0$ sequentially.

For a threshold m, the circuit for the approximate QFT (AQFT$_m$) is defined as the QFT except that R_k is ignored when $k > m$ [10]. When the AQFT$_m$ is used in the circuit for addition of two n-bit numbers by Draper in place of the QFT, the probability of error is roughly bounded above by $\pi(n-m) \cdot 2^{-m} \cdot \sqrt{2}$ [14]. This shows that the circuit for addition that uses the AQFT$_m$ is a good approximation of the circuit for addition that uses the QFT when $m = O(\log n)$. Thus, our circuit for order-finding can use the AQFT$_m$ in place of the QFT when $m = O(\log n)$. The size of the resulting circuit is $O(n^3 \log n)$ since the size of the AQFT$_m$ is $O(n \log n)$ when $m = O(\log n)$.

The number of the QFTs and their inverses in our circuit for order-finding is $8n^2$. On the other hand, the number in Beauregard's circuit is $16n^2 + 8n$. Roughly speaking, the difference is implied by the fact that the number in our circuit for MA(a) is half that in Beauregard's. Each number determines the coefficient of the leading term of the polynomial representing the size of each circuit because of the construction of each circuit. Thus, the coefficient of the leading term of the polynomial that represents the size of our circuit is half that of Beauregard's circuit. That is, the size of our circuit is about half that of Beauregard's.

4 Conclusions and Future Work

We constructed a quantum circuit for order-finding that uses $2n + 2$ qubits. When the approximate QFT is used in the circuit, the depth and size of the circuit are $O(n^3)$ and

$O(n^3 \log n)$, respectively. The number of qubits used in the circuit is less than that in any other quantum circuit ever constructed for order-finding. Moreover, the size of the circuit is about half that of the circuit for order-finding constructed by Beauregard.

An interesting challenge would be to construct an $O(n^3)$-size quantum circuit for order-finding that uses $2n + O(1)$ qubits. It seems difficult to construct such a circuit if we use the circuit for addition using the QFT. Linear-size quantum circuits for addition would be useful for constructing such circuits. For example, using the circuit for addition in [15], we can construct an $O(n^3)$-size quantum circuit for order-finding that uses $3n + 2$ qubits. The number of qubits can be decreased to $2n + O(1)$ without increasing the size of the circuit if we have a linear-size quantum circuit for computing $a + b$ that uses a constant number of ancillary qubits and that does not need to prepare qubits for a when a is classical and we know what a is beforehand. Can we construct such a circuit for addition?

Acknowledgements

The authors thank Dr. Kiyoshi Shirayanagi, Dr. Yasuhito Kawano, Dr. Seiichiro Tani, Dr. Go Kato, and the anonymous referees for helpful comments.

References

1. P. W. Shor (1994), *Algorithms for quantum computation: discrete logarithms and factoring*, Proc. 35th Annual IEEE Symposium on Foundations of Computer Science, pp. 124–134.
2. S. Beauregard (2003), *Circuit for Shor's algorithm using* $2n+3$ *qubits*, Quantum Information and Computation, Vol. 3 No. 2, pp. 175–185.
3. D. Beckman, A. N. Chari, S. Devabhaktuni, and J. Preskill (1996), *Efficient networks for quantum factoring*, Physical Review A, Vol. 54 No. 2, pp. 1034–1063.
4. R. Cleve and J. Watrous (2000), *Fast parallel circuits for the quantum Fourier transform*, Proc. 41st Annual IEEE Symposium on Foundations of Computer Science, pp. 526–536.
5. T. G. Draper (2000), *Addition on a quantum computer*, quant-ph/0008033.
6. N. Kunihiro (2005), *Exact analyses of computational time for factoring in quantum computers*, IEICE TRANS. FUNDAMENTALS, Vol. E88–A No.1, pp. 105–111.
7. R. Van Meter (2004), *Fast quantum modular exponentiation*, quant-ph/0408006.
8. V. Vedral, A. Barenco, and A. Ekert (1996), *Quantum networks for elementary arithmetic operations*, Physical Review A, Vol. 54 No. 1, pp. 147–153.
9. C. Zalka (1998), *Fast versions of Shor's quantum factoring algorithm*, quant-ph/9806084.
10. D. Coppersmith (1996), *An approximate Fourier transform useful in quantum factoring*, quant-ph/0201067.
11. M. Mosca and A. Ekert (1999), *The hidden subgroup problem and eigenvalue estimation on a quantum computer*, Lecture Notes in Computer Science, Vol. 1509, pp. 174–188.
12. M. A. Nielsen and I. L. Chuang (2000), *Quantum Computation and Quantum Information*, Cambridge University Press.
13. T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore (2004), *A logarithmic-depth quantum carry-lookahead adder*, Proc. ERATO Conference on Quantum Information Science 2004, pp. 23–24. Also on quant-ph/0406142.
14. N. W. Panike (2000), *Approximate errors*, quant-ph/0008056.
15. Y. Takahashi and N. Kunihiro (2005), *A linear-size quantum circuit for addition with no ancillary qubits*, Quantum Information and Computation, Vol. 5 No. 6, pp. 440–448.