Group Theory Based Synthesis of Binary Reversible Circuits

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

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

Abstract. This paper presents an important result addressing a fundamental question in synthesizing binary reversible logic circuits for quantum computation. We show that any even-reversible-circuit of n ($n > 3$) qubits can be realized using NOT gate and Toffoli gate ('2'-Controlled-Not gate), where the numbers of Toffoli and NOT gates required in the realization are bounded by $(n+\lfloor\frac{n}{3}\rfloor)(3\times 2^{2n-3}-2^{n+2})$ and $4n(n+\lfloor\frac{n}{3}\rfloor)2^n$, respectively. A provable constructive synthesis algorithm is derived. The time complexity of the algorithm is $\frac{10}{3}n^2 \cdot 2^n$. Our algorithm is exponentially lower than breadth-first search based synthesis algorithms with respect to space and time complexities.

1 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, 4, 5] from reversible gates. There have been extensive works [2, 6, 7, 8, 9, 10] on constructing reversible logic gates.

A fundamental question on reversible logic is what kind of reversible functions can be implemented, given a library of reversible logic gates. In this paper, we show that any even permutation with n ($n > 3$) qubits can be constructed by NOT and Toffoli gates. We present a novel, concise and constructive proof based on group theory. Our proof does not require the use of '1'-CNOT gates to synthesize n ($n > 3$) qubit functions. A synthesis algorithm is derived based on the constructive proof, where the numbers of Toffoli and NOT gates required in the realization are bounded by $(n + \lfloor\frac{n}{3}\rfloor)(3 \times 2^{2n-3} - 2^{n+2})$ and $4n(n + \lfloor\frac{n}{3}\rfloor)2^n$, respectively. The provable synthesis algorithm outperforms search-based approaches. The time complexity of our algorithm is $\frac{10}{3}n^2 \cdot 2^n$. In contrast, a search based synthesis algorithm may have a worst case time complexity of over $(2^n)!/2$.

The rest of the paper is organized as follows. In Section 2, we present the definitions of reversibility, even and odd permutations, and some elementary reversible logic gates. In Section 3, we prove that every even permutation can be synthesized using the bounded number of gates. Based on this proof, we present a synthesis algorithm with an example of synthesizing a function into 8 NOT gates and 48 Toffoli gates in Section 4. We analyze the time complexity of our algorithm in Section 5 and conclude in Section 6.

2 Preliminaries

In this section, we introduce some basic concepts and results on permutation group theory from [11] and binary reversible logic from [12, 13, 14].

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 a permutation group and its relationship with reversible circuits.

Definition 2 (Permutation). *Let $M = \{d_1, d_2, \cdots, d_k\}$. A bijection [1] 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 [11]. A permutation group is simply a subgroup [11] of a symmetric group.*

A mapping $s : M \to M$ can be written as a product of disjoint cycles (Definition 3) as an alternative notation for a mapping [11]. For example,

$$\begin{pmatrix} d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9 \\ d_1, d_4, d_7, d_2, d_5, d_8, d_3, d_6, d_9 \end{pmatrix} \tag{1}$$

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}.

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 $j \le k$, $1 \le i_1, i_2, \cdots, i_j \le k$.*

Definition 4 (even and odd permutations). *A permutation is even if it is a product of an even number of 2-cycles; and odd if it is a product of an odd number of 2-cycles.*

[1] Bijection: one-to-one, and onto mapping.

Obviously, a '3'-cycle is an even permutation. For instance, (1, 3, 2) = (2, 3) (3, 1). The product of some even permutations is also an even permutation. The product of an odd number of odd permutations is an odd permutation. The product of an even number of even permutations with an odd number of odd permutations is an odd permutation. The product of an even number of odd permutations is an even permutation.

Lemma 1. *Let S_k be a symmetric group of symbols $\{d_1, d_2, \cdots, d_k\}$. Then any even permutation in S_{2^n} can be expressed as a product of at most 2^{n-1} '3'-cycles.*

Proof. 1. Consider a 'm'-cycle: $(a_1, a_2, \cdots, a_m)$, $m \geq 4$. It is a product of a '3'-cycle and a '$(m-2)$'-cycle.

$$(a_1, a_2, \cdots, a_m) = (a_1, a_2, a_3) * (a_1, a_4, \cdots, a_m). \qquad (2)$$

If m is an odd number, $(a_1, a_2, \cdots, a_m)$ is a product of $\frac{(m-1)}{2}$ '3'-cycles. If m is an even number, $(a_1, a_2, \cdots, a_m)$ is a product of $\frac{m}{2}$ '3'-cycles and one '2'-cycle.

2. 'm'-cycle is called even (or odd, respectively) cycle if m is even (or odd). An even cycle is an odd permutation; and an odd cycle is an even permutation. Even cycle must appear as a pair of even permutation, and

$$(a, b) * (c, d) = (a, b) * (b, c) * (b, c) * (c, d) = (a, c, b) * (b, d, c), \qquad (3)$$

which means that a product of a pair of '2'-cycles is equal to a product of a pair of '3'-cycles.

Therefore, any even permutation in S_{2^n} can be expressed as a product of at most 2^{n-1} '3'-cycles. □

Remark 1. Lemma 1 is a well-known result in permutation group theory [11]. We give a proof in order to analyze the number of '3'-cycles which will be used in the decomposition process of our synthesis algorithm.

Definition 5 (NOT gate). *A NOT gate N_j connects an inverter to the j-th wire, i.e.: $P_j = B_j \oplus 1$; $\quad P_i = B_i, \quad if \quad i \neq j. \quad 1 \leq j \leq n$.*

Definition 6 ('k'-CNOT gate). *A 'k'-Controlled-NOT (CNOT) gate $C_{i_1, i_2, \cdots, i_k; j}$ is defined as follows:*

- *If $m \neq j$, then $P_m = C_{i_1, i_2, \cdots, i_k; j}(B_m) = B_m$.*
- *If $m = j$, and $B_{i_1} = \cdots = B_{i_k} = 1$, then $P_j = C_{i_1, i_2, \ldots, i_k; j}(B_j) = B_j \oplus 1$, else, $P_j = B_j$.*

A Toffoli gate is a '2'-CNOT gate where two inputs control an output of another input.

3 Theoretical Results

Lemma 2. *If $n \geq 5$ and $2 \leq k \leq n-3$, then any '$(k+1)$'-CNOT gate can be constructed by $(3 \times 2^{k-1} - 2)$ '2'-CNOT (Toffoli) gates without ancilla qubit. In particular, '$(n-2)$'-CNOT gate can be constructed by $(3 \times 2^{n-4} - 2)$ '2'-CNOT gates without ancilla qubit.*

Proof. Given a '$(k+1)$'-CNOT gate $C_{i_1,\cdots,i_k,i_{k+1};j}$, since $n \geq 5$ and $2 \leq k \leq n-3$, there is h, $1 \leq h \leq n$, where h is different from $i_1,\cdots,i_k,i_{k+1},j$. In other words, among these n qubits, there is a qubit B_h different from the qubits $B_{i_1},\cdots,B_{i_{k+1}},B_j$. We will prove the following equation:

$$C_{i_1,\cdots,i_k,i_{k+1};j} = (C_{i_1,\cdots,i_k;h} * C_{h,i_{k+1};j})^2 \tag{4}$$

Consider the outputs of the left side of the equation 4, we have:

$$\begin{aligned} P_h &= B_h \oplus (B_{i_1} \cdots B_{i_k}) \oplus (B_{i_1} \cdots B_{i_k}) \\ P_j &= B_j \oplus B_{i_{k+1}}(B_h \oplus B_{i_1} \cdots B_{i_k}) \oplus B_{i_{k+1}} B_h \\ &= B_j \oplus (B_{i_1} \cdots B_{i_{k+1}}) \end{aligned}$$

Therefore, equation 4 holds. □

This equation tells us that any '$(k+1)$'-CNOT gate can be constructed by two 'k'-CNOT gates and two '2'-CNOT gates. Using this equation recursively, any '$(k+1)$'-CNOT gate can be constructed by $3 \times 2^{k-1} - 2$ number of '2'-CNOT gates.

For any three different bit vectors u, s and t, the following matrix P is called the characteristic matrix of the '3'-cycle permutation (u,s,t).

$$P = \begin{bmatrix} u \\ s \\ t \end{bmatrix} = \begin{bmatrix} u_1, u_2, \ldots, u_n \\ s_1, s_2, \ldots, s_n \\ t_1, t_2, \ldots, t_n \end{bmatrix}$$

In this 3-row matrix P, a column having different elements (different bits) is called a heterogeneous column. Otherwise, it is called homogeneous column.

Definition 7 (Neighboring '3'-cycle). *If the characteristic matrix P of a '3'-cycle (u,s,t) has only two heterogeneous columns, this '3'-cycle (u,s,t) is called a neighboring '3'-cycle. In other words, only two bits are different among these three assignment vectors u, s and t.*

Lemma 3. *Any neighboring '3'-cycle permutation (u,s,t) can be generated by four '$(n-2)$'-CNOT gates and at most $2n$ NOT gates without ancilla qubit.*

Proof. Suppose in P, the i^{th} and j^{th} columns are heterogeneous, the other columns are all 1's (if some are 0's, we can first apply at most $(n-2)$ NOT gates to make them become 1's, then after applying four '$(n-2)$'-CNOT gates, we apply these NOT gates to restore these 1's that became 0's). The vector values in the i^{th} and j^{th} columns are three out of these four vectors: $\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle$.

There are 8 cases and we will prove one of them: $(u, s, t) = (\langle 1, 0\rangle, \langle 1, 1\rangle, \langle 0, 1\rangle)$. Let k, l be two indexes different from i, j, respectively. If $n > 4$, we denote $x = \{1, 2, \cdots, n\} - \{i, j, k, l\}$, namely, the index numbers except i, j, k, l. $B_x = \prod_{h \neq i,j,k,l} B_h$ is the product of variables B_h except B_i, B_j, B_k, B_l.

1. If $(u, s, t) = (\langle 1, 0\rangle, \langle 1, 1\rangle, \langle 0, 1\rangle)$, then

$$\begin{bmatrix} u \\ s \\ t \end{bmatrix} = \begin{array}{c} 1,\cdots,i,\cdots,j,\cdots,n \\ \begin{bmatrix} 1,\cdots,1,\cdots,0,\cdots,1 \\ 1,\cdots,1,\cdots,1,\cdots,1 \\ 1,\cdots,0,\cdots,1,\cdots,1 \end{bmatrix} \end{array} \overset{(u,s,t)}{\rightarrow} \begin{array}{c} 1,\cdots,i,\cdots,j,\cdots,n \\ \begin{bmatrix} 1,\cdots,1,\cdots,1,\cdots,1 \\ 1,\cdots,0,\cdots,1,\cdots,1 \\ 1,\cdots,1,\cdots,0,\cdots,1 \end{bmatrix} \end{array} = \begin{bmatrix} s \\ t \\ u \end{bmatrix} \tag{5}$$

We have the following equation:

$$(u, s, t) = C_{i,k,x;j} * C_{j,l,x;i} * C_{i,k,x;j} * C_{j,l,x;i} \tag{6}$$

After a CNOT gate, only one output qubit changes its value.
After the first CNOT gate, only the second qubit B_j changes it value. Let the changed value be $B_j^{(1)}$:

$$B_j^{(1)} = B_j \oplus B_i B_k B_x.$$

After the second CNOT gate, only the first qubit B_i changes it value. Let the changed value be $B_i^{(2)}$:

$$\begin{aligned} B_i^{(2)} &= B_i \oplus B_j^{(1)} B_l B_x \\ &= B_i \oplus B_j B_l B_x \oplus B_i B_k B_l B_x. \end{aligned}$$

After the third CNOT gate, only the second qubit changes it value again. Let the changed value be $B_j^{(3)}$:

$$\begin{aligned} B_j^{(3)} &= B_j^{(1)} \oplus B_i^{(2)} B_k B_x \\ &= B_j \oplus (B_i \oplus B_j) B_k B_l B_x. \end{aligned}$$

After the forth CNOT gate, only the first qubit changes it value. Let the changed value be $B_i^{(4)}$:

$$\begin{aligned} B_i^{(4)} &= B_i^{(2)} \oplus B_j^{(3)} B_l B_x \\ &= B_i \oplus B_j B_k B_l B_x. \end{aligned}$$

These exactly consist with the truth table 5 of the reversible circuit (u, s, t). Therefore, equation 6 holds.
Similarly, we have the following equations:

2. If $(u, s, t) = (\langle 1, 0\rangle, \langle 1, 1\rangle, \langle 0, 1\rangle)$, then

$$(u, s, t) = C_{j,k,x;i} * C_{i,l,x;j} * C_{j,k,x;i} * C_{i,l,x;j} \tag{7}$$

3. If $(u, s, t) = (\langle 0, 0\rangle, \langle 1, 0\rangle, \langle 1, 1\rangle)$, then

$$(u, s, t) = N_j * C_{j,k,x;i} * C_{i,l,x;j} * C_{j,k,x;i} * C_{i,l,x;j} * N_j \tag{8}$$

4. If $(u, s, t) = (\langle 0,0\rangle, \langle 1,1\rangle, \langle 1,0\rangle)$, then

$$(u, s, t) = N_j * C_{i,k,x;j} * C_{j,l,x;i} * C_{i,k,x;j} * C_{j,l,x;i} * N_j \tag{9}$$

5. If $(u, s, t) = (\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,1\rangle)$, then

$$(u, s, t) = N_i * C_{j,k,x;i} * C_{i,l,x;j} * C_{j,k,x;i} * C_{i,l,x;j} * N_i \tag{10}$$

6. If $(u, s, t) = (\langle 0,0\rangle, \langle 1,1\rangle, \langle 0,1\rangle)$, then

$$(u, s, t) = N_i * C_{i,k,x;j} * C_{j,l,x;i} * C_{i,k,x;j} * C_{j,l,x;i} * N_i \tag{11}$$

7. If $(u, s, t) = (\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle)$, then

$$(u, s, t) = N_i * N_j * C_{i,k,x;j} * C_{j,l,x;i} * C_{i,k,x;j} * C_{j,l,x;i} * N_j * N_i \tag{12}$$

8. If $(u, s, t) = (\langle 0,0\rangle, \langle 1,0\rangle, \langle 0,1\rangle)$, then

$$(u, s, t) = N_i * N_j * C_{j,k,x;i} * C_{i,l,x;j} * C_{j,k,x;i} * C_{i,l,x;j} * N_j * N_i \tag{13}$$

Therefore, Lemma 3 holds. □

Lemma 4. *If the characteristic matrix of a '3'-cycle (u, s, t) has k heterogeneous columns, then there is an ordered set $M = \{a_1, a_2, \cdots, a_m\}$ with m vector assignments, such that $u, s, t \in M$ and $a_1, a_m \in \{u, s, t\}$. For any r, $1 \le r < m$, there are only r bits different among $a_i, \cdots, a_{i+r}$, and $m \le k + \lfloor \frac{k}{3} \rfloor + 1$.*

Proof. Let k_1 be the number of same bits and k_2 be the number of different bits between u and s, respectively. Let k_{i1} and k_{i2} be the number of same and different bits between s and t in k_i bits, respectively, $i = 1, 2$. Then, $k_{11} + k_{12} + k_{21} = k$, and $k_{11} + k_{12} + k_{21} + k_{22} = n$.

1. According to the order u, s, t, we set $a_1 = u, a_i = s, a_{m_1} = t$, such that $i = 1 + k_{11} + k_{12}, m_1 = i + k_{12} + k_{21} = k + k_{12} + 1$. We add $i - 1$ vectors $a_2, \cdots, a_{i-1}$ without changing the same bits between u and s. Each time we change only one bit in the different bits between u and s. We apply the same method to vectors $a_{i+1}, \cdots, a_{m_1-1}$.
   Then we get a ordered set M_1 with m_1 vectors such that there are only r bits different among r neighboring vectors, and $m_1 = k + k_{12} + 1$.
2. Similarly, according to the order u, t, s, we can get an ordered set M_2 with m_2 vectors such that there are only r bits different among r neighboring vectors, and $m_2 = k + k_{21} + 1$.
3. According to the order s, u, t, we can get an ordered set M_3 with m_3 vectors such that there are only r bits different among r neighboring vectors, and $m_3 = k + k_{11} + 1$.

We choose a minimal set M from these three sets M_1, M_2, M_3. Let $M = \{a_1, a_2, \cdots, a_m\}$ be an ordered set with m vector assignments such that there are only r bits different among r neighboring vectors for any $r \ge 2$, and $m \le k + \lfloor \frac{k}{3} \rfloor + 1$. □

Theorem 1. *All n-qubit even binary reversible circuits can be constructed by NOT and '2'-CNOT gates without ancilla qubit.*

Proof. For any even reversible circuit, its permutation on assignments of inputs can be expressed by the product of some '3'-cycles (Lemma 1).

By lemma 4, for any '3'-cycle (u, s, t), there are m vector assignments $a_1, a_2, \ldots, a_m$, where $a_1, a_m \in \{u, s, t\}$, such that there is only one bit different between a_i and a_{i+1}. We can decompose the '3'-cycle $(u, s, t) = (a_1, a_{1+r_1}, a_{1+r_1+r_2})$, or $(u, s, t) = (a_1, a_{1+r_1+r_2}, a_{1+r_1})$, where $1 + r_1 + r_2 = m$, applying the following equations:

$$(a_1, a_{1+r_1}, a_{1+r_1+r_2}) = (a_{1+r_1}, a_{1+r_1+r_2}, a_{1+r_1+1}) * (a_1, a_{1+r_1}, a_{1+r_1+1}) \qquad (14)$$

$$(a_1, a_{1+r_1+r_2}, a_{1+r_1}) = (a_1, a_{1+r_1+1}, a_{1+r_1}) * (a_{1+r_1}, a_{1+r_1+1}, a_{1+r_1+r_2}) \qquad (15)$$

$$(a_1, a_{1+r_1}, a_{1+r_1+1}) = (a_{r_1}, a_{1+r_1}, a_{1+r_1+1}) * (a_1, a_{1+r_1}, a_{r_1}) \qquad (16)$$

$$(a_1, a_{1+r_1+1}, a_{1+r_1}) = (a_1, a_{r_1}, a_{1+r_1}) * (a_{r_1}, a_{1+r_1+1}, a_{1+r_1}) \qquad (17)$$

By recursively applying equation 14 or 15 (if $r_2 > 1$) and equation 16 or 16 (if $r_1 > 1$), we can decompose (u, s, t) into neighboring '3'-cycles.

By Lemma 3, any neighboring '3'-cycle can be constructed by NOT and '$(n-2)$'-CNOT gates. Applying equation 4 recursively, it can be constructed by NOT and '2'-CNOT gates.

Therefore, all n-qubit even binary reversible circuits can be constructed by NOT, '2'-CNOT gates without ancilla qubit. □

Next, we establish the upper bounds on the number of '2'-CNOT gates and the number of NOT gates used in our construction.

Theorem 2. *The number of '2'-CNOT gates used in the above construction is no more than $(n + \lfloor \frac{n}{3} \rfloor)(3 \times 2^{2n-3} - 2^{n+2})$. The number of NOT gates is no more than $4n(n + \lfloor \frac{n}{3} \rfloor)2^n$.*

Proof. Let $g(r_1, r_1 + r_2)$ be the number of '$(n-2)$'-CNOT gates. We need to synthesize a '3'-cycle $(a_j, a_{j+r_1}, a_{j+r_1+r_2})$ or $(a_j, a_{j+r_1+r_2}, a_{j+r_1})$, $r_1 \geq 1, r_2 \geq 1$.

According to Lemma 3, $g(1, 2) = 4$.

Using equation 14 or 15, we get $g(r_1, r_1 + r_2) \leq g(1, r_2) + g(r_1, r_1 + 1)$.

Using equation 16 or 17, we get $g(r_1, r_1 + 1) \leq g(1, 2) + g(r_1 - 1, r_1)$. Recursively, we get $g(r_1, r_1 + 1) \leq (r_1 - 1)g(1, 2)$. Similarly, $g(1, r_2) \leq (r_2 - 1)g(1, 2)$. Therefore,

$$g(r_1, r_1 + r_2) \leq (r_1 + r_2 - 1)g(1, 2) = 4(r_1 + r_2 - 1) = 4(m - 2). \qquad (18)$$

From equation 18 and Lemma 4, we have

$$g(r_1, r_1 + r_2) \leq 4(k + \lfloor \frac{k}{3} \rfloor - 1) < 4(n + \lfloor \frac{n}{3} \rfloor). \qquad (19)$$

Based on Lemmas 1, 2, 4, and equation 19, the upper bound of the number of '2'-cycles that are needed to synthesize any given even reversible circuit is:

$$2^{n-1} \times 4(n + \lfloor \frac{n}{3} \rfloor) \times (3 \times 2^{n-4} - 2)$$

$$= (n + \lfloor \frac{n}{3} \rfloor)(3 \times 2^{2n-3} - 2^{n+2}).$$

In terms of Lemmas 1, 3, and equation 19, the upper bound on the number of NOT gates is:

$2^{n-1} \times 4(n + \lfloor \frac{n}{3} \rfloor) \times 2n = 4n(n + \lfloor \frac{n}{3} \rfloor)2^n$. □

Remark 2. The upper bound for NOT gates can be reduced by removing pair of the adjacent same NOT gates. (There is commutativity in the product of NOT gates). This is illustrated by the example in the next section.

4 Algorithm and Synthesis Example

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

Algorithm

Step 1. Rewrite f as the product of '3'-cycles by using equations 2 and 3.

Step 2. For every '3'-cycle (u, s, t), find an ordered set $M = \{a_1, \cdots, a_k\}$ according to Lemma 4. Based on equations 14, 15, 16, and 17, rewrite this '3'-cycle (u, s, t) as a product of some neighboring '3'-cycles (a_i, a_{i+1}, a_{i+2}) or (a_i, a_{i+2}, a_{i+1}).

Step 3. Synthesize all neighboring '3'-cycles by NOT and '$(n-2)$'-CNOT gates by Lemma 3 and remove pair of the adjacent same NOT gates.

Step 4. Decompose each '$(n-2)$'-CNOT gates into $(3 \times 2^{n-4} - 2)$ '2'-CNOT gates by using equation 4.

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

Table 1. An even binary reversible function f

input						output					
B_1	B_2	B_3	B_4	B_5	encoding	P_1	P_2	P_3	P_4	P_5	encoding
0	1	0	1	0	b_1	1	1	1	1	0	b_5
0	1	1	1	0	b_2	1	0	1	1	0	b_4
1	0	0	1	0	b_3	0	1	0	1	0	b_1
1	0	1	1	0	b_4	1	0	0	1	0	b_3
1	1	1	1	0	b_5	0	1	1	1	0	b_2

Therefore, $f = (b_1, b_5, b_2, b_4, b_3)$.

Step 1. Decompose f into '3'-cycles by equation 2. $f = (b_1, b_5, b_2)(b_1, b_4, b_3)$.

Step 2. Decompose each '3'-cycle into the product of neighboring '3'-cycles.

– For '3'-cycle (b_1, b_5, b_2). This is a neighboring '3'-cycle.
– For '3'-cycle (b_1, b_4, b_3). Using Lemma 4, we get an ordered set $M = \{a_1, a_2, a_3, a_4\}$, where $a_1 = b_1, a_2 = \langle 0, 0, 0, 1, 0\rangle$ (a new vector), $a_3 = b_3, a_4 = b_4$. Using equation 17, we get

$$(b_1, b_4, b_3) = (a_1, a_4, a_3) = (a_1, a_2, a_3)(a_2, a_4, a_3).$$

Step 3. By applying NOT gates and equation 11, we have:

$$(b_1, b_5, b_2) = N_5 * N_1 * C_{1,2,5;3} * C_{3,4,5;1} * C_{1,2,5;3} * C_{3,4,5;1} * N_1 * N_5.$$

By applying NOT gates and equation 13, we have:

$$(a_1, a_2, a_3) = N_5 * N_3 * N_2 * N_1 * C_{2,3,5;1} * C_{1,4,5;2} * C_{2,3,5;1} * C_{1,4,5;2} * N_1 * N_2 * N_3 * N_5.$$

By applying NOT gates and equation 9, we have:

$$(a_2, a_4, a_3) = N_5 * N_3 * N_2 * C_{1,2,5;3} * C_{3,4,5;1} * C_{1,2,5;3} * C_{3,4,5;1} * N_2 * N_3 * N_5.$$

By removing pair of the adjacent same NOT, f is decomposed into the product of 8 NOT gates (without removing NOT gates, there are 18 NOT gates) and 12 '3'-CNOT gates.

$$\begin{aligned} f = & N_5 * N_1 * C_{1,2,5;3} * C_{3,4,5;1} * C_{1,2,5;3} * C_{3,4,5;1} \\ & * N_3 * N_2 * C_{2,3,5;1} * C_{1,4,5;2} * C_{2,3,5;1} * C_{1,4,5;2} * N_1 \\ & * C_{1,2,5;3} * C_{3,4,5;1} * C_{1,2,5;3} * C_{3,4,5;1} * N_2 * N_3 * N_5. \end{aligned}$$

Step 4. Using equation 4, decompose each '3'-CNOT gate into 4 '2'-CNOT gates.

The synthesis process is finished, and f is decomposed into the product of 8 NOT gates and 48 '2'-CNOT gates.

5 Complexity Analysis

Theorem 3. *The time complexity of the synthesis algorithm is no more than* $\frac{10}{3}n^2 \cdot 2^n$.

Proof. Calculate the time complexity of each step, then add them up.

Remark 3. Our method is constructive, 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 method. The computational complexity of our synthesis algorithm is exponentially lower than the complexity of breadth-first search based synthesis algorithm. The space complexity of any breadth-first search based synthesis algorithm for n qubits even reversible circuit is more than $(2^n)!/2$, since in the worst case, it at least needs to remember all $(2^n)!/2$ even reversible circuits. This is impossible even when $n = 4$ since $(2^4)!/2 \approx 1.0 \times 10^{13}$. The time complexity is also greater than $(2^n)!/2$, because in the worst case, it at least needs to compute all even 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 $(2^n)!/2$.

6 Conclusions

In this paper, we present a constructive proof that any even reversible circuit can be implemented by NOT and Toffoli gates. Our proof is essentially a construction of the reversible circuit and we also give the upper bounds for the number of NOT gates and Toffoli gates in such circuit. We present a constructive synthesis algorithm based on this proof and give a synthesis example based on this algorithm, which shows that even by hand, synthesizing any 5-qubit even 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] Michael A. Nielsen and Isaac 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. Proc. DAC, 2002, New Orleans, Louisiana, 28.4

[3] R. Landauer: Irreversibility and heat generation in the computational process. IBM Journal of Research and Development. **5** (1961) 183–191

[4] C. Bennett: Logical reversibility of computation. IBM Journal of Research and Development. **17** (1973) 525–532

[5] E. Fredkin and T. Toffoli: Conservative logic. Int. Journal of Theoretical Physics. **21** (1982) 219–253

[6] D. Deutsch: Quantum computational networks. Royal Society of London Series A. **425** (1989) 73–90

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

[8] T. Toffoli: Bicontinuous extensions of invertible combinatorial functions. Mathematical Systems Theory. **14** (1981) 13–23

[9] G. Yang and W. N. N. Hung and X. Song and M. Perkowski: Majority-Based Reversible Logic Gates. Theoretical Computer Science. **334** (2005) 295–274

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

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

[12] Alexis De Vos: Reversible computing. Qutantum Electronics, **23** (1999) 1–49

[13] L. Storme and Alexis De Vos, A. and G. Jacobs: Group theoretical aspects of reversible logic gates. Journal of Universal Computer Science, **5** (1999) 307–321

[14] D. Michael Miller and Dmitri Maslov and Gerhard W. Dueck: A Transformation Based Algorithm for Reversible Logic Synthesis. Proc. DAC, 2003, 318-323