Explicit implementation of quantum circuits on a quantum-cellular-automata-like architecture

Y. Kawano*
NTT Communication Science Laboratories, NTT Corporation, 3-1 Morinosato Wakamiya, Atsugi-shi, Kanagawa 243-0198 Japan

S. Yamashita
NTT Communication Science Laboratories, NTT Corporation, 3-1 Morinosato Wakamiya, Atsugi-shi, Kanagawa 243-0198 Japan and Quantum Computation and Information, ERATO (JST), Matsuo-bldg 2F 406, Iseya-cho, Kawaramachi Marutamachi, Kamigyo-ku, Kyoto 602-0873 Japan

M. Kitagawa
Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama-cho, Toyonaka, Osaka 560-8531 Japan and NTT Communication Science Laboratories, NTT Corporation, 3-1 Morinosato Wakamiya, Atsugi-shi, Kanagawa 243-0198 Japan

(Received 18 April 2003; revised manuscript received 9 January 2004; published 1 July 2005)

We present an efficient strategy to translate a normal quantum algorithm into a sequence of operations on the quantum-cellular-automata-like architecture (QCALA) originally proposed by Lloyd. The QCALA assumes arrays of weakly coupled quantum systems where an interaction exists only between neighboring qubits and can only perform the same quantum operation onto all the qubits. The sequence obtained by the strategy proposed by Lloyd needs at most $12n$ operations, where n is the number of qubits for the original circuit. The sequence obtained by our strategy needs at most $6n$ operations. We also clarified the relations between the upper bound of the number of translated operations and the period of the QCALA and between the upper bound of the number of qubits and the period of the QCALA.

DOI: 10.1103/PhysRevA.72.012301 PACS number(s): 03.67.Lx

I. INTRODUCTION

It is widely believed that quantum computers, once they are actually built, will be much more powerful than today's computers. However, the normal quantum circuit model of quantum computation, where we can perform any unitary operation onto any pair of qubits, seems to be very difficult to physically implement by current technologies.

Another quantum computation model, which may be easier to implement, has been proposed by Lloyd [1]. The model assumes arrays of weakly coupled quantum systems, where an interaction exists only between neighboring qubits. Although this model seems realizable, we cannot construct a normal quantum circuit directly, i.e., we need to translate a normal quantum circuit description into a specific description on a sparse network of qubits. This paper describes the logic operations that can be performed using the model and presents an efficient strategy to translate a normal quantum algorithm into a sequence of those operations for the model. Lloyd's model also contains a third level for error correction, which is removed from our model for simplicity, considering the recent progress in quantum error correction.

After Lloyd's model, similar models [2–6] have been extensively studied by Benjamin *et al.*, who have devoted particular effort to reducing the minimum period of the structure from three (Lloyd's $ABCABC\cdots$ model) to two. They have shown that the symmetric $ABAB\cdots$ model is universal if every logical qubit is encoded into four physical qubits [3]. Although the physical system is simplified, a logical qubit occupies the two periods of the structure. They have also shown the asymmetric $ABAB\cdots$ model is universal if the interactions between A and B (H^{AB}) and those of B and A (H^{BA}) can be collectively but *independently* switched on and off. The $ABAB\cdots$ model with asymmetric interactions $H^{AB} \neq H^{BA}$ could be the minimum periodic structure with unidirectionality and could be realized physically as an $ABAB\cdots$ structure [4], where the distance between A and B and that between B and A is different. However, H^{AB} and H^{BA} cannot be decoupled independently by flipping A or B with π pulses or by any other kind of time reversal techniques. Therefore, the independent switching ability of the interactions must be facilitated, which is far beyond what Lloyd's model requires. Individual or collective Zeeman tuning [5,6] of qubits is also external to Lloyd's model. Those interesting models [3–6] should be regarded as new models rather than the minimum case of Lloyd's model.

Lloyd's model itself has not been developed since the original proposal [1] and his extended preprint [7]. The model should be further elaborated in various directions, such as the hardware (molecules), the pulse shapes and sequences [8], the quantum circuit implementation, the input-output implementation, and the initialization. This paper discusses the implementation of a quantum circuit with improved efficiency compared with Lloyd's original proposal.

Our main purpose here is to introduce a transform algorithm that shortens the sequence of operations compared to the sequence obtainable by Lloyd's original transform algorithm. The idea is as follows. In Lloyd's model, quantum operations only between adjacent qubits are allowed; therefore, operations between remote qubits must be performed after moving data on these remote qubits to adjacent posi-

*Electronic address: kawano@theory.brl.ntt.co.jp

tions by swap operations. In addition, the sequence of operations obtained by Lloyd's transform algorithm moves data back to the initial position after that operation each time. This back tracking is necessary because without it the maximum length between qubits having data may increase as operations are performed, which means that the number of necessary qubits (space resource of quantum computation) cannot be bounded. We propose circumventing this data diffusion by introducing a special region of qubits, called a *bundle* in this paper. The data diffusion can be avoided without back tracking, if we perform all meaningful operations in the bundle. The upper bound of the length of operations (time resource) obtained by our transform algorithm is about half of that by Lloyd, since back-tracks are abbreviated.

Lloyd's model can be naturally extended by changing the number of types of qubits. The number will be called the "period," and the extended model will be called the quantum-cellular-automata-like architecture (QCALA). (For example, the period of the model $ABCABC\cdots$ is three.) We will give the transform algorithm for the QCA-like architecture. The transform algorithm for Lloyd's model will be omitted, because it is just a special case of the algorithm for the QCA-like architecture. The relations between the period and the upper bound of time-space resources will be clarified. Counterintuitively, when the period gets large, the upper bound of the length of operations does not get small monotonically. The upper bound gets small if we can select an adequate number as the period, whereas it often gets large even if we make the period large. The upper bound of the number of qubits (space resource) and the period is also clarified. It is always bounded by three times of the number of qubits of the original quantum circuit, independently from the period; however, it gets small if we can select an adequate number as the period. The clarification of the relations between the period and the upper bounds of time-space resources is one of the main results of our paper.

II. LOGIC OPERATIONS IN A QCA-LIKE ARCHITECTURE

A. Physical model

The simplest example of the physical systems we study is a one-dimensional sequential polymer of the $ABCABC\cdots ABC$ type, which has been considered as a potentially realizable quantum computer by Lloyd [1]. Each atom A, B, or C is a two-level system representing a qubit (quantum bit) with energy f^A, f^B, or f^C, respectively, where f^i $(i=A,B,C)$ represents the resonant frequencies when none of them are coupled.

In his original proposal [1], each atom is assumed to have an additional excited state, which is used for error correction. The third level and the associated error correction feature have been totally eliminated here for simplicity. The field of quantum error-correction and fault-tolerant quantum computation (see, for example, Sec. 10 of Ref. [9], and references therein) have been much developed since 1995. In view of these developments, the third level scheme must eventually be replaced with a legitimate quantum error-correction scheme. However, it is out of the scope of this paper. Incorporating fault tolerance into the quantum circuit on the realistic physical model would be a significant future subject.

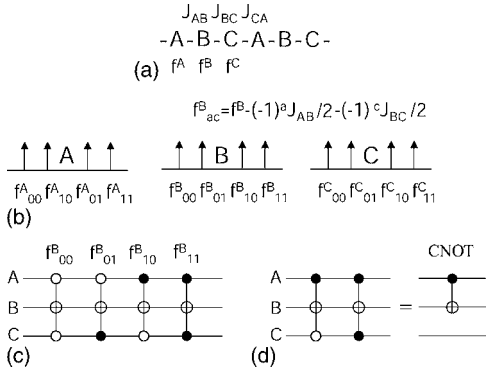

FIG. 1. Physical model (a) $(ABC)_n$-type sequential polymer with only neighboring interactions, (b) spectral splittings due to interactions with neighboring qubits, (c) line-selective π pulse as a *classical* controlled-controlled-NOT, and (d) construction of CNOT.

Only the neighboring atoms are assumed to have interactions. For example, B is coupled with its left neighbor A by interaction energy J_{AB} and its right neighbor C by J_{BC}. If the interaction Hamiltonian is diagonalized in the computational basis, the resonant frequency of atom B when A is in $|a\rangle$ and C is in $|c\rangle$ can be written as

$$f^B_{ac} = f^B - \frac{1}{2}(-1)^a J_{AB} - \frac{1}{2}(-1)^c J_{BC}, \quad a=0,1, \quad c=0,1.$$

The spectrum of an atom is split into four lines corresponding to the states of its two neighboring atoms as shown in Fig. 1(b).

When a qubit is represented by a spin-half or two-level system, quantum operation corresponding to the classical NOT logic can be performed by applying a resonant π pulse. When the line-selective π pulse is applied at f^B_{11}, B is flipped (negated) if A and C, which are adjacent to B, are both in $|1\rangle$. This is *classically* a controlled-controlled-NOT (CCNOT or Toffoli) gate, where A and C are the control qubits and B is the target qubit. Each line-selective π pulse corresponds to a different CCNOT as shown in Fig. 1(c), where the solid circle means the control is ON if the qubit is in $|1\rangle$ and the open circle means the control is ON if it is in $|0\rangle$. These line-selective pulses are similar but not exactly equivalent to *quantum* CCNOTs due to the differences in controlled phases. The differences can be compensated up to the total phase by applying additional pulses as shown in Appendix A.

A controlled-NOT (CNOT) can be constructed by combining two CCNOTs. For example, *classical* CNOT, where B is controlled by A, can be constructed by applying π pulses at f^B_{10} and f^B_{11} as shown in Fig. 2(d). This can be done sequentially in any order since the two operations commute. Also, it can be done simultaneously by applying two line-selective pulses at the same time or by applying a band-selective pulse that excites the two lines without disturbing other lines. Care must be taken when using band-selective pulses so as not to stimulate other lines, including the lines of the end atoms, which we will discuss later. Again by applying additional pulses, those CNOTs can be made equivalent to *quantum*

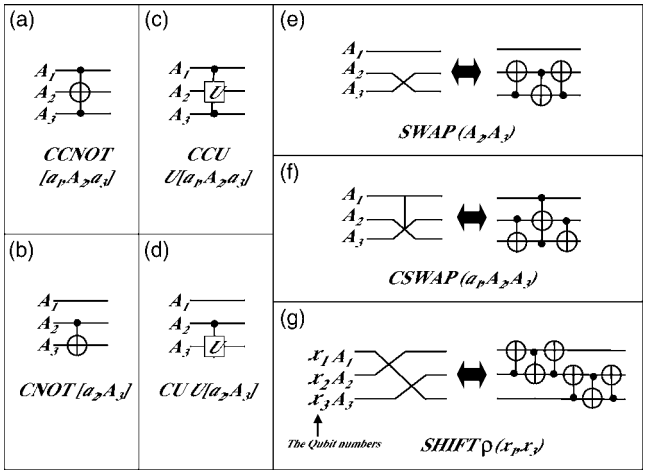

FIG. 2. Operations on a QCALA, where (a)–(d) are the basic operations and (e)–(g) are derived ones. The target bit(s) and the control bit in each operation (a)–(d) must be adjacent. The target bits in the CCNOT and CCU operations (a) and (c) must be between the control bits. Operations (a)–(f) are denoted by the types of qubits. However, the SHIFT operation is described by using the qubit numbers (absolute coordinates of the qubits).

CNOTs up to total phase as shown in Appendix A.

In Appendix A, we show that any controlled^2-U and controlled-U operations can be realized up to total phase by applying line-selective and additional pulses. CCNOT and CNOT are the special cases of $U=X$. Therefore, we will use these operations as elementary building blocks for our implementation.

Although we have assumed interactions only between neighboring atoms, the model is also applicable to cases where interactions between non-neighboring atoms exist but are much weaker than those between neighboring ones. This can be done by replacing each line-selective pulse with an appropriate band-selective pulse that covers the range of fine splittings caused by weak non-neighboring interactions. Since the modification can be done physically by changing the pulse design and will not affect the discussion afterwards, we can follow the simplest model of neighboring interactions without much loss of applicability.

The atom at the end of the polymer has only one interacting neighbor and therefore has different energy from atoms of the same type in the inner units. The first A has $f^A_{eb}=f^A-\frac{1}{2}(-1)^b J_{AB}$ when the neighboring B is in state $|b\rangle$. The last C has $f^C_{be}=f^C-\frac{1}{2}(-1)^b J_{BC}$ when the neighboring B is in $|b\rangle$. This does not mean that the end-qubit atoms are chemically unterminated. They can be terminated with *silent* atoms, which have no interaction with them. Since each end qubit has a distinct frequency, it can be flipped or rotated independently and arbitrarily without affecting inner units. It can be used for loading quantum data into the circuit as described by Lloyd [1]. For data loading, we will follow Ref. [1] and will not discuss it further.

B. Physical applications

The model has a significant application in quantum computation experiments using molecules under nuclear magnetic resonance (NMR) [10,11]. Nuclear spin is one of the most promising candidates for a qubit because of the relatively long coherence time and suitable couplings for quantum operations between qubits. A nucleus with spin-half, such as ^{1}H, ^{19}F, ^{31}P, ^{13}C, or ^{15}N, can be used as a qubit and manipulated by the NMR technique. In this scheme, a molecule in a static magnetic field works as a quantum computer. Each qubit is addressed by its magnetic resonance frequency. Any single qubit operation can be performed as rotation(s) of spin by resonant rf magnetic field pulse(s). Two-qubit operation can be performed using the interaction between two nuclear spins, which is called J coupling.

This scheme is so far the most successful in demonstrating quantum algorithms. Experiments with a few qubits have been intensively exploited and the factoring of 15 by Shor's quantum algorithm [12] has been recently demonstrated using 7-qubit molecules [13].

In a small molecule with up to several qubits, all qubits may have distinct frequencies and direct J couplings so that the qubit can be easily specified and two-qubit operation on any pair of qubits may be performed easily. However, as the number of qubits increases, frequencies get closer or degenerate because the available spectral resource is limited. It is difficult to find a molecule with many distinguishable qubits. Also, J couplings between qubits get weaker or disappear. Both of these factors not only lead to slower operations but also require a different strategy for implementing a quantum circuit on a molecule.

The model discussed here is suitable for implementing a large-scale quantum circuit on a molecule since it requires only a finite number of frequencies, regardless of the number of qubits, and only neighboring couplings. However, the application of the model is not limited to the NMR quantum computation and therefore the specific problem concerning the NMR is not discussed in this paper.

C. Generalization and notations

Here we consider a general structure of the above ABC structure, which we call the QCA-like architecture (QCALA) as follows.

(1) The structure is a repetition of m qubits $A_1,\ldots,A_m$, which is expressed as $A_1,A_2,\ldots,A_m,\ldots,A_1,A_2,\ldots,A_m$.

(2) The interaction between qubits exists only between neighboring qubits.

(3) We can perform only the same operation to the same type of qubits.

For logical operations, we consider the following four operations.

(a) An operation that performs CCNOT operation whose target bit is A_j and the control bits are A_i and A_k (A_i, A_j and A_k must be placed in this order) to all the sequences. This operation is denoted by $[a_i,A_j,a_k]$ or $[a_k,A_j,a_i]$. (The target bit is always denoted by a capital letter.)

(b) An operation that performs CNOT operation whose target bit is A_i and the control bit is A_j (A_j must be the neighbor of A_i) to all the sequences, i.e., CNOT gates are operated onto all the pairs of A_i and A_j at the same time. This operation is denoted by $[A_i,a_j]$ or $[a_j,A_i]$.

(c) An operation that performs controlled-controlled-U

(CCU) operation whose target bit is A_j and the control bits are A_i and A_k (A_i, A_j and A_k must be placed in this order) to all the sequences. This operation is denoted by $U[a_i,A_j,a_k]$ or $U[a_k,A_j,a_i]$.

(d) An operation that performs controlled-U (CU) operation whose target bit is A_i and the control bit is A_j (A_j must be the neighbor of A_i) to all the sequences, i.e., CU gates are operated onto all the pairs of A_i and A_j at the same time. This operation is denoted by $U[A_i,a_j]$ or $U[a_j,A_i]$.

By using the above operations, we can perform the following operations.

(e) A swap operation (SWAP) that swaps bits on the adjacent qubits (e.g., A_i and A_j, where A_j must be the neighbor of A_i). This can be done by three CNOT operations. SWAP is denoted by (A_i,A_j) or (A_j,A_i), which is decomposed to $[A_i,a_j][a_i,A_j][A_i,a_j]$ or $[a_i,A_j][A_i,a_j][a_i,A_j]$.

(f) A controlled swap operation (CSWAP) that swaps bits on the adjacent qubits (e.g. A_j and A_k, where A_j must be the neighbor of A_k) depending on the state of the adjacent qubit A_i. This can be done by two CNOT operations and one CCNOT operation. CSWAP is denoted by (a_i,A_j,A_k). (The swapped bits are always denoted by capital letters. (a_i,A_j,A_k) is decomposed to $[a_j,A_k][a_i,A_j,a_k][a_j,A_k]$.)

(g) A shift operation (SHIFT) that moves a bit on a qubit onto another qubit. This can be done by SWAP. For example, let x,y,z be the qubit numbers of qubits A_i,A_{i+1},A_{i+2}, respectively. Here, the "qubit number" means the absolute coordinate of the qubits. (Thus, x,y,z are successive numbers.) Then, SHIFT $\rho(x,z)$ is defined as $(A_i,A_{i+1})(A_{i+1},A_{i+2})$. After this operation, a bit sequence k_1,k_2,k_3 on A_i,A_{i+1},A_{i+2} will be changed to k_2,k_3,k_1. SHIFT $\rho(z,x)$ is defined as $(A_{i+2},A_{i+1})(A_{i+1},A_i)$. After this operation, a bit sequence k_1,k_2,k_3 on A_i,A_{i+1},A_{i+2} will be changed to k_3,k_1,k_2.

Generally, let $x_0,x_1,\ldots,x_{jm+l}$ be the qubit numbers of qubits $A_i,\ldots,A_{(i+l \bmod m)}$, respectively, where $j \geqslant 0$ and $0 \leqslant l \leqslant m-1$. SHIFT $\rho(x_0,x_{jm+l})$ is defined as $(A_i,A_{(i+1 \bmod m)})\ldots(A_{[i+j(m-1)+l-1 \bmod m]},A_{[i+j(m-1)+l \bmod m]})$. SHIFT $\rho(x_{jm+l},x_0)$ is defined as $(A_{(i+l \bmod m)},A_{(i+l-1 \bmod m)})\ldots(A_{[i-j(m-1)+1 \bmod m]},A_{[i-j(m-1) \bmod m]})$.

Our essential task is to transform a normal quantum circuit into a sequence of the above seven logic operations (a)–(g). This is described in the next section.

III. IMPLEMENTATION AND UPPER BOUND

It is well known that any quantum circuit can be implemented by CNOT gates and single-qubit gates. However, as we saw in the previous section, CNOT gates and single-qubit gates are not the available (primitive) operations on the QCALA [1], and, therefore, we need to transform CNOT gates and single-qubit operations to a sequence of the available (primitive) operations on the QCALA.

Although Lloyd showed that we can realize a universal quantum computer on the one-dimensional sequential polymer of the $ABCABC\cdots ABC$ type [1], it has not yet been clarified how we can translate a given quantum circuit to an efficient operation sequence on the general QCALA. Thus we show an efficient algorithm in this section.

A. Intuitive description of the implementation

In our method, for each CNOT gate (the situation is almost the same for a single-qubit gate) we perform the following.

(1) By using SWAP and CSWAP operations, we move the contents of qubits so that the contents of the special bit (explained later) and the control and the target bits of the given CNOT gate are placed on the adjacent qubits.

(2) In our algorithm, as we will explain later, it is guaranteed that the content of the special bit is always 1, whereas the other contents on A_k are 0's when the content of the special bit is on A_k. Therefore, we perform the desired CNOT gate by CCNOT whose control bit is the special bit since only the CCNOT whose control bit is the special bit becomes effective.

Note that, for better exposition, we often say "we move two qubits so that they are adjacent" instead of "we move the contents of two qubits so that the content of two qubits are on the two adjacent qubits."

Although our whole strategy itself is similar to the one in Ref. [1], our method needs only half the number of operations as the method implicitly denoted in Ref. [1]. This is because we carefully omit essentially unnecessary movement of qubits in the method in Ref. [1] by introducing the concept of the "bundle," which will be explained in the following section.

B. Formal description of the implementation

The formal description of the input to our translation algorithm is as follows.

Input. An n-qubit quantum circuit $\mathcal{C}$ consisting of d gates, i.e., $\mathcal{C}=G_1,G_2,\ldots,G_d$, where each G_i is a CNOT gate, single-qubit gate, or controlled-U gate. Each qubit on $\mathcal{C}$ is labeled by one of numbers $1,2,\ldots,n$.

Although any quantum circuit can be implemented by CNOT gates and single-qubit gates only, we assume that we also have controlled-U gates in initial quantum circuits. The reason is that controlled-U gates are considered to be almost the same on the QCALA in terms of the implementation cost, and they are sometimes useful to describe a quantum algorithm concisely. Note that any quantum algorithm can be converted to a circuit consisting of CNOT gates, single-qubit gates, and controlled-U gates.

Let t_i and c_i be the target bit and the control bit of G_i, respectively, i.e., $1 \leqslant t_i,c_i \leqslant n$. We will introduce an algorithm to translate C to a sequence of operations on a QCALA, named $\mathcal{S}$, $A_1,A_2,\ldots A_m,\ldots A_1,A_2,\ldots,A_m$. The output of the algorithm can be described as follows.

Output. O, which is a sequence of operations on $\mathcal{S}$ (O is the translated result), and V, which is an array with $n+1$ variables. V specifies the correspondence between *the qubit numbers* (which are used to denote the locations of qubits) on $\mathcal{C}$ and $\mathcal{S}$. That is, $V[j]$ is the qubit number on $\mathcal{S}$ corresponding to the jth qubit on $\mathcal{C}$. By referring to V, we can assign each qubit on $\mathcal{C}$ to the corresponding qubit on $\mathcal{S}$.

In our algorithm, we select one special qubit on S, and call it the 0th qubit. This special qubit works as a base to express the locations of qubits, i.e., each qubit is labeled by a unique integer, called the qubit number, according to the

distance from the 0th qubit. Qubits that are on the right side of the 0th qubit are labeled by positive integers, and ones on the other side by negative integers.

We say that the xth and yth qubits are in the same group on $\mathcal{S}$ if they are on the same type of A_k: In other words, they are in the same group if and only if $x=y \bmod m$ (period). We denote $x \sim y$ if the xth and yth qubits are in the same group.

The 0th qubit is initially set to 1, and this special bit is called "the head." The content of this special bit will be moved during the operations and used as a control bit for CSWAP operations as we will see. Thus, the qubit number of the head is initially 0, and will be changed. We refer to this number as "the head number," and this is stored in $V[0]$ in our algorithm. The other qubits in the same group of the head are set to 0. Therefore, we can perform a SWAP operation only at the desired location, i.e., at the head, by using a CSWAP operation. We use the other groups for storing the inputs (data) of a given quantum circuit. That is, we assign $x_1, x_2, \ldots, x_n$ of the given quantum circuit to each qubit on $\mathcal{S}$ one by one from the right side of the 0th qubit, but we do not use qubits in the same group as the 0th qubit. None of the other qubits are used, i.e., they are just garbage. Thus, the initialization of O and V can be written as in Appendix B.

Our algorithm takes each gate on $\mathcal{C}$, and translates it into a sequence of operations on $\mathcal{S}$. Translated sequences are added one by one to the initial O, and we maintain V so that it holds the appropriate correspondence between the qubits on $\mathcal{C}$ and $\mathcal{S}$.

While moving (the content of) a qubit to the desired location, if we perform CSWAP operations without any consideration, (the content of) a qubit storing data and (the content of) a qubit without data (garbage qubit) may be swapped. Then, there is a chance that the data contents of qubits will be spread widely on $\mathcal{S}$. In such a case, we cannot upper bound the necessary numbers of qubits on $\mathcal{S}$ and operations such as theorem 1 and theorem 2 in the next section.

To remedy the above situation, we introduce a special concept called "the bundle." The bundle is a region of m (which is equivalent to the period) adjacent qubits such that it is guaranteed by our algorithm that the bundle always contains the head and all the qubits in the bundle contain data. Then we restrict ourselves to perform CSWAP operations only in the bundle. Accordingly, our algorithm guarantees that qubits for storing data stay in some region on $\mathcal{C}$ during the whole operation.

During the translation, we may need to move the bundle because the head might be moved; therefore, we use an array B to store a pair of numbers $\{B^-, B^+\}$ such that $B^+ - B^- = m - 1$. (In other words, B has the information about the bundle, i.e., the xth qubit is in the bundle if $B^- \leqslant x \leqslant B^+$.) B is initialized as $\{0, m-1\}$.

In contrast to our strategy, that in Ref. [1] always moves qubits back to their original positions after simulating one original CNOT gate. By using the bundle concept, we do not need such restorations. Therefore, our algorithm produces more efficient results. (We will see this in theorem 2 in the next section.)

Now we are ready to explain our algorithm. The formal description of the algorithm is in Appendix B. We just give an intuitive explanation in this section. The algorithm splits into three cases according to the gate being translated; the procedure is different for CNOT gates, single-qubit gates, or controlled-U. We first explain the case where the target gate is a CNOT gate.

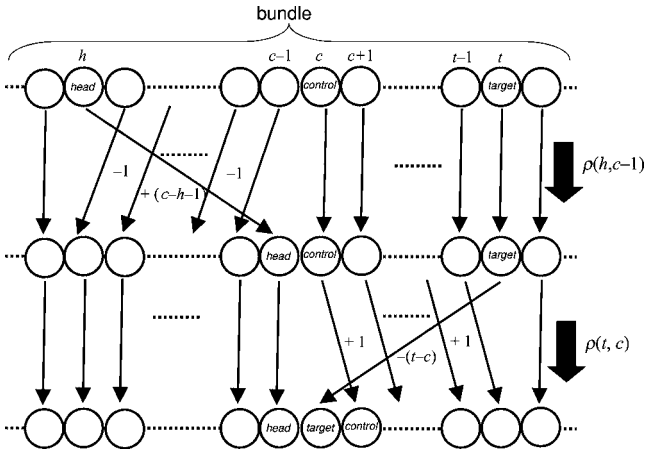

FIG. 3. This shows the case when all the head, control, and target bits are in the bundle (the situation for step IV), and $h<c<t$, where h, c, and t are the qubit numbers for the head, control, and target bits, respectively. In the upper part of the figure, qubits are moved by $\rho(h, c-1)$. By this operation, the qubits in the same group as the head are moved right by $c-h-1$, and the qubits in the same group as the qubits between the head and the control bit are moved left by one. The arrows in the figure show how the contents of the qubits move, and the number attached to an arrow shows the change of the qubit number for the content. At the lower part of the figure, we move the target bit to the left of the control bit by $\rho(t, c)$.

Case: CNOT *gate*. To translate a CNOT G_i whose target and control bits are t_i and c_i on $\mathcal{C}$, respectively, into a sequence of operations on $\mathcal{S}$, our algorithm has four steps according to the situation.

Step IV. The easiest situation is the case treated by step IV where the target bit ($V[t_i]$) and the control bit ($V[c_i]$) on $\mathcal{S}$ are already in the bundle as shown in Fig. 3. In this case, we just move qubits by SWAP operations so that the head, the control bit, and the target bit are placed adjacently in this order. Then, we perform the original CNOT gates by using a CCNOT operation on $\mathcal{S}$. We need not change the bundle in this step because all the qubits in the bundle remain in it (they are just swapped with each other).

Define h, c, t as $h := V[0], c := V[c_i], t := V[t_i]$. Then split the translation algorithm into six cases according to the order of them as in Appendix B.

We explain case $h<c<t$, which is illustrated in Fig. 3. First, we move the head so that it is next to the control bit. This is done by $\rho(h, c-1)$. By this operation, the qubits in the same group as the head move right by $c-h-1$, and the qubits in the same group as the qubits between the head and the control bit move left by one. (The arrows in the figure show how the contents of the qubits moves.)

Next, we move the target bit to the left of the control bit. This is done by $\rho(t, c)$. By this operation, the qubits in the same group as the target bit move left by $t-c$, and the qubits in the same group as the qubits between the target bit and the head [after $\rho(h, c-1)$] move right by one.

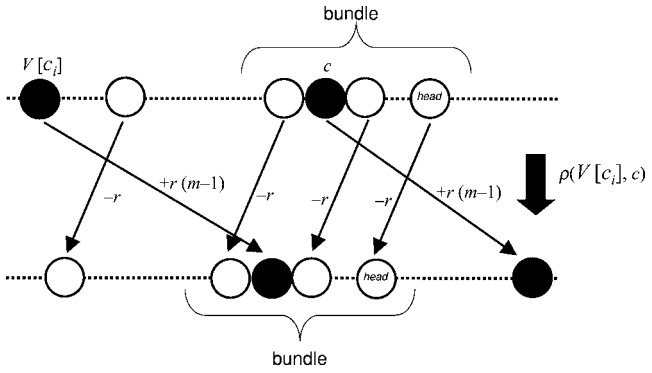

FIG. 4. This shows the case where we want to move the control bit into the bundle (step II). Let c be a number such that c and $V[c_i]$ are the qubit numbers for the same group, and the cth qubit is in the bundle. The figure shows the case when $V[c_i]<c$. By the operation $\rho(V[c_i],c)$ (the contents of) the qubits of the same group as the $V[c_i]$-th qubit and the c-th qubit (denoted by black circles) are moved to the right by $r(m-1)$, and the others to the left by r, where $r=(c-V[c_i])/m$. Then, we should move the bundle to the left by r for the bundle condition.

The other cases are similar to the $h<c<t$ case. (The formal description of the translation results can be found in Appendix B.)

Steps II and III. If the control bit is not in the bundle, we need to move the bit into it before step IV. This is done by step II. Likewise, if the target bit is not in the bundle, we need to move it there before step IV. This is done by step III. It seems we are able to combine steps II, III, and IV: we just move the target bit and the control bit so that they are next to the head by SWAP operations. However, we have the bundle condition and therefore need to move the bundle for steps II and III. Accordingly, we describe steps II, III, and IV separately.

Here we explain step II by using Fig. 4. (Step III is similar to step II.) Suppose we want to move the control bit ($V[c_i]$)) to the bundle. Let c be a number such that c and $V[c_i]$ are the qubit numbers for the same group, and the cth qubit is in the bundle. (For better exposition, we often say "c" instead of "the cth qubit" if the context is clear.) Then, what we should do is to move $V[c_i]$ to c as denoted in the figure. Therefore, we need to perform $\rho(V[c_i],c)$, and this is added to (the translated operation sequence) O. Here we consider the case when $V[c_i]<c$. By the operation $\rho(V[c_i],c)$, (the contents of) the qubits of the same group as $V[c_i]$ and c (denoted by black circles) are moved to the right by $r(m-1)$, and the other qubits are moved to the left by r, where $r=(c-V[c_i])/m$. Then, we should move the bundle to the left by r for the bundle condition. Accordingly, we revise B and V in the formal description in Appendix B.

Step I. If $V[c_i]$ and $V[t_i]$ are in the same group, the contents on $V[t_i]$ and $V[c_i]$ move together by SWAP operations. So we cannot move qubits so that the head, the target and the control bits are placed adjacently by steps II–IV. Therefore, if we encounter such a case, we move one of them onto a different group by using CSWAP. This is done by step I.

We choose the target or control bit that is nearer to the head. Let this qubit number be y. That is, y is defined to be

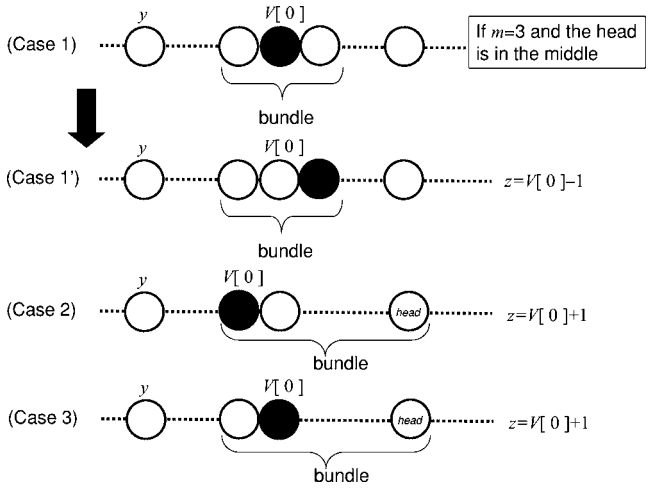

FIG. 5. This shows the three cases for step I. Case 1 is that when $m=3$ and the head is in the middle of the bundle. If we encounter this case, we first swap the head with the right qubit to get case 1′ (see the text for the reason). Then we will move the yth qubit to the zth qubit, where the yth qubit is either the control or the target bit that we will move to the next $V[0]$ (the head, denoted by black circles) in step I, and z is defined as follows to satisfy the bundle condition: $z=V[0]+1$ if the head is the leftmost or the or 3). In other cases, $z=V[0]-1$, that is, we move the yth qubit to the left of the head (see the text for more details about y).

$V[t_i]$ if $|V[t_i]-V[0]|\leq|V[c_i]-V[0]|$, and $V[c_i]$ otherwise. What we should do first is to move the yth qubit to the right or left neighbor of the head, and swap the content of the qubit to a qubit in another group by a CSWAP operation.

Consider case 1 in Fig. 5, where $m=3$ and the head is in the middle of the bundle. In this case, we cannot perform the above CSWAP operation in the bundle because one of the qubits swapped by CSWAP is not in it. Therefore, if we encounter case 1, we first swap the head with the right qubit to get to case 1′.

After the above modification (case 1 to case 1′), we move the yth qubit to the zth qubit, where z is defined as follows to satisfy the bundle condition (Recall that we should perform CSWAP operations in the bundle): $z=V[0]+1$ if the head is the leftmost or the second leftmost in the bundle (case 2 or case 3 in Fig. 5), and $z=V[0]-1$ otherwise. Thus, we revise O as $O\rho(y,z)$ as described in Appendix B.

For the revision of B and V, the algorithm is split into two cases according to the order of y and z. Here, we explain the case $y<z$ using Fig. 6.

By the operation $\rho(y,z)$, qubits are moved in three ways according to the qubit groups as follows. Here we define $z':=y+m\lceil(z-y)/m\rceil$ and $z''::=y+m\lfloor(z-y)/m\rfloor$, and use z' and z'' to specify qubit groups. [z' and z'' are the qubit numbers for the two qubits such that (i) they are on the same type of qubit as the yth qubit (denoted by black circles in Fig. 6), (ii) they are nearest to the head, and (iii) they are on the right-hand and the left-hand sides of the head, respectively.]

Group 1. The qubits in the same group as the yth qubit.

Group 2. The qubits in the same group as the wth qubit, where $z<w<z'$.

Group 3. The qubits in the same group as the wth qubit, where $z''<w\leq z'$.

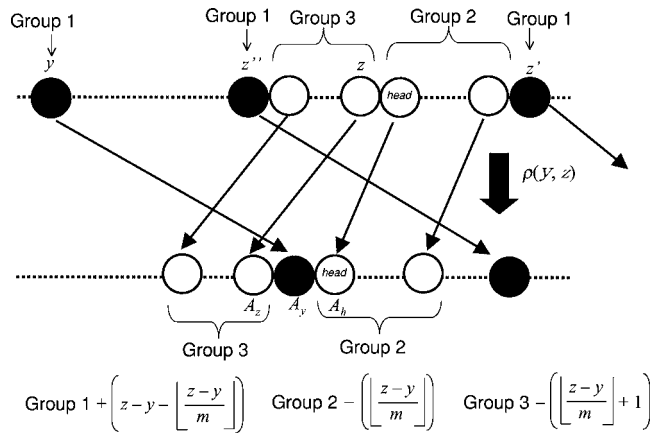

FIG. 6. This shows the case $y<z$ for step I, where we want to move the yth qubit to the zth qubit (the left of the head). This is done by $\rho(y,z)$, and the contents of qubits are moved in three ways. Here we define $z' := y+\lceil(z-y)/m\rceil$ and $z'' := y+\lfloor(z-y)/m\rfloor$, and use z' and z'' to specify qubit groups as follows. The qubits in group 1 are those in the same group as the yth qubit. They are moved right by $z-y-\lfloor(z-y)/m\rfloor$. The qubits in group 2 are those in the same group as the wth qubit, where $z<w<z'$. They are moved left by $\lfloor(z-y)/m\rfloor$. The qubits in group 3 are those in the same group as the wth qubit, where $z''<w\leqslant z'$. They are moved left by $\lfloor(z-y)/m\rfloor+1$.

Between the yth qubit and the head, there are $z-y$ qubits, and among them there are $\lfloor(z-y)/m\rfloor$ qubits of the same group of the yth qubit. Therefore, by $\rho(y,z)$, the yth qubit is swapped with the qubits in the other group $z-y-\lfloor(z-y)/m\rfloor$ times. Therefore, the qubit in group 1 moves right by $z-y-\lfloor(z-y)/m\rfloor$. The qubits in group 2 are swapped with those in group 1 exactly one less time than those in group 3. Thus, the qubits in group 2 move left by $\lfloor(z-y)/m\rfloor$, and those in group 3 move left by $\lfloor(z-y)/m\rfloor+1$.

According to the above movement of qubits, we change V in order to maintain the consistency of the correspondence between the qubit numbers on $\mathcal{C}$ and $\mathcal{S}$ as written in Appendix B.

For the bundle condition, we need to move the bundle as the head moves, i.e., to the left by $\lfloor(z-y)/m\rfloor+x$, where x is defined as follows: $x=1$ if the zth qubit is right of the head, and $x=0$ if it is left of the head. (Note that the head is in group 3 if $x=1$, and it is in group 2 if $x=0$.)

By $\rho(y,z)$, the yth qubit is moved to the neighbor of the head. In the situation after $\rho(y,z)$, let A_h, A_y, and A_z be the type of the qubit holding the head, the yth qubit and the other neighbor of the yth qubit other than the head, respectively. (The neighbors of A_y are A_h and A_z as shown in the lower part of the figure.) What we need to do next is to move the yth qubit to another qubit group by a CSWAP operation. Therefore, we further revise O as $O := O(A_y,A_z,a_h)$ and maintain V as written in Appendix B.

Case: controlled-U gate. If the target gate is a controlled-U gate, the last operation for the CNOT gate case, i.e., the operation $[a_{k-1},A_k,a_{k+1}]$ in step IV should be replaced with controlled-controlled-U operation $U[a_{k-1},A_k,a_{k+1}]$. Therefore, step IV is modified to step IV′ in this case.

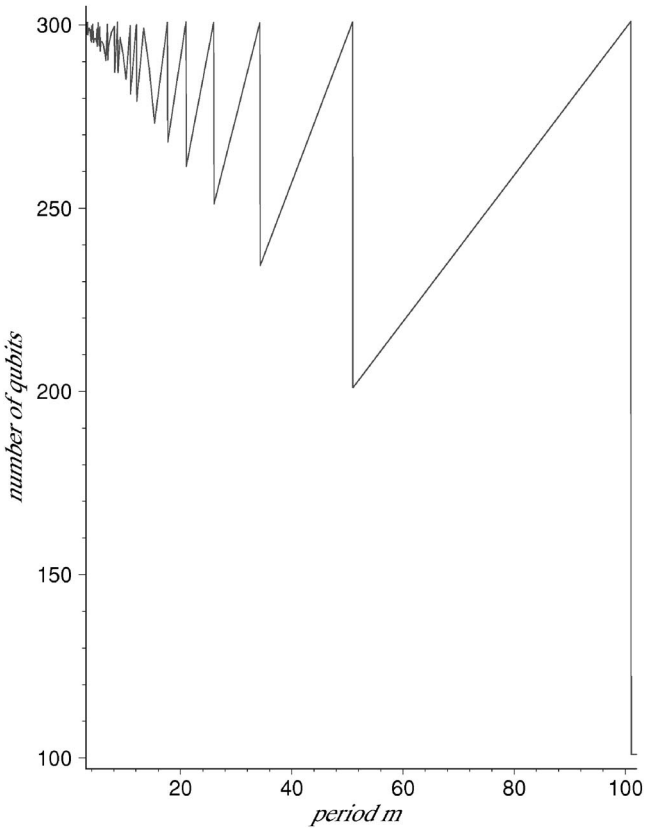

FIG. 7. This shows the relationship between the upper bound of the number of the qubits and the period m, when a 100-qubit quantum circuit is translated to a QCALA. The number of qubits that we need is bounded by $n+1+2(m-1)\{\lceil n/(m-1)\rceil-1\}$, which is less than $3n$, by theorem 1. So, the graph is bounded by 300 (=3 $\times$100).

Case: single-qubit gate. If the target gate is a single-qubit gate, there is no control bit. Therefore, we need neither step I nor step II. Also we do not need to move the control bit in step IV, and step IV is modified to step IV′ in this case.

C. Upper bound of the number of operations

Let $\mathcal{C}$ be an n-qubit quantum circuit. Let S be a QCALA $A_1,A_2,\ldots,A_m,\ldots,A_1,A_2,\ldots,A_m$. Let O be the sequence of operations output by the algorithm.

Lemma 1. The bundle contains the only qubit in each group during the operations by O.

Proof. This is obvious by the definition of the translation algorithm.

Lemma 2. For each group, the maximum distance between qubits in the group is $m(\lceil n/(m-1)\rceil-1)$.

Proof. In the initial state of operations, the maximum distance between qubits in a group is $m(\lceil n/(m-1)\rceil-1)$. By the definition of the translation algorithm, the size holds consistently.

Theorem 1. During the operations by O, the head moves in the range of $[0,n]$, and all data bits move in the range of $[-\delta,n+\delta]$, where $\delta=(m-1)(\lceil n/(m-1)\rceil-1)$. We thus need at most $n+1+2(m-1)(\lceil n/(m-1)\rceil-1)$ qubits. It is always less than $3n$ since $\delta<n$ (see Fig. 7).

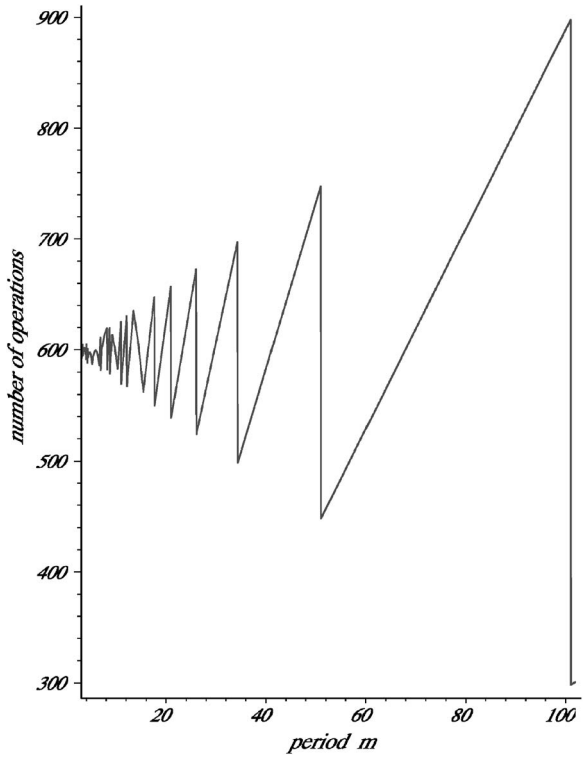

FIG. 8. This shows the relationship between the upper bound of the number of the translated operations and the period m, when one CNOT (or controlled-U) gate on a 100-qubit quantum circuit is translated to a QCALA. The number of operations is bounded by $6(m-1)\lceil n/(m-1)\rceil-3m+1$ by theorem 2. It is about six times the number of the qubits in the original circuit if m is small. When one single-qubit gate on a 100-qubit quantum circuit is translated to a QCALA, the upper bound of the translated operations is much smaller than this.

Proof. The head number has the minimum value when the head is in the leftmost position of all data bits, e.g., the initial state. So, the minimum value of the head number is 0. The head number has the maximum value when all data bits move left and they pass the head. The head moves right by one if a data bit passes the head by SWAP. The number of the data bits is n, so the maximum value of the head number is n.

The left-most qubit number used in the operations of O is obtained as follows. We consider the case that the $(m-1)$-th qubit in the initial state moves to the left-most position, because the left-most qubit is used in this case. Then, the head is on the $\lceil n/(m-1)\rceil$-th qubit. The rightmost data bit in the group is adjacent at the left side of the head, so it is on the $(\lceil n/(m-1)\rceil-1)$-th qubit. The distance between the leftmost and the rightmost data bits in the group is $m(\lceil n/(m-1)\rceil-1)$. Hence, the leftmost data bit in the group is on the $(\lceil n/(m-1)\rceil-1-m(\lceil n/(m-1)\rceil-1))$-th qubit. It is $-\delta$. The rightmost qubit used in the operations of O is obtained in the same way.

Theorem 2. When $m\leqslant n$, a single-qubit gate in $\mathcal{C}$ is translated to a sequence of at most $3(m-1)\lceil n/(m-1)\rceil+1$ logic operations, and a two-qubit gate (CNOT or controlled-U) in $\mathcal{C}$ is translated to a sequence of at most $6(m-1)\lceil n/(m-1)\rceil-3m+1$ logic operations. When $m>n$, each gate is translated to a sequence of at most $3n-2$ logic operations (see Fig. 8).

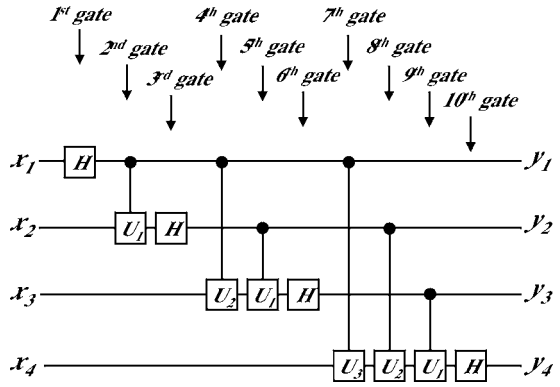

FIG. 9. This shows a circuit of discrete quantum Fourier transform DFT_{16} on four qubits. Single-qubit gate H is the Hadamard gate, and U_j is the control phase shift by $2\pi/2^j$. The output values are described as $y_1,\ldots,y_4$ when $x_1,\ldots,x_4$ are input to the circuit. Each gate is labeled by a number 1, …, 10 from the left-hand side. When we translate this circuit by the translation algorithm, the sequence of operations represented in Fig. 10 is obtained.

Proof. When $m>n$, all data are contained in the bundle, so we consider only step IV, IV′, or IV″. They need $3(n-1)+1$ operations. Hence, the number of the operations is bounded by $3n-2$.

Suppose $m\leqslant n$. When the gate that we want to translate is a single-qubit one, we calculate the maximum number of operations in steps III and IV″. Step III needs at most $3(m-1)(\lceil n/(m-1)\rceil-1)$ operations. Step IV″ needs at most $3(m-1)+1$ operations. Hence, at most $3(m-1)\lceil n/(m-1)\rceil+1$ operations are necessary in this case.

When the gate G_i that we want to translate is a two-qubit one, we count the maximum number of operations such that $V[c_i]$ and $V[t_i]$ are not in the same group, because this is the worst case. Step I is then skipped. Both steps II and III need at most $3(m-1)(\lceil n/(m-1)\rceil-1)$ operations. Step IV (IV′) needs at most $3(m-2)+1$ operations. We thus need at most $6(m-1)\lceil n/(m-1)\rceil-3m+1$ operations for each two-qubit gate in the original quantum circuit.

D. Examples of translation

Figure 9 shows a quantum circuit for the quantum discrete Fourier transform DFT_{16} on four qubits. It has 10 gates. The gate represented by H is the Hadamard gate, i.e.,

$$H=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}.$$

The gate represented by U_j means the control phase shift by $2\pi/2^j$, i.e.,

$$U_j=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & e^{i\theta_j}\end{pmatrix},$$

where $\theta_j=2\pi/2^j$.

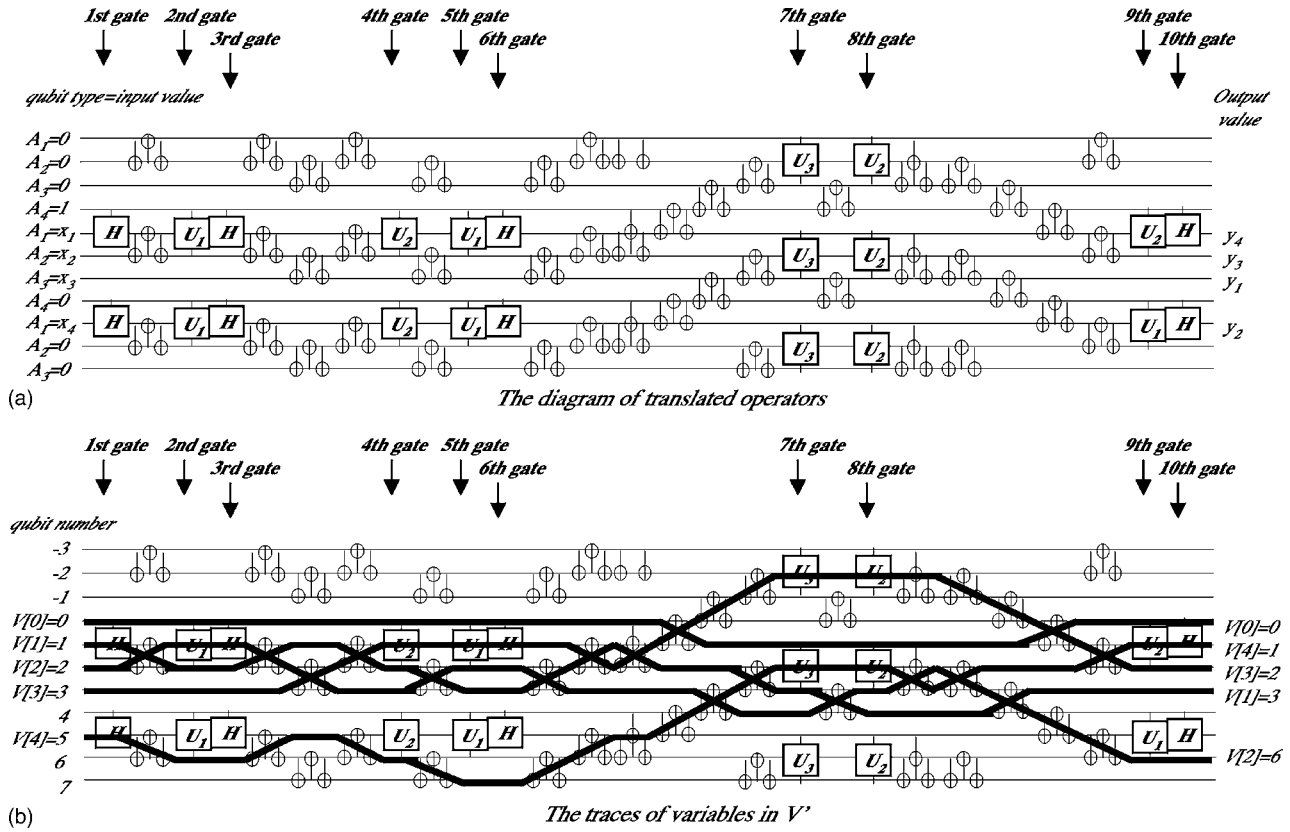

FIG. 10. This represents a sequence of operations translated from Fig. 9. [Panels (a) and (b) show the same sequence of operations.] Since $m=4$, $n=4$, and $\delta=3$ by theorem 1, using $11(=n+1+2\delta)$ qubits is sufficient for implementing the translated operations. The qubit numbers (absolute coordinates of the qubits) are set to $-3,-2,\ldots,6,7$. Each qubit is initialized as $0,0,0,1,x_1,x_2,x_3,0,x_4,0,0$, since the 0th qubit is always initialized as 1 and input values of Fig. 9 are input to ones having positive qubit numbers except for the same type qubits as the 0th qubit. The first to tenth gates in Fig. 9 correspond to the 1st, 5th, 6th, 16th, 20th, 21st, 40th, 44th, 60th, 61st operations in this figure, respectively. All of the remaining operators are used for moving variables. Panel (b) shows the traces of the head and variables $x_1,\ldots,x_4$ in Fig. 9. They are started from the qubit numbers 0, 1, 2, 3, 5 (shown by $V[0],\ldots,V[4]$ in the left-hand side terminal), and finally they are achieved at the qubit numbers 0, 3, 6, 2, 1 (shown by $V[0],\ldots,V[4]$ in the right-hand side terminal). Thus, by observing the values of the qubits at 3, 6, 2, 1 after the operations, we can obtain the same values of $y_1,\ldots,y_4$ in Fig. 9.

When we translate the circuit to a sequence of operations on a QCALA consisting of a repetition of four qubits, we obtain the following as the translation result:

$$H[a_4,A_1](A_1,A_2)U_1[a_4,A_1,a_2]H[a_4,A_1](A_1,A_2)(A_2,A_3)$$

$$(A_1,A_2)U_2[a_4,A_1,a_2](A_2,A_3)U_1[a_4,A_1,a_2]H[a_4,A_1]$$

$$(A_2,A_3)(A_1,A_2)(a_4,A_1,A_2)(A_4,A_1)(A_3,A_4)(A_2,A_3)$$

$$U_3[a_1,A_2,a_3](A_3,A_4)U_2[a_1,A_2,a_3](a_1,A_2,A_3)(A_2,A_3)$$

$$(A_3,A_4)(A_4,A_1)(A_1,A_2)U_1[a_4,A_1,a_2]H[a_4,A_1].$$

By theorem 2, the number of translated operations is at most $226[=4\times(3\times3\times\lceil\frac{4}{3}\rceil+1)+6\times(6\times3\times\lceil\frac{4}{3}\rceil-3\times4+1)]$, since DFT_{16} has four single-qubit gates and six two-qubit gates. However, it is only 61 in this case [see Fig. 10].

By theorem 1, the translation operations can be achieved on a 11-qubit QCALA, because $\delta=3$ $(=3\times(\lceil\frac{5}{3}\rceil-1))$. Figure 10 shows the sequence of translated operations.

IV. CONCLUSIONS

We have proposed an explicit implementation to translate a quantum circuit that consists of single-qubit, CNOT, and controlled-U gates to a sequence of operations on a QCALA. The translated operations is more efficient than that proposed by Lloyd [1,7].

When we compare the sequences of operations translated from a two-qubit gate in the case that the period m of the QCALA is three (i.e., $ABCABC\cdots ABC$), the sequence obtained by our strategy needs at most $6n$ operations (theorem 2), while the one obtained by Lloyd's strategy needs at most $12n$ operations. Here, n is the number of qubits for the original circuit. Thus, the upper bound of the length of operations obtained by our transform algorithm is about half of that by Lloyd.

Our translation strategy can be used in the case that m is larger than three, though Lloyd's papers did not give an explicit strategy to translate a quantum circuit to a QCALA with a period m larger than three. We clarified the relation between the upper bound of the number of translated operations and the period of the QCALA (theorem 2). The upper

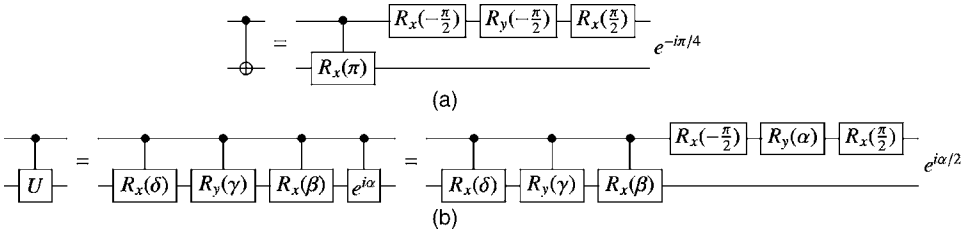

FIG. 11. Realizations of controlled gates; (a) CNOT and (b) controlled-U. The scalar phase factor indicated at the end of each quantum circuit is the total phase, which can be safely omitted in actual implementations.

bound gets small if we can select an adequate number as the period, whereas it often gets large even if we make the period large. However, it is always bounded by $9n$, which is less than the upper bound of one obtained by Lloyd's strategy. The upper bound of the translated operations in the case $n=100$ is figured in Fig. 8.

We also clarified the relation between the upper bound of the number of qubits and the period of the QCALA (theorem 1, Fig. 7). The number of qubits that we need is always less than $3n$, and gets small if we select a good m value.

ACKNOWLEDGMENTS

The authors thank Dr. Yasuhiro Takahashi and Dr. Sei-ichiro Tani for their helpful comments. M.K. would like to acknowledge CREST of JST (Japan Science and Technology Agency) for support.

APPENDIX A: IMPLEMENTATIONS OF ELEMENTARY QUANTUM GATES

We show that any controlled-U and controlled^2-U operations, which include CNOT and CCNOT as special cases of $U=X$, can be realized up to the total phase by applying line-selective and additional pulses in liquid state NMR. A single qubit NOT operation can be realized as a rotation about the x axis by an angle π by applying a π pulse on resonance. The actual unitary transformation performed $R_x(\pi)=-iX$ is not exactly a NOT $(=X)$ operation. However, the difference is only in the total phase factor $-i=\exp(-i\pi/2)$, which has no physical effect.

When two qubits, A and B, are J-coupled, it is known that controlled rotation, such as controlled-$R_x(\theta)$, can be realized by applying a line-selective weak pulse to the target qubit (B) on the transition frequency corresponding to the state of the control qubit (A) being $|1\rangle$ [14]. At this time, controlled-$R_x(\pi)$ is different from controlled-NOT by the conditional phase, which may have a major impact. However, it can be fixed rather easily by applying $R_z(\pi/2)=R_x(\pi/2)R_y(\pi/2)R_x(-\pi/2)$ to the control qubit as shown in Fig. 11(a).

This can be systematically generalized to any controlled-U by decomposing U as follows:

$$U=e^{i\alpha}R_x(\beta)R_y(\gamma)R_x(\delta), \tag{A1}$$

where $\alpha,\beta,\gamma,\delta\in[0,2\pi)$. Then controlled-$U$ can be decomposed into controlled-$e^{i\alpha}$, controlled-$R_x(\beta)$, controlled-$R_y(\gamma)$, and controlled-$R_x(\delta)$ as shown in the middle of Fig. 11(b). The controlled rotations can be realized as line-selective pulses. The controlled phase shift is equivalent [9] to total phase factor $e^{i\alpha/2}$ and $R_z(\alpha)=R_x(\pi/2)R_y(\alpha)R_x(-\pi/2)$ on the control qubit which can be realized by three pulses as shown in the right of Fig. 11(b).

CNOT can be realized as a special case of $\alpha=-\pi/2$, $\beta=\pi$, and $\gamma=\delta=0$ as already shown in Fig. 11(a).

Next, we consider the case of the three qubits, A, B, and C, linearly aligned with J couplings only between neighboring qubits as shown in Fig. 1. We consider the case that B is the target qubit and A and C the controlled qubits. By using the decomposition in Eq. (A1), controlled-controlled-U can be decomposed into controlled-controlled-$e^{i\alpha}$, controlled-controlled-$R_x(\beta)$, controlled-controlled-$R_y(\gamma)$, and controlled-controlled-$R_x(\delta)$ as shown in Fig. 12(a). The three doubly controlled rotations can be realized as line-selective pulses in a manner similar to the case of two qubits [8]. The doubly controlled phase shift can be decomposed into the total phase factor $e^{i\alpha/4}$ and $R_z(\alpha/2)$ on A and controlled rotation on C by A as shown in the middle of Fig. 12(b). Since A and C are not J coupled, the last part cannot be directly realized by the method described so far. We have to swap A and B, apply controlled rotations on C by B, and then swap A and B back, as shown in the right of Fig. 12(b). The whole procedure of controlled-controlled-U is shown in Fig. 12(c) and that of CCNOT as a special case of $U=X$ is shown in Fig. 12(d).

Time evolutions of the qubits not involved in the intended operations must be decoupled or refocused [8]. It can be done efficiently by applying decoupling π-pulse sequences based on Walsh orthogonal functions [15] or equivalently on Hadamard matrices [16]. The detailed physical implementations of controlled-controlled-U, including pulse conditions and decouplings, will be discussed elsewhere [8]. For the purpose of this paper, it is sufficient to know whether any controlled-U, controlled-controlled-U, CNOT, and CCNOT are efficiently realizable up to the total phase.

APPENDIX B: EXPLICIT ALGORITHM

We give an explicit algorithm to translate an n-qubit quantum circuit $\mathcal{C}$ consisting of CNOT gates, single-qubit gates, and controlled-U gates to a sequence of operations on a QCALA $\mathcal{S}$ consisting of a repetition of m qubits.

Variables i,O,V,B are used in the algorithm, where i is the variable of the gate number, and V is the array with $n+1$ variables. $V[0]$ stores the head number, and $V[j]$ is the qubit number on $\mathcal{S}$, which corresponds to the jth qubit on $\mathcal{C}$. O is a variable that stores a sequence of operations on $\mathcal{S}$, and B is the array storing a pair of numbers $\{B^-,B^+\}$ such that $B^+-B^-=m-1$. The xth qubit is called in the *bundle* B if $B^-\leqslant x\leqslant B^+$. The bundle B will be defined as always containing the head. $\mathcal{C}$ is input to the algorithm, and O and V are output.

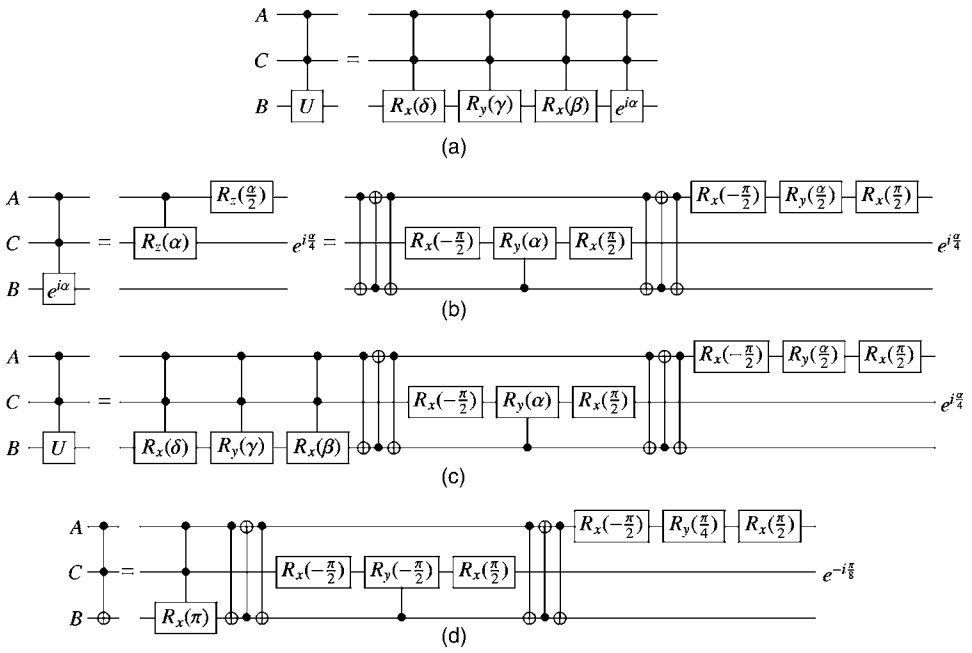

FIG. 12. Realizations of controlled-controlled gates; (a) decomposition of controlled-controlled-U, (b) realization of controlled-controlled phase shift, (c) realization of controlled-controlled-U, and (d) realization of CCNOT. The scalar phase factor indicated at the end of each quantum circuit is the total phase, which can be safely omitted in actual implementations.

1. Initialization

Our method initially assigns each input of a given quantum circuit, say $x_1, x_2, \ldots, x_n$, to each qubit on $\mathcal{S}$ one by one from the right side of the 0th qubit, but we do not use qubits that are in the same group as the 0th qubit. The 0th qubit is initialized as 1. All the others are set to 0. We thus initialize V as follows: $V[0]=0$ and $V[j]=j+\lfloor (j-1)/(m-1)\rfloor$ for $j=1,2,\ldots,n$. For example, when $m=4$ and $n=5$, the input sequence of bits are $\ldots,0,0,0,1,x_1,x_2,x_3,0,x_4,x_5,0,0,0,\ldots$. Then, $V=\{0,1,2,3,5,6\}$. i is initialized as 1. O is initialized as an empty set. B is initialized as $\{0,m-1\}$.

2. Step I

Suppose $V[c_i]$ and $V[t_i]$ are in the same group. In this case, bits on $V[t_i]$ and $V[c_i]$ move the same together by SWAP operations. So, we move one of them onto a different group by CSWAP.

As a special case, if $m=3$ and $B^- < V[0] < B^+$ (i.e., the head is in the middle of the bundle), then A_i and A_j are swapped by a SWAP operation, where A_i and A_j are the types of the qubit numbers $V[0]$ and $V[0]+1$, respectively. Otherwise, this operation is skipped.

$$O := O[A_i, A_j].$$

$$V[j] := \begin{cases} V[j]+1 & \text{if} \quad V[j] \sim V[0], \\ V[j]-1 & \text{if} \quad V[j] \sim V[0]+1, \\ \text{Unchanged otherwise}. \end{cases}$$

Define $x=1$ if the head is the leftmost or the second left in the bundle (case 2 or case 3 in Fig. 5), and $x=0$ otherwise. Define y as $V[t_i]$ if $|V[t_i]-V[0]| \leqslant |V[c_i]-V[0]|$, and otherwise as $V[c_i]$. Define z as

$$z := \begin{cases} V[0]-1 & \text{if} \quad y < V[0] \quad \text{and} \quad x=0, \\ V[0] & \text{if} \quad y < V[0] \quad \text{and} \quad x=1, \\ V[0] & \text{if} \quad V[0] < y \quad \text{and} \quad x=0, \\ V[0]+1 & \text{if} \quad V[0] < y \quad \text{and} \quad x=1. \end{cases}$$

The translation algorithm is split into two cases according to the order of y and z.

Case $z<y$. Define $z' := y - m\lceil (y-z)/m \rceil$ and $z'' := y - m\lfloor (y-z)/m \rfloor$:

$$O := O\rho(y,z),$$

$$B := \left\{ B^- + \left\lfloor \frac{y-z}{m} \right\rfloor - x, B^+ + \left\lfloor \frac{y-z}{m} \right\rfloor - x \right\},$$

$$V[j] := \begin{cases} V[j] - \left(y - z - \left\lfloor \dfrac{y-z}{m} \right\rfloor \right) & \text{if} \quad V[j] \sim y, \\ V[j] + \left\lfloor \dfrac{y-z}{m} \right\rfloor & \text{if} \quad V[j] \sim w \text{ for some } z' < w < z, \\ V[j] + \left\lceil \dfrac{y-z}{m} \right\rceil & \text{if} \quad V[j] \sim w \text{ for some } z \leqslant w < z''. \end{cases}$$

Case $y<z$. Define $z' := y + \lceil (z-y)/m \rceil$ and $z'' := y + \lfloor (z-y)/m \rfloor$:

$$O := O\rho(y,z),$$

$$B := \left\{ B^- - \left\lfloor \frac{z-y}{m} \right\rfloor - x, B^+ - \left\lfloor \frac{z-y}{m} \right\rfloor - x \right\},$$

$$V[j] := \begin{cases} V[j] + \left(z - y - \left\lfloor \frac{z-y}{m} \right\rfloor \right) & \text{if} \quad V[j] \sim y, \\ V[j] - \left\lfloor \frac{z-y}{m} \right\rfloor & \text{if} \quad V[j] \sim w \text{ for some } z < w < z', \\ V[j] - \left\lceil \frac{z-y}{m} \right\rceil & \text{if} \quad V[j] \sim w \text{ for some } z'' < w \leqslant z. \end{cases}$$

By (one of) the above operations, the bit on the yth qubit (before the operation) moves on the neighbor of the head. Let A_h be the type of qubit holding the head. Let A_y be the type of the qubit to which the yth qubit moved. Then, A_y adjoins A_h. Let A_z be another neighbor of A_y.

We further revise O and V as follows:

$$O := O(A_y, A_z, a_h),$$

$$V[j] := \begin{cases} z & \text{if} \quad V[j] = y, \\ y & \text{if} \quad V[j] = z, \\ \text{Unchanged otherwise.} \end{cases}$$

3. Step II

Suppose $V[c_i]$ is out of the bundle. (Otherwise, go to step III.) Let c be a number such that c and $V[c_i]$ are in the same group, and c is in the bundle. Define $r=(V[c_i]-c)/m$.

$$O := O\rho(V[c_i], c),$$

$$B := \{B^- + r, B^+ + r\},$$

$$V[j] := \begin{cases} V[j] - r(m-1) & \text{if} \quad V[j] \sim c, \\ V[j] + r & \text{otherwise.} \end{cases}$$

4. Step III

Suppose $V[t_i]$ is out of the bundle. (Otherwise, go to the step IV.) Let t be a number such that t and $V[t_i]$ are in the same group, and t is in the bundle. Define $r=(V[t_i]-t)/m$:

$$O := O\rho(V[t_i], t),$$

$$B := \{B^- + r, B^+ + r\},$$

$$V[j] := \begin{cases} t - r(m-1) & \text{if} \quad V[j] \sim t, \\ V[j] + r & \text{otherwise.} \end{cases}$$

5. Step IV

Suppose $V[c_i]$, $V[t_i]$, and $V[0]$ (the head) are in the bundle. We move them so that they are adjacent in order of the head $V[c_i]$ and $V[t_i]$. Then, we perform the original CNOT gates by using a CCNOT operation on $\mathcal{S}$.

Define h, c, t as $h := V[0]$, $c := V[c_i]$, $t := V[t_i]$. The translation algorithm is split into six cases according to the order of them. The bundle is not changed in each case.

Case $h < c < t$.

$$O := O\rho(h, c-1)\rho(t, c)[a_{c-1}, A_c, a_{c+1}],$$

$$V[j] := \begin{cases} V[j] + c - h - 1 & \text{if} \quad V[j] \sim h, \\ V[j] - 1 & \text{if} \quad V[j] \sim w \text{ for some } h < w < c, \\ V[j] + 1 & \text{if} \quad V[j] \sim w \text{ for some } c \leqslant w < t, \\ V[j] + c - t & \text{if} \quad V[j] \sim t, \\ \text{Unchanged otherwise.} \end{cases}$$

Case $h < t < c$.

$$O := O\rho(h, t-1)\rho(c, t+1)[a_{t-1}, A_t, a_{t+1}],$$

$$V[j] := \begin{cases} V[j] + t - h - 1 & \text{if} \quad V[j] \sim h, \\ V[j] - 1 & \text{if} \quad V[j] \sim w \text{ for some } h < w < t, \\ V[j] + 1 & \text{if} \quad V[j] \sim w \text{ for some } t < w < c, \\ V[j] + t - c + 1 & \text{if} \quad V[j] \sim c, \\ \text{Unchanged otherwise.} \end{cases}$$

Case $c < h < t$.

$$O := O\rho(c, h-1)\rho(t, h)[a_{h-1}, A_h, a_{h+1}],$$

$$V[j] := \begin{cases} V[j] + h - c - 1 & \text{if} \quad V[j] \sim c, \\ V[j] - 1 & \text{if} \quad V[j] \sim w \text{ for some } c < w < h, \\ V[j] + 1 & \text{if} \quad V[j] \sim w \text{ for some } h \leqslant w < t, \\ V[j] + h - t & \text{if} \quad V[j] \sim t, \\ \text{Unchanged otherwise.} \end{cases}$$

Case $t < h < c$.

$$O := O\rho(t, h)\rho(c, h+1)[a_{h-1}, A_h, a_{h+1}],$$

$$V[j] := \begin{cases} V[j] + h - t & \text{if} \quad V[j] \sim t, \\ V[j] - 1 & \text{if} \quad V[j] \sim w \text{ for some } t < w \leqslant h, \\ V[j] + 1 & \text{if} \quad V[j] \sim w \text{ for some } h < w < c, \\ V[j] + h - c + 1 & \text{if} \quad V[j] \sim c, \\ \text{Unchanged otherwise.} \end{cases}$$

Case $c < t < h$.

$$O := O\rho(c, t-1)\rho(h, t+1)[a_{t-1}, A_t, a_{t+1}],$$

$$V[j] := \begin{cases} V[j] + t - c - 1 & \text{if} \quad V[j] \sim c, \\ V[j] - 1 & \text{if} \quad V[j] \sim w \text{ for some } c < w < t, \\ V[j] + 1 & \text{if} \quad V[j] \sim w \text{ for some } t < w < h, \\ V[j] + t - h + 1 & \text{if} \quad V[j] \sim h, \\ \text{Unchanged otherwise.} \end{cases}$$

Case $t < c < h$.

$$O := O\rho(t, c)\rho(h, c+1)[a_{c-1}, A_c, a_{c+1}],$$

$$V[j] := \begin{cases} V[j]+c-t & \text{if} \quad V[j] \sim t, \\ V[j]-1 & \text{if} \quad V[j] \sim w \text{ for some } t < w \leqslant c, \\ V[j]+1 & \text{if} \quad V[j] \sim w \text{ for some } c < w < h, \\ V[j]+h-c+1 & \text{if} \quad V[j] \sim h, \\ \text{Unchanged otherwise.} \end{cases}$$

We need $(m-2)$ SWAPS $[3(m-2)$ CNOT operations$]$ at most to move them so that they are adjacent. Thus, we need at most $3(m-2)+1$ operations at step IV.

6. Step IV′

The difference between step IV′ and step IV is the last operation in the revised O; the operation $[a_{k-1},A_k,a_{k+1}]$ in step IV is replaced with controlled-controlled-U operation $U[a_{k-1},A_k,a_{k+1}]$. All the index numbers are the same.

7. Step IV″

Unlike step IV, we do not consider the control bit, so the procedure is modified as follows.

Case $h<t$.

$$O := O\rho(h,t-1)U[a_{t-1},A_t],$$

$$V[j] := \begin{cases} V[j]+t-h-1 & \text{if} \quad V[j] \sim h, \\ V[j]-1 & \text{if} \quad V[j] \sim w \text{ for some } h < w < t, \\ \text{Unchanged otherwise.} \end{cases}$$

Case $t<h$.

$$O := O\rho(h,t+1)U[a_{t+1},A_t]$$

$$V[j] := \begin{cases} V[j]+t-h+1 & \text{if} \quad V[j] \sim h, \\ V[j]+1 & \text{if} \quad V[j] \sim w \text{ for some } t < w < h, \\ \text{Unchanged otherwise.} \end{cases}$$

8. Output and flow chart

Figure 13 is the flow chart of the translation algorithm.

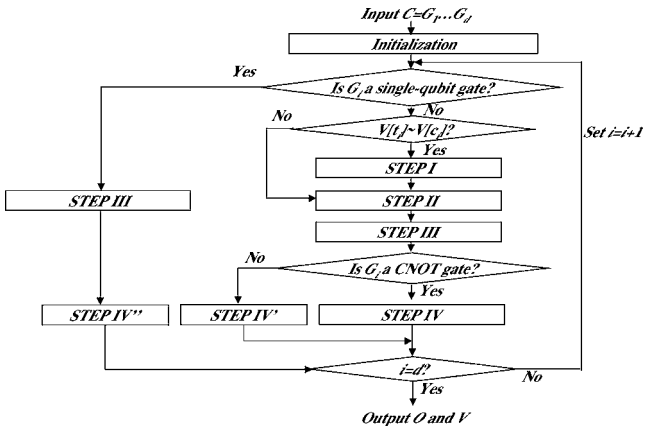

FIG. 13. This is the flow chart of the algorithm translating an n-qubit quantum circuit $\mathcal{C}$ consisting of d gates (CNOT gates, single-qubit gates, or controlled-U gates) to a sequence O of operations on a QCALA $\mathcal{S}$. $\mathcal{C}$ is input and O is output. The array V of $n+1$ variables is also output. $\mathcal{C}$ can be written as $G_1,G_2,\ldots,G_d$, a sequence of CNOT gates. The translation is performed for each gate from G_1 to G_d. O is initialized as empty. New operations are added to the end of O at each step.

We explain here how to obtain output values of a quantum circuit $\mathcal{C}$ by using the translation algorithm.

We suppose that a quantum circuit $\mathcal{C}$ is input into the algorithm, and the algorithm outputs O and V. Let x be a sequence of bits $x_1,\ldots,x_n$. We denote the output sequence of $\mathcal{C}$ as $\mathcal{C}(x)_1,\ldots,\mathcal{C}(x)_n$. The x is transformed to the input sequence of O according to the method described in Appendix B 1. Let x' be the transformed input sequence of x on $\mathcal{S}$. When x' is input to O, the output sequence is denoted by $\ldots,O(x')_j,O(x')_{j+1},\ldots$, where j's are qubit numbers. Then, it is inductively proved that

$$\mathcal{C}(x)_k = O(x')_{V[k]}$$

for all k $(1\leqslant k\leqslant n)$. Thus, we are able to know the kth output qubit value of $\mathcal{C}$ by observing the $V[k]$-th output qubit value of O.

[1] S. Lloyd, Science **261**, 1569 (1993).
[2] S. C. Benjamin and N. F. Johnson, Phys. Rev. A **60**, 4334 (1999).
[3] S. C. Benjamin, Phys. Rev. A **61**, 020301(R) (2000).
[4] S. C. Benjamin, Phys. Rev. Lett. **88**, 017904 (2002).
[5] S. C. Benjamin and S. Bose, Phys. Rev. Lett. **90**, 247901 (2003).
[6] S. C. Benjamin and S. Bose, Phys. Rev. A **70**, 032314 (2004).
[7] S. Lloyd, quant-ph/9912086.
[8] K. Nagashima and M. Kitagawa (unpublished).
[9] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).
[10] N. A. Gershenfeld and I. L. Chuang, Science **275**, 350 (1997).
[11] D. G. Cory, M. D. Price, and T. F. Havel, Physica D **120**, 82 (1998).
[12] P. W. Shor, SIAM J. Comput. **26**, 1484 (1997).
[13] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, Nature (London) **414**, 883 (2001).
[14] D. G. Cory, A. E. Dunlop, T. F. Havel, S. S. Somaroo, and W. Zhang, quant-ph/9809045.
[15] M. Kitagawa, *Mathematical Sciences* (SAIENSU-SHA, Tokyo, 1998), No. 424, pp. 43–50.
[16] D. W. Leung, I. L. Chuang, F. Yamaguchi, and Y. Yamamoto, Phys. Rev. A **61**, 042310 (2000).