A Constructive Algorithm for Reversible Logic Synthesis

Guowu Yang[1], Fei Xie[1], Xiaoyu Song[2], William N. N. Hung[2], and Marek A. Perkowski[2]

***Abstract*— This paper presents a constructive synthesis algorithm for any n-qubit reversible function. Given any n-qubit reversible function, there are N distinct input patterns different from their corresponding outputs, where $N \leq 2^n$, and the other $(2^n - N)$ input patterns will be the same as their outputs. We show that this circuit can be synthesized by at most $2n \cdot N$ '$(n-1)$'-CNOT gates and $4n^2 \cdot N$ NOT gates. The time complexity of our algorithm has asymptotic upper bound $O(n \cdot 4^n)$. The space complexity of our synthesis algorithm is also $O(n \cdot 2^n)$. The computational complexity of our synthesis algorithm is exponentially lower than the complexity of breadth-first search based synthesis algorithm.**

I. Introduction

Reversible logic plays an important role in quantum computing [1], [2]. It has been shown that any computing system of irreversible logic gates leads inevitably to energy dissipation [3]–[5]. To avoid power dissipation, circuits must be constructed [4], [5] from reversible gates. Readers interested in various physical and technological aspects of realizing quantum circuits from reversible gates are referred to [1], [6]–[10]. There are a lot of research [2], [5], [9], [11]–[22] on the construction of reversible logic gates and circuits.

A fundamental question on reversible logic is what kind of reversible circuits can be implemented, given a library of reversible logic gates. In this paper, we show that any reversible logic function with n $(n > 2)$ qubits can be constructed by NOT and '$(n-1)$'-CNOT gates. We present a novel, concise and constructive proof based on group theory. Our quasi-minimal synthesis algorithm is derived based on the constructive proof, where the numbers of '$(n-1)$'-CNOT (definition 5) and NOT gates required in the realization are bounded by $2n \cdot N$ and $4n^2 \cdot N$, respectively, where N is the number of distinct input patterns that are different from their corresponding output patterns. Our provable synthesis algorithm outperforms search-based approaches. The time complexity of our quasi-minimal algorithm has asymptotic upper bound $O(n \cdot 4^n)$. In contrast, a search based exact synthesis algorithm may have a worst case time complexity of $(2^n)!$.

The rest of the paper is organized as follows. In section II, we present definitions of reversibility, permutation, and some elementary reversible logic gates. Then, in section III, we proceed to prove a few lemmas and subsequently prove that every reversible function can be synthesized within our upper bounded number of gates. To showcase the practicality of our proof, we rephrase the proof as an synthesis algorithm in section IV and present some synthesis examples. We analyze the complexity of our algorithm in section V. Our conclusion is given in section VI.

[1]Dept. of Computer Science, Portland State University, Portland, OR 97207, USA. Contact author's e-mail: guowu@ece.pdx.edu

[2]Dept. of Electrical and Computer Engineering, Portland State University, Portland, OR 97207, USA.

II. Preliminaries

In this section, we introduce some basic concepts and results on permutation group theory from [23] and binary reversible logic from [24]–[27].

Definition 1 (Binary reversible gate): Let $B = \{0, 1\}$. A binary logic circuit f with n inputs and outputs is denoted by a binary multiple-output function $f : B^n \to B^n$. Let $\langle B_1, \cdots, B_n \rangle \in B^n$ and $\langle P_1, \cdots, P_n \rangle \in B^n$ be the input and output vectors, where $B_1, \cdots, B_n$ are input variables and $P_1, \cdots, P_n$ are output variables. There are 2^n different assignments for the input vectors. A binary logic circuit f is $reversible$ if it is a one-to-one and onto function (bijection). A binary reversible logic circuit with n inputs and n outputs is also called an n-qubit binary reversible gate. There are a total of $(2^n)!$ different n-qubit binary reversible circuits.

We introduce permutation groups and their relationship with reversible circuits.

Definition 2 (Permutation): Let $M = \{d_1, d_2, \cdots, d_k\}$. A bijection of M onto itself is called a permutation on M. The set of all permutations on M forms a group under composition of mappings, called a symmetric group on M. It is denoted by S_k [23]. A permutation group is simply a subgroup [23] of a symmetric group.

A mapping $s : M \to M$ can be written as:

$$s = \begin{pmatrix} d_1, d_2, \cdots, d_k \\ d_{i_1}, d_{i_2}, \cdots, d_{i_1} \end{pmatrix} \tag{1}$$

Here we use a product of disjoint cycles (Definition 3) as an alternative notation for a mapping [23]. For example,

$$\begin{pmatrix} d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9 \\ d_1, d_4, d_7, d_2, d_5, d_8, d_3, d_6, d_9 \end{pmatrix} \tag{2}$$

can be written as $(d_2, d_4)(d_3, d_7)(d_6, d_8)$. Denote "( )" as the identity mapping (i.e., direct wiring) and call this the unity element in a permutation group. The inverse mapping of mapping f is denoted as f^{-1}. Per convention, a product $f * g$ of two permutations applies mapping f before g.

A n-qubit reversible circuit is a permutation in S_{2^n}, and vice versa. Cascading two gates is equivalent to multiplying two permutations in S_{2^n}. Thus, in what follows, we will not distinguish a n-qubit reversible circuit from a permutation in S_{2^n}.

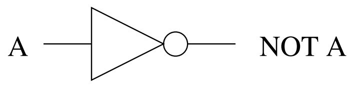

Fig. 1. NOT gate

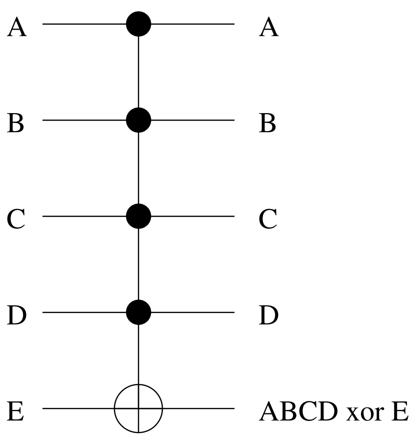

Fig. 2. '4'-CNOT gate

Definition 3 ('j'-cycle): Let S_k be a symmetric group of symbols $\{d_1, d_2, \cdots, d_k\}$, then $(d_{i_1}, d_{i_2}, \cdots, d_{i_j})$ is called a 'j'-cycle, where $j \leq k$, $1 \leq i_1, i_2, \cdots, i_j \leq k$.

Definition 4 (NOT gate): A NOT gate N_j connects an inverter to the j-th wire, i.e.: $P_j = B_j \oplus 1; \quad P_i = B_i, \quad if \quad i \neq j. \quad 1 \leq j \leq n.$

An example NOT gate is shown in figure 1.

Definition 5 ('$n-1$'-CNOT gate): A '$n-1$'-Controlled-NOT (CNOT) gate C_j is defined as follows:

- If $m \neq j$, then $P_m = C_j(B_m) = B_m$.
- If $m = j$, and $B_i = 1$ for all $i \neq j$, then $P_j = C_j(B_j) = B_j \oplus 1$, else, $P_j = B_j$.

An example '4'-CNOT gate is shown in figure 2.

A '$n-1$'-CNOT gate is a generalized Toffoli gate where two inputs control an output of another input.

III. Theoretical results

In this section, we show the process how to constructively synthesize any 'n'-qubit reversible circuit by Not and '$n-1$'-CNOT gates without ancilla qubits. It will be used in our synthesis algorithm in section IV.

Lemma 1: All permutations can be generated by some '2'-cycles.

Proof: Any permutation can be written as product of some disjoint cycles. So we only need to show that every cycle $(d_1, d_2, \ldots, d_k)$ can be expressed as a product of some 2-cycles.

$$(d_1, d_2, \ldots, d_k) = (d_1, d_2)(d_1, d_3, \ldots, d_k) \tag{3}$$

Recursively using this equation, lemma 1 holds.

■

Remark 1: This lemma is a well-known result in group theory, we give a proof for showing the equation 3 which will be used in our algorithm.

TABLE I
The ordered set M of u and s

P_1	P_2	P_3	P_4	P_5	encode
0	0	1	1	1	$d_1 = s$
0	1	1	1	1	d_2
0	1	0	1	1	d_3
0	1	0	0	1	d_4
0	1	0	0	0	$d_5 = u$

Definition 6 (neighboring '2'-cycle): If two n-dimension vectors u, s have only one bit different, we call the permutation (u, s) a neighboring '2'-cycle.

Lemma 2: Suppose between u and s, there is only one bit B_j different, and l same bits are zeros. These zero bits are $B_{i_1}, \ldots, B_{i_l}$. Then,

$$(u, s) = N_{i_1} * \ldots * N_{i_l} * C_j * N_{i_l} * \ldots * N_{i_1} \tag{4}$$

Proof: This lemma holds by direct computation.

■

Lemma 3: If two n-dimension vectors u, s have k bits different, then there is an ordered set $M = \{d_1, d_2, \cdots, d_{k+1}\}$ such that $d_1 = u, d_{k+1} = s$ and for any i, $1 \leq i < k+1$, there is only one bit different between d_i and d_{i+1}, and

$$(u, s) = (d_1, d_2)(d_2, d_3) \ldots (d_k, d_{k+1})(d_k, d_{k-1}) \ldots (d_2, d_1) \tag{5}$$

Proof: First, set $d_1 = u, d_{k+1} = s$. Then, each time we change one bit in the k different bits between u and s. So we get an ordered set $M = \{d_1, d_2, \cdots, d_{k+1}\}$ such that $d_1 = u, d_{k+1} = s$ and for any i, $1 \leq i < k+1$, there is only one bit difference between d_i and d_{i+1}. Equation 5 is a directly computable result.

■

Remark 2: In order to make the number of NOT gates as small as possible, we give two rules for constructing the ordered set M.

- If the number of 1s in the vector u is more than that in s, then $d_1 = u, d_{k+1} = s$. Else, $d_1 = s, d_{k+1} = u$.
- In the different bits between u and s, change the zero bit to one first, then change one bit to zero bit.

For example, if $u = \langle 0, 1, 0, 0, 0 \rangle$, $s = \langle 0, 0, 1, 1, 1 \rangle$, then $k = 4, d_1 = s, d_5 = u$ and, d_2, d_3, d_4 are given in Table I.

Remark 3: There is commutativity in the product of NOT gates, and we can remove a pair of the adjacent NOT gates in the same quantum wire.

Lemma 4: In decomposing '2'-cycle (u, s) to NOT gates and '$n-1$'-CNOT gates, suppose there are j bits different between u and s, and there are j_0 zero bits in these bits in d_1 and j_1 one bits in these bits in d_1, where $j_0 \leq j_1$, then the number of NOT gates is no more than $2(j-2) + 2(j_0 - 1)$ if $j_0 \geq 1$; or $2(j-1)$ if $j_0 = 0$.

Proof: Using two rules in remark 2 and the property of NOT gate in remark 3, we can calculate the number of the needed NOT gates, no more than $2(j-2) + 2(j_0 - 1)$ if $j_0 \geq 1$; or $2(j-1)$ if $j_0 = 0$.

■

Theorem 1: For a given n-qubit reversible circuit f, if there are N distinct input patterns that are different from their corresponding outputs, where $N \leq 2^n$, and the other $(2^n - N)$ input patterns are the same as their outputs, then this circuit can be synthesized by at most $2n \cdot N$ '$(n-1)$'-CNOT gates and $4n^2 \cdot N$ NOT gates without ancilla qubit.

Proof: According to equation 3, this reversible circuit can be decomposed to at most $N-1$ '2'-cycles. According to lemma 3 and 2, the number of '$(n-1)$'-CNOT gates is no more than $2n \cdot N$; the number of NOT gates is no more than $2n \cdot 2n \cdot N = 4n^2 \cdot N$.

■

Theorem 2: All n-qubit reversible circuits can be constructed by less than $2n \cdot 2^n$ NOT gates and less than $(2n-1) \cdot 2^n$ '$n-1$'-CNOT gates without ancilla qubit.

Proof: Let $(d_1, d_2), (d_3, d_4), \ldots, (d_{m-1}, d_m)$, where $m = 2^n$, be the all '2'-cycles where d_{2i-1} and d_{2i} have the maximal number of different bits, n.

When we optimally decompose any permutation p in S_m to a product of some neighboring '2'-cycles, let function $N(p)$ be the minimal number of neighboring '2'-cycles.

When $p = (d_1, d_2)(d_3, d_4) \ldots (d_{m-1}, d_m)$, $N(p)$ achieves the maximal number $(2n-1) \cdot 2^n$. The reason is the following. When use equation 1 to optimally decompose p to some neighboring 2-cycles based equation 5, if (d_{2i-1}, d_{2i}) is in the decomposition, then (d_{2i-1}, a) or (d_{2i}, b) will not be included in this decomposition. By contradiction, assume (d_{2i-1}, a) is in the optimal decomposition.

$$\begin{aligned}(d_{2i-1}, d_{2i})(d_{2i-1}, a) &= (d_{2i-1}, d_{2i}, a) \\ &= (d_{2i-1}, a)(d_{2i}, a)\end{aligned}$$

The number r of different bits between d_{2i} and a is less than n. According to equation 5

$$N((d_{2i}, a)) = 2r - 1 < 2n - 1 = N((d_{2i-1}, d_{2i})).$$

Thus $(d_{2i-1}, a)(d_{2i}, a)$ is a better decomposition than $(d_{2i-1}, d_{2i})(d_{2i-1}, a)$.

Therefore, (d_{2i-1}, a) or (d_{2i}, b) will not be included in this decomposition. So, the product p of all these '2'-cycles with maximal n different bits makes $N(p)$ to be $(2n-1) \cdot 2^n$.

Using equation 2, the number of '$n-1$'-CNOT gates is no more than $(2n-1) \cdot 2^n$.

Let the number of NOT gates be Y, $C(i; j)$ be a binomial coefficient [28]. Using lemma 4 and properties of binomial coefficient, when $n = 2k-1$, $k \geq 2$, an odd number,

$$\begin{aligned}Y &= 2(2k-3)C(2k-1; 0) \\ &\quad +(2(2k-3) + 2(1-1))C(2k-1; 1) \\ &\quad + \ldots + (2(2k-3) + 2(k-2)C(2k-1; k-1)) \\ &= (3k-4)2^{2k-1} - (2k-1)C(2k-2; k-1) + 2 \\ &< 2n \cdot 2^n\end{aligned}$$

when $n = 2k, k \geq 2$, an even number,

$$\begin{aligned}Y &= 2(2k-2)C(2k; 0) \\ &\quad +(2(2k-2) + 2(1-1))C(2k; 1) \\ &\quad + \ldots + (2(2k-2) + 2(k-1)C(2k; k)) \\ &= (3k-4)2^{2k} + (2k-3)C(2k; k) \\ &< 2n \cdot 2^n\end{aligned}$$

■

Remark 4: The upper bound for NOT gates can be reduced by further removing the pair of the adjacent NOT gates in the same quantum wire. This is illustrated by the example in the next section.

Remark 5: The product of all '2'-cycles with maximal n different bits indeed is the product of all n different NOT gates. Thus the approach by directly using equation 3 and lemma 3 has some defects. We should consider the NOT gate before using equation 3.

The idea of considering the NOT gate before using equation 3 is:

Consider the truth table of a given reversible circuit, each output bit has 2^n values, compare them with the input values, if the number of different values is bigger than 2^{n-1}, tie a NOT gate to this bit. After dealing with all bits, count the changed vectors, if the number of the changed vectors is less than that of the original circuit, then decompose the reversible circuit dealt by NOT gates using equation 3 and lemma 3. Otherwise, decompose the original reversible circuit. The decomposition algorithm and examples are given in the next section.

IV. Algorithm and Synthesis Example

Based on the above analysis, we present the following constructive algorithm for synthesizing any given binary reversible circuit f without using ancilla qubits.

Algorithm:

Step 1. Check the truth table of f to determine before using equation 3 and lemma 3, whether we need NOT gates or not.

Step 2. After step 1, write the reversible circuit in a product of cycles form. For every cycle $(d_1, d_2, \ldots, d_k)$, calculate the number r_i of different bits between d_i and d_{i+1}, $i = 1, 2, \ldots, k$ where $d_{k+1} = d_1$. Let r_j be the maximal number. The basic idea to decompose the reversible circuit by equation 3 is to break the mapping relation from d_j to d_{j+1} without increasing the number of different bits between adjacent vectors.

$$(d_1, d_2, \ldots, d_k) = (d_j, d_{j+1})(d_j, d_{j+2}, d_{j+3}, \ldots, d_1) \quad (6)$$

Recursively repeating this process, we decompose the reversible circuit to '2'-cycles.

Step 3. Decompose every '2'-cycle by NOT and '$n-1$'-CNOT gates by Lemma 3, two rules in remark 2, lemma 2, and remove pairs of adjacent NOT gates when possible.

Example 1: Given a binary reversible circuit f which has a truth table as shown in Table II.

TABLE II

A BINARY REVERSIBLE CIRCUIT f

input					output				
B_1	B_2	B_3	B_4	encoding	P_1	P_2	P_3	P_4	encoding
0	0	0	0	a_1	0	1	0	0	a_3
1	0	0	0	a_2	1	0	1	0	a_6
0	1	0	0	a_3	1	1	0	0	a_4
1	1	0	0	a_4	1	1	1	1	a_{16}
0	0	1	0	a_5	0	0	1	0	a_5
1	0	1	0	a_6	1	0	1	1	a_{14}
0	1	1	0	a_7	0	1	1	0	a_7
1	1	1	0	a_8	1	1	1	0	a_8
0	0	0	1	a_9	0	0	0	1	a_9
1	0	0	1	a_{10}	1	0	0	1	a_{10}
0	1	0	1	a_{11}	0	1	0	1	a_{11}
1	1	0	1	a_{12}	1	1	0	1	a_{12}
0	0	1	1	a_{13}	0	0	1	1	a_{13}
1	0	1	1	a_{14}	1	0	0	0	a_2
0	1	1	1	a_{15}	0	1	1	1	a_{15}
1	1	1	1	a_{16}	0	0	0	0	a_1

Therefore, $f = (a_1, a_3, a_4, a_{16})(a_2, a_6, a_{14})$.

Step 1. The total number of changed vectors is 7, less than $2^{4-1} = 8$, thus, we deal with the input reversible circuit f without pre-cascading NOT gates.

Step 2. Decompose each cycle into the product of 2-cycles by using equation 6.

$$\begin{aligned}(a_1, a_3, a_4, a_{16}) &= (a_4, a_{16})(a_4, a_1, a_3)\\ &= (a_4, a_{16})(a_4, a_3)(a_1, a_3)\\ (a_2, a_6, a_{14}) &= (a_6, a_{14})(a_6, a_2)\end{aligned}$$

Step 3. By applying equation 6 and lemma 2, we have:

$$\begin{aligned}(a_4, a_{16}) &= (a_{16}, a_{12})(a_{12}, a_4)(a_{16}, a_{12})\\ &= C_3 * N_3 * C_4 * N_3 * C_3\\ (a_4, a_3) &= N_3 * N_4 * C_1 * N_3 * N_4\\ (a_1, a_3) &= N_1 * N_3 * N_4 * C_2 * N_1 * N_3 * N_4\\ (a_6, a_{14}) &= N_2 * C_4 * N_2\\ (a_6, a_2) &= N_2 * N_4 * C_3 * N_4 * N_2\end{aligned}$$

Therefore,

$$\begin{aligned}f = \;& C_3 * N_3 * C_4 * N_3 * C_3 * N_3 * N_4\\ & *C_1 * N_1 * C_2 * N_1 * N_3 * N_4\\ & *N_2 * C_4 * N_4 * C_3 * N_4 * N_2.\end{aligned}$$

The synthesis process is finished, and f is decomposed into the product of 12 NOT gates and 7 '$n-1$'-CNOT gates, shown in figure 3.

Example 2: Given a binary reversible circuit g which has a truth table as shown in Table III.

Step 1. Only the output P_1 has over $2^{4-1} = 8$ different values with input B_1. So we need cascade a NOT gate N_1 after g, shown in figure 4.

The remaining steps. $g * N_1 = f$, so the rest of the steps is the same as Example 1, and $g = f * N_1$.

Remark 6: From these two examples, especially the second example, the numbers of NOT gates and '$n-1$'-CNOT gates are much less than the upper bound that we gave in Theorem 1. The optimal upper bound of our algorithm is still our future research.

V. COMPLEXITY ANALYSIS

In this section, we analyze the computation complexity of our algorithm. Compared with exact breadth-first search based synthesis algorithm, the computation complexity of our algorithm is exponentially lower.

Theorem 3: The time complexity of our synthesis algorithm is $O(n \cdot 4^n)$.

Proof: The time complexity of step 1 is $n \cdot 2^n$, since we need to check all values in truth table. In step 2, to get equation 6, in worst case ($k = 2^n$), we need $n \cdot 2^n$ computations. And we need recursively use equation 6 $k-1$ times, so the time complexity of step 2 is $n \cdot (2^n)^2/2 = n \cdot 4^n/2$. In step 3, there are $2^n - 1$ '2'-cycles in worst case. According lemma 3 and 2, decompose every '2'-cycle to NOT and '$n-1$'-CNOT gates, we need $2n \cdot 2n = 4n^2$. Removing NOT gates needs to check all these $2n \cdot 2^n$ NOT gates. So the time complexity of step 3 is $4n^2 \cdot 2^n + 2n \cdot n \cdot 2^n = 6n^2 \cdot 2^n$.

Therefore, the total time complexity of the synthesis algorithm is:

$$n \cdot 2^n + n \cdot 4^n/2 + 6n^2 \cdot 2^n = O(n \cdot 4^n).$$

■

Remark 7: Our method is a constructive algorithm, since for each step, we are simply transforming the formula to obtain the synthesized gates. We do not need to search other reversible circuits that do not appear in our result. The computational complexity of our synthesis algorithm

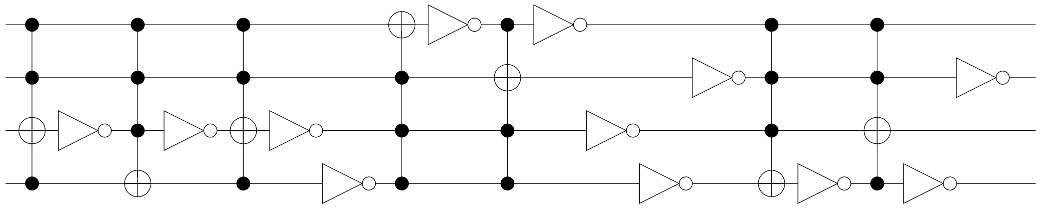

Fig. 3. Decomposed circuit for f

TABLE III

A BINARY REVERSIBLE CIRCUIT g

input					output				
B_1	B_2	B_3	B_4	encoding	P_1	P_2	P_3	P_4	encoding
0	0	0	0	a_1	1	1	0	0	a_4
1	0	0	0	a_2	0	0	1	0	a_5
0	1	0	0	a_3	0	1	0	0	a_3
1	1	0	0	a_4	0	1	1	1	a_{15}
0	0	1	0	a_5	1	0	1	0	a_6
1	0	1	0	a_6	0	0	1	1	a_{13}
0	1	1	0	a_7	1	1	1	0	a_8
1	1	1	0	a_8	0	1	1	0	a_7
0	0	0	1	a_9	1	0	0	1	a_{10}
1	0	0	1	a_{10}	0	0	0	1	a_9
0	1	0	1	a_{11}	1	1	0	1	a_{12}
1	1	0	1	a_{12}	0	1	0	1	a_{11}
0	0	1	1	a_{13}	1	0	1	1	a_{14}
1	0	1	1	a_{14}	0	0	0	0	a_1
0	1	1	1	a_{15}	1	1	1	1	a_{16}
1	1	1	1	a_{16}	1	0	0	0	a_2

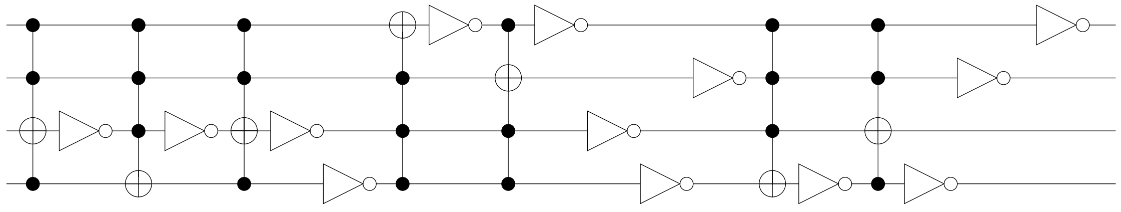

Fig. 4. Decomposed circuit for g

is exponentially lower than the complexity of breadth-first search based synthesis algorithm, which needs to explore a number of different reversible gates in each step and only a subset of them are used in the result. The space complexity of any breadth-first search based synthesis algorithm for n qubits reversible circuit is more than $(2^n)!$, since in the worst case, it at least needs to remember all $(2^n)!$ reversible circuits. This is impossible even when $n = 4$ since $(2^4)! \approx 2.0 \times 10^{13}$. The time complexity is also greater than $(2^n)!$, because in the worst case, it at least needs to compute all reversible circuits. In fact, it also has to do a lot of comparisons of equality to determine whether the calculated circuit is the given circuit or not, so the time complexity of any breadth-first search based synthesis algorithm is much more than $(2^n)!$.

Theorem 4: The space complexity of our synthesis algorithm is $6n \cdot 4^n$.

Proof: The space complexity of step 1 is $2n \cdot 2^n$, since we need to store the input assignments and output assignments in truth table for computing the different number of values between the input and the output. After we finish step 1, we do not have to store the input assignment. In step 2, we need to store all '2'-cycles, and we need n^2 space units to compute r_j which can be ignored by comparing with the exponential number of the needed space. So, the space complexity of step 2 is $n \cdot 2^n$. In step 3, we need to store all Not gates and '$n-1$'-CNOT gates. According to Theorem 2, the space complexity of step 3 is $4n \cdot 2^n$. Thus, the space complexity of our synthesis algorithm is $6n \cdot 2^n$. ∎

In the worst case, the breadth-first search based synthesis algorithm at least needs to store all $(2^n)!$ reversible circuits.

So, the space complexity of our synthesis algorithm is still exponentially lower than that of the breadth-first search based synthesis algorithm.

VI. Conclusions

In this paper, we presented a constructive algorithm for synthesizing reversible circuits by NOT and '$n-1$'-CNOT gates and give two synthesis examples based on this algorithm, which show that even by hand, synthesizing any '4'-qubit reversible circuit is not difficult. The computational complexity of our synthesis algorithm is exponentially lower than the complexity of breadth-first search based synthesis algorithm.

References

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, December 2000.

[2] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation rules for designing CNOT-based quantum circuits," in *Proc. DAC*, 2002, p. 28.4.

[3] R. Landauer, "Irreversibility and heat generation in the computational process," *IBM Journal of Research and Development*, vol. 5, pp. 183–191, 1961.

[4] C. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, pp. 525–532, 1973.

[5] E. Fredkin and T. Toffoli, "Conservative logic," *Int. Journal of Theoretical Physics*, vol. 21, pp. 219–253, 1982.

[6] D. Copsey, M. Oskin, T. Metodiev, F. T. Chong, and I. L. Chuang, "The effect of communication costs in solid-state quantum architectures," in *Symposium on Parallel Architectures and Applications (SPAA)*, 2003.

[7] L. M. K. Vandersypen and I. L. Chuang, "NMR techniques for quantum control and computation," *Reviews of Modern Physics*, vol. 76, no. 4, pp. 1037–1069, October 2004.

[8] L. Xiao and J. A. Jones, "Robust Logic Gates and Realistic Quantum Computation," 2005. [Online]. Available: http://arxiv.org/abs/quant-ph/0511100

[9] D. Deutsch, "Quantum computational networks," *Royal Society of London Series A*, vol. 425, pp. 73–90, 1989.

[10] D. Maslov and D. M. Miller, "Comparison of the Cost Metrics for Reversible and Quantum Logic Synthesis," 2005. [Online]. Available: http://arxiv.org/abs/quant-ph/0511008

[11] G. W. Dueck and D. Maslov, "Reversible function synthesis with minimum garbage outputs," in *Proc. 6th Int. Symp. Representations and Methodology of Future Computing Technologies (RM2003)*, 2003.

[12] P. Kerntopf, "Maximally efficient binary and multi-valued reversible gates," in *Proc. ULSI Workshop*, May 2001, pp. 55–58.

[13] ——, "Synthesis of multipurpose reversible logic gates," in *Proc. EUROMICRO Symp. Digital Systems Design*, 2002, pp. 259–266.

[14] A. Khlopotine, M. Perkowski, and P. Kerntopf, "Reversible logic synthesis by gate composition," in *Proc. IEEE/ACM Int. Workshop on Logic Synthesis*, 2002, pp. 261–266.

[15] D. Maslov and G. W. Dueck, "Garbage in reversible designs of multiple-output functions," in *Proc. 6th Int. Symp. Representations and Methodology of Future Computing Technologies (RM2003)*, 2003.

[16] A. Mishchenko and M. Perkowski, "Logic synthesis of reversible wave cascades," in *Proc. IEEE/ACM International Workshop on Logic and Synthesis*, June 2002, pp. 197–202.

[17] P. Picton, "A universal architecture for multiple-valued reversible logic," *Mutiple Valued Logic: an Int. Journal*, vol. 5, pp. 27–37, 2000.

[18] T. Toffoli, "Bicontinuous extensions of invertible combinatorial functions," *Mathematical Systems Theory*, vol. 14, pp. 13–23, 1981.

[19] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Quantum Logic Synthesis by Symbolic Reachability Analysis," in *Design Automation Conference (DAC)*, 2004, pp. 838–841.

[20] G. Yang, W. N. N. Hung, X. Song, and M. Perkowski, "Majority-Based Reversible Logic Gates," *Theoretical Computer Science*, vol. 334, no. 1–3, pp. 295–274, April 2005. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2004.12.026

[21] X. Song *et al.*, "Algebraic Characteristics of Reversible Gates," *Theory of Computing Systems*, vol. 39, no. 2, pp. 311–319, May–April 2005.

[22] G. Yang, X. Song, W. N. N. Hung, and M. Perkowski, "Fast Synthesis of Exact Minimal Reversible Circuits using Group Theory," in *ACM/IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2005, pp. 1002–1005.

[23] J. D. Dixon and B. Mortimer, *Permutation Groups*. New York: Springer, 1996.

[24] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic gates," *IEEE Trans. CAD*, vol. 22, no. 6, pp. 710–722, June 2003.

[25] A. De Vos, "Reversible computing," *Qutantum Electronics*, vol. 23, pp. 1–49, 1999.

[26] L. Storme, A. De Vos, and G. Jacobs, "Group theoretical aspects of reversible logic gates," *Journal of Universal Computer Science*, vol. 5, pp. 307–321, 1999.

[27] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proc. DAC*, 2003, pp. 318–323.

[28] K. P. Bogart, *Introductory Combinatorics*. Harcourt Brace Jovanovich, 1990.