Theoretical Computer Science 412 (2011) 1606–1613

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

# Realization and synthesis of reversible functions

Guowu Yang [a], Fei Xie [b], William N.N. Hung [c,*], Xiaoyu Song [d], Marek A. Perkowski [d]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China
[b] Department Computer Science, Portland State University, Portland, OR 97207, USA
[c] Synopsys Inc., Mountain View, CA 94043, USA
[d] Department Electrical & Computer Engineering, Portland State University, Portland, OR 97207, USA

ARTICLE INFO

*Article history:*
Received 3 July 2009
Received in revised form 28 July 2010
Accepted 14 November 2010
Communicated by M. Hirvensalo

*Keywords:*
Reversible logic
Group theory
Permutation

ABSTRACT

Reversible circuits play an important role in quantum computing. This paper studies the realization problem of reversible circuits. For any $n$-bit reversible function, we present a constructive synthesis algorithm. Given any $n$-bit 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 and space complexities of the algorithm are $\Omega(n \cdot 4^n)$ and $\Omega(n \cdot 2^n)$, respectively. The computational complexity of our synthesis algorithm is exponentially lower than that of breadth-first search based synthesis algorithms.



## 1. Introduction

Reversible computing provides a way to improve the energy efficiency beyond the von Neumann–Landauer limit [1,2]. It has been shown that any computing system of irreversible logic gates leads inevitably to energy dissipation [2–4]. To avoid power dissipation, circuits must be constructed [3,4] from reversible gates. Reversible circuit plays an important role in quantum computing [5,6]. There is a lot of research [7,8,4,6,9–18] on the construction of reversible logic gates.

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 $n$ $(n > 2)$ bits can be constructed by NOT and '$(n-1)$'-CNOT gates. We also investigate the realization problem of 3-bit reversible circuits specifically. Using group theory, we present two sets of new 3-bit reversible logic gates. We show that any 3-bit reversible logic circuit is realizable by cascading NOT and Feynman gates, and at most one instance of the proposed gates. We present a novel, concise and constructive proof based on group theory. Our synthesis algorithm is based on a constructive proof, where the numbers of '$(n-1)$'-CNOT 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 algorithm is $\Omega(n \cdot 4^n)$. In contrast, a search based synthesis algorithm may have a worst case time complexity of $(2^n)!$.

The rest of the paper is organized as follows. In Section 2, we present definitions of reversibility, permutation, and some elementary reversible logic gates. Then, in Section 3, we proceed to prove a few lemmas for $n$-bit reversible gates 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 a synthesis algorithm in Section 4 and present some synthesis examples. We analyze the complexity of our algorithm in Section 5. Our conclusion is given in Section 6.

* Corresponding author. Tel.: +1 650 584 5204.
*E-mail address:* william_hung@alumni.utexas.net (W.N.N. Hung).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.11.031

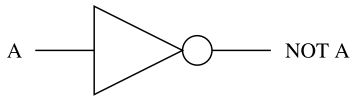


**Fig. 1.** NOT gate.

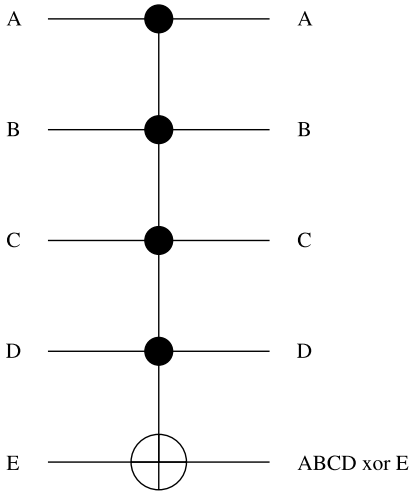


**Fig. 2.** 4-CNOT gate.

## 2. Preliminaries

In this section, we introduce some basic concepts and results on permutation group theory from [19] and binary reversible logic from [20–22].

**Definition 1** (*Binary Reversible Gate*)**.** Let $B = \{0, 1\}$. Given any binary logic circuit $f$ with $n$ inputs and outputs, we can denote it as a binary multiple-output function $f : B^n \to B^n$. Let $\langle B_1, \ldots, B_n\rangle \in B^n$ and $\langle P_1, \ldots, P_n\rangle \in B^n$ be the input and output vectors, where $B_1, \ldots, B_n$ are input variables and $P_1, \ldots, P_n$ 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 $n$-bit binary reversible gate. There are $(2^n)!$ different $n$-bit binary reversible circuits.

We introduce a permutation group [23,21,19] and its relationship with reversible circuits.
A mapping $s : M \to M$ can be written as:

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

Here we use a product of disjoint cycles as an alternative notation for a mapping [19]. 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 $g$.

An $n$-bit 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 an $n$-bit reversible circuit from a permutation in $S_{2^n}$.

**Definition 2** (*NOT Gate*)**.** A NOT gate $N_j$ connects an inverter to the $j$th wire, i.e.: $P_j = B_j \oplus 1$; $P_i = B_i$, *if* $i \neq j$, $1 \leq j \leq n$.

An example NOT gate is shown in Fig. 1.

**Definition 3** (‘$n-1$’-*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, $P_j = B_j$.

A 1-CNOT gate is also called a Feynman gate. A 2-CNOT gate is also called a Toffoli gate. A ‘$n-1$’-CNOT gate is a generalized Toffoli gate where $n$ inputs control the output of another input. An example 4-CNOT gate is shown in Fig. 2.

For Feyman gates, we also use $Fe_{i,j}$ to denote the gate:

- If $m = i$, then $P_m = B_i \oplus B_j$.
- If $m \neq i$, then $P_m = B_m$.

On 3-bit circuits, the gate can be represented as the following permutation: $Fe_{1,2} = (3, 4)(7, 8)$. Similarly, we have: $Fe_{1,3} = (5, 6)(7, 8)$, $Fe_{2,1} = (2, 4)(6, 8)$, $Fe_{2,3} = (5, 7)(6, 8)$, $Fe_{3,1} = (2, 6)(4, 8)$, $Fe_{3,2} = (3, 7)(4, 8)$.

**Definition 4** (*SWAP Gate*)**.** A SWAP gate $Ex_{i,j}$ is defined as follows:

- If $m = i$, then $P_m = B_j$.
- If $m = j$, then $P_m = B_i$.
- If $m \neq i, j$, then $P_m = B_m$.

## 3. '*n*'-bit theoretical results

In this section, we show the process to constructively synthesize any '$n$'-bit reversible circuit by NOT and '$n - 1$'-CNOT gates without ancilla bits. It will be used in our synthesis algorithm in Section 4.

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

**Proof.** Any permutation can be written as a 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. □

**Definition 5** (*Neighboring '2'-Cycle*)**.** Given two integers $u, s \in 1, \ldots, 2^n$, both $u$ and $s$ can be encoded using $n$ bits (binary representation). If the $n$-bit encodings for $u$ and $s$ are the same except for only one bit, we call the permutation $(u, s)$ a neighboring '2'-cycle. This is because their binary representations differ in one bit only.

**Lemma 2.** *Given two integers u and s, and in their binary representations:*

1. *there is only one bit $B_j$ different; and*
2. *all other bits are the same between u and s:*
   (a) *l bits are* 0*s; and*
   (b) *remaining bits are all* 1*s.*

*Suppose those l zero bits are $B_{i_1}, \ldots, B_{i_l}$, we have:*

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

**Proof.** We need to show that R.H.S. will turn the number $u$ into $s$, and the number $s$ into $u$, and it will not change any other numbers.

After using the first set of NOT gates $N_{i_1}, \ldots, N_{i_l}$, the binary numbers $u$ and $s$ become $u'$ and $s'$ with only one bit (the $j$th bit) different and all other bits are 1's. Then connecting $C_j$, we can exchange $u'$ and $s'$, which becomes $s'$ and $u'$ respectively. Finally, connecting the second set of NOT gates $N_{i_1}, \ldots, N_{i_l}$, the values of bits $B_{i_1}, \ldots, B_{i_l}$ become all zeros, and the binary numbers $s'$ and $u'$ become $s$ and $u$ respectively.

Suppose we are given a number $t$, where $t \neq u$ and $t \neq s$, there exists a bit (the $k$th bit) in the binary representation of $t$ that is different from the corresponding bit in the binary representation of $u$ and $s$, such that $k \neq j$. After using the first set of NOT gates $N_{i_1}, \ldots, N_{i_l}$, the number $t$ becomes $t'$. But the $k$th bit in the binary representation of $t'$ is still zero. Then after the action of gate $C_j$, the number $t'$ is still $t'$ (unchanged). This is because the $k$th bit is zero, which disables the controlled-NOT operation. Hence, after the second set of NOT gates $N_{i_1}, \ldots, N_{i_l}$, $t'$ becomes $t$ again.

Therefore, the R.H.S. will only exchange the numbers $u$ and $s$, and nothing else. □

**Lemma 3.** *If two n-dimension vectors u, s have k bits different, then there is an ordered set $M = \{d_1, d_2, \ldots, d_{k+1}\}$ 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, we have $d_1 = u$, $d_{k+1} = s$. Then, in each '2'-cycle we change one bit in the $k$ different bits between $u$ and $s$. So we get an ordered set $M = \{d_1, d_2, \ldots, d_{k+1}\}$ 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 $d_{i+1}$. Hence Eq. (5) is valid.

Alternatively, we can denote the image of $d$ after the action of R.H.S. as $I(d)$. Then $I(d_1) = d_{k+1}$, $I(d_{k+1}) = d_1$, and for any $1 < j < k + 1$, $I(d_j) = d_j$. Thus the equation is correct. □

**Remark 1.** In order to make the number of NOT gates as small as possible, we give two rules for constructing the ordered set $M$.

**Table 1**
The ordered set $M$ of $u$ and $s$ encoded using the Gray-code trick from page 191 of [5].

| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | 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 | $d_5 = u$ |

- 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 1.

**Remark 2.** There is commutativity in the product of NOT gates, and we can remove adjacent pairs of identical NOT gates.

**Lemma 4.** *Suppose there are $j$ bits different between $u$ and $s$ in decomposing the '2'-cycle $(u, s)$ to NOT gates and '$n-1$'-CNOT gates, and*

- *there are $j_0$ zero bits in these bits in $d_1$, and*
- *there are $j_1$ one bits in these bits in $d_1$, and*
- *$j_0 \le j_1$,*

*then the number of NOT gates is no more than $2(j-2)+2(j_0-1)$ if $j_0 \ge 1$; or $2(j-1)$ if $j_0 = 0$.*

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

**Theorem 1.** *For a given $n$-bit reversible circuit $f$, if there are $N$ distinct input patterns that are different from their corresponding output patterns, where $N \le 2^n$, and the other $(2^n - N)$ input patterns are the same as their corresponding output patterns, 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 bit.*

**Proof.** According to Eq. (3), this reversible circuit can be decomposed to at most $N-1$ '2'-cycles. According to Lemmas 2 and 3, 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 $n$-bit 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 bit.*

**Proof.** Let $(d_1, d_2), (d_3, d_4), \ldots, (d_{m-1}, d_m)$, where $m = 2^n$, be all the '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 using Eq. (3) of Lemma 1 to optimally decompose $p$ to some neighboring 2-cycles based on Eq. (5), if $(d_{2i-1}, d_{2i})$ is in the decomposition, then $(d_{2i-1}, a)$ or $(d_{2i}, b)$ will not be in this decomposition. We can prove 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 Eq. (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 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 Eq. (4) of Lemma 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 the binomial coefficient [25]. Using Lemma 4 and properties of binomial coefficient, when $n = 2k-1$, $k \ge 2$, $n$ is an odd number,

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

**Table 2**
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 | $a_{10}$ | 0 | 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 | $a_1$ |

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

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

**Remark 3.** The upper bound for NOT gates can be reduced by further removing adjacent pairs of identical NOT gates. This is illustrated by the example in the next section.

**Remark 4.** The product of all '2'-cycles with maximal $n$ different bits indeed is the product of all $n$ different NOT gates. Thus the approach of directly using Eq. (3) and Lemma 3 has some defects. We should consider the NOT gate before using Eq. (3).

The idea of considering the NOT gate before using Eq. (3) is given as follows. Consider the truth table of a given reversible circuit, each output bit has $2^n$ values, we compare them with the input values. If the number of different values is bigger than $2^{n-1}$, we apply a NOT gate to this bit. After processing with all bits, we count the changed vectors. If the number of the changed vectors is less than that of the original circuit, we decompose the reversible circuit with the inserted NOT gates using Eq. (3) and Lemma 3. Otherwise, we decompose the original reversible circuit. The decomposition algorithm and examples are given in the next section.

## 4. 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 bits.

**Algorithm**:

**Step 1**. Check the truth table of $f$ to determine before using Eq. (3) and Lemma 3, whether we need NOT gates or not.

**Step 2**. After Step 1, write the reversible circuit as a product of cycles. 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 minimal number. The basic idea to decompose the reversible circuit by Eq. (3) is to break the mapping relation from $d_j$ to $d_{j+1}$ without increasing the number of different bits between adjacent vectors.

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

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

**Step 3**. Decompose every '2'-cycle by NOT and '$n-1$'-CNOT gates using Lemma 3, two rules in Remark 1, Lemma 2, and removing adjacent pairs of identical NOT gates as much as possible.

**Example 1.** Given a binary reversible circuit $f$ which has a truth table shown in Table 2.

From the truth table, $f = (a_1, a_3, a_4, a_{16})(a_2, a_6, a_{14})$.

**Step 1**. The total 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 using Eq. (6).

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

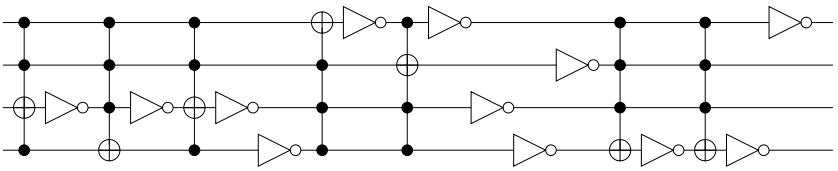

**Fig. 3.** Decomposed circuit for $f$.

**Table 3**
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 | $a_{10}$ | 1 | 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 | $a_2$ |

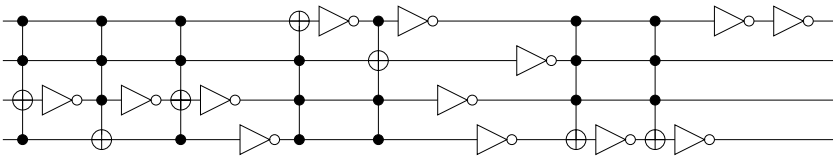

**Fig. 4.** Decomposed circuit for $g$.

**Step 3**. Using Eqs. (4) and (5), 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,

$$f = C_3 * N_3 * C_4 * N_3 * C_3 * N_3 * N_4 * C_1 * N_1 * C_2 * N_1 * N_3 * N_4 * N_2 * C_4 * N_4 * C_4 * N_4 * N_2.$$

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

**Example 2.** Given a binary reversible circuit $g$ which has a truth table shown in Table 3.

**Step 1**. Only the output $P_1$ has over $2^{4-1} = 8$ different values with input $B_1$. So we need to cascade a NOT gate $N_1$ after $g$, shown in Fig. 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 5.** 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.

## 5. Complexity analysis

In this section, we analyze the computation complexity of our algorithm. Compared with 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* $\Omega(n \cdot 4^n)$.

**Proof.** The time complexity of Step 1 is $n \cdot 2^n$, since we need to check the whole values in truth table. In Step 2, to get Eq. (6), in the worst case ($k = 2^n$), we need $n \cdot 2^n$ computations. And we need recursively use Eq. (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 the worst case. According to Lemmas 2 and 3, to 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:

$$n \cdot 2^n + n \cdot 4^n/2 + 6n^2 \cdot 2^n = \Omega(n \cdot 4^n). \quad \square$$

**Remark 6.** 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 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 $n$ bits reversible circuit is more than $(2^n)!$, since in the worst case, it needs to remember all $(2^n)!$ reversible circuits. This is impossible even when $n = 4$ because $(2^4)! \approx 2.0 \times 10^{13}$. The time complexity is also greater than $(2^n)!$, because in the worst case, it needs to compute all reversible circuits. In fact, it also has to do a lot of equality comparisons 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 number of different 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 $6n \cdot 2^n$. $\square$

In the worst case, breadth-first search based synthesis algorithm needs to store all $(2^n)!$. So, the space complexity of our synthesis algorithm is still exponentially lower than that of breadth-first search based synthesis algorithm.

## 6. Conclusion

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

## Acknowledgement

The first author's work is supported by NSFC under Grant 60773205 and RFDP under Grant 20090185110006.

## References

[1] J. Von Neumann, Theory of Self-Reproducing Automata, University of Illinois Press, Champaign, IL, USA, 1966.
[2] R. Landauer, Irreversibility and heat generation in the computational process, IBM Journal of Research and Development 5 (1961) 183–191.
[3] C. Bennett, Logical reversibility of computation, IBM Journal of Research and Development 17 (1973) 525–532.
[4] E. Fredkin, T. Toffoli, Conservative logic, International Journal of Theoretical Physics 21 (1982) 219–253.
[5] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.
[6] K. Iwama, Y. Kambayashi, S. Yamashita, Transformation rules for designing CNOT-based quantum circuits, in: Proc. DAC, 2002, p. 28.4.
[7] D. Deutsch, Quantum computational networks, Royal Society of London Series A 425 (1989) 73–90.
[8] G.W. Dueck, D. Maslov, Reversible function synthesis with minimum garbage outputs, in: Proc. 6th Int. Symp. Representations and Methodology of Future Computing Technologies, RM2003, 2003.
[9] P. Kerntopf, Maximally efficient binary and multi-valued reversible gates, in: Proc. ULSI Workshop, 2001, pp. 55–58.
[10] P. Kerntopf, Synthesis of multipurpose reversible logic gates, in: Proc. EUROMICRO Symp. Digital Systems Design, 2002, pp. 259–266.
[11] A. Khlopotine, M. Perkowski, P. Kerntopf, Reversible logic synthesis by gate composition, in: Proc. IEEE/ACM Int. Workshop on Logic Synthesis, 2002, pp. 261–266.
[12] D. Maslov, 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.
[13] A. Mishchenko, M. Perkowski, Logic synthesis of reversible wave cascades, in: Proc. IEEE/ACM Int. Workshop on Logic Synthesis, 2002, pp. 197–202.

[14] P. Picton, A universal architecture for multiple-valued reversible logic, Mutiple Valued Logic: an International Journal 5 (2000) 27–37.
[15] T. Toffoli, Bicontinuous extensions of invertible combinatorial functions, Mathematical Systems Theory 14 (1981) 13–23.
[16] G. Yang, W.N.N. Hung, X. Song, M. Perkowski, Majority-based reversible logic gates, Theoretical Computer Science 334 (1–3) (2005) 259–274.
[17] X. Song, et al., Algebraic characteristics of reversible gates, Theory of Computing Systems 39 (2) (2006) 311–319.
[18] G. Yang, X. Song, W.N.N. Hung, 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.
[19] J.D. Dixon, B. Mortimer, Permutation Groups, Springer, New York, 1996.
[20] A. De Vos, Reversible computing, Qutantum Electronics 23 (1999) 1–49.
[21] L. Storme, A. De Vos, G. Jacobs, Group theoretical aspects of reversible logic gates, Journal of Universal Computer Science 5 (1999) 307–321.
[22] D.M. Miller, D. Maslov, G.W. Dueck, A transformation based algorithm for reversible logic synthesis, in: Proc. DAC, 2003, pp. 318–323.
[23] A. De Vos, B. Raa, L. Storme, Generating the group of reversible logic gates, Journal of Physics A: Mathematical and General 35 (2002) 7063–7078.
[24] M.I. Kargapolov, J.I. Merzljakov, Fundamentals of the Theory of Groups, Springer-Verlag, New York, 1979.
[25] K.P. Bogart, Introductory Combinatorics, Harcourt Brace Jovanovich, 1990.