Vol.2009-AL-127 No.2
2009/11/27

On Generating Test Sets for Detecting Stuck-at Faults in Reversible Circuits

Kaku Tabei[†1] and Toshinori Yamada[†1]

Reversible circuits are quite attractive because of the possibility of nearly energy-free computation. During constructing a reversible circuit, it is important to test the circuit and detect faults in the circuit. However, very few algorithms are known to generate a test set for detecting faults in a given reversible circuit.

In this paper, first of all, it is proved to be NP-hard to generate a minimum test set for detecting stuck-at faults in a given reversible circuit even when the circuit is restricted to use only three kinds of simple reversible gates, that is NOT, 1-CNOT, and Toffoli gates. Next, the paper presents a randomized algorithm to generate a test set for detecting stuck-at faults in a given reversible circuit. As far as the authors know, the proposed algorithm is the first one to guarantee that the expected time complexity is polynomial and that the size of the obtained test size is bounded. Finally, the effectiveness of the proposed algorithm is shown by experiments.

1. Introduction

Reversible circuits are quite attractive because of the possibility of nearly energy-free computation. Landauer [4] showed that traditional irreversible circuits necessarily dissipate energy due to the erasure of information. On the other hand, Bennett [1], and Fredkin and Toffoli [2] showed that reversible circuits can be performed with arbitrarily small energy dissipation. Futhermore, reversible circuits has potential applications in nanocomputing [5], digital signal processing [8], and quantum computing [6]. These facts give motivation to the study of reversible circuits.

During designing and constructing a reversible circuit, it is important to test the circuit and detect faults in the circuit. Patel *et al.* [7] pointed out that testing of reversible circuits is relatively easier than conventional irreversible circuits. However, it was shown by Tayu *et al.* [9, 10] that given a reversible circuit C, it is NP-hard to generate a minimum complete test set for stuck-at faults, which fix the values of wires in C to either 0 or 1. A large number of CNOT gates with 7 inputs are used in their proof of NP-hardness. This paper proves that it is NP-hard to generate minimum complete test sets even for NCT circuits, which are reversible circuits consisting of only CNOT gates with 3 or less inputs, that is NOT, 1-CNOT, and Toffoli gates. So, it seems to be quite difficult, or even impossible, to generate minimum complete test sets for practical reversible circuits.

On the other hand, very few algorithms for generating a complete test set for a given reversible circuit are known. Patel *et al.* [7] proved that, for any reversible circuit C, there exists a complete test set T with $|T| = O(\log |W(C)|)$, where $W(C)$ is the set of wires in C. However, the proof is non-constructive, and a polynomial-time algorithm are not known to generate a complete test set T with $|T| = O(\log |W(C)|)$ for a given reversible circuit C. Patel *et al.* [7] also presented an algorithm for finding a complete test set for a given reversible circuit, based on circuit decomposition and integer programming. The performance of their algorithm was shown by experiments, but not by theoretical analysis. This paper proposes an algorithm to generate a complete test set T with $|T| = O(\log |W(C)|)$ for a given reversible circuit, and shows that the expected time complexity of the proposed algorithm is polynomial. As far as the authors know, the proposed algorithm is the first one whose performance is shown by theoretical analysis. Moreover, by experiments, the performance of our algorithm is compared with that of the algorithm proposed by Patel *et al.* [7].

The rest of the paper is organized as follows. Section 2 gives terminologies on reversible circuits, CNOT circuits, and complete test sets for stuck-at faults. Section 3 presents the relation between the problem of generating minimum test sets for 1-target CNOT circuits and Not-All-Equal SAT, a variant of SAT(SATisfiability), a very famous NP-complete problem, and shows several results obtained from this relation. In Section 4, NP-hardness of generating a minimum complete test set for NCT circuits is proved. Section 5 proposes an expected polynomial-time algorithm for generating a complete test set for a given

†1 Division of Mathematics, Electronics and Informatics, Graduate School of Science and Engineering, Saitama University

reversible circuit, and analyzes the performance of the proposed algorithm theoretically. Furthermore, the proposed algorithm is compared with the algorithm by Patel *et al.* [7] by experiments. Finally, the paper concludes with Section 6.

2. Reversible Gates, Reversible Circuits, and Complete Test Sets

2.1 Reversible Gates

A logic gate is *reversible* if the mapping of inputs to outputs is bijective, that is, every distinct input yields a distinct output, and the number of output bits is equal to that of input ones. A reversible gate with k input bits and k output bits is called a $k \times k$ reversible gate.

2.2 Reversible Circuits

A circuit is reversible if the circuit consists of reversible gates and the mapping of inputs to outputs is bijective. A (reversible) circuit is called an identity circuit if its input is equal to its output.

Let n be a positive integer. An n-wire well-formed reversible circuit C is constructed recursively as follows.

(1) A circuit consisting of only n wires is a n-wire well-formed reversible circuit. One and the other ends of each wire is the input and output of the wire, respectively. Then, C is represented by $\varepsilon[1, 2, \ldots, n]$, or simply empty sequence. See Fig. 1(a).

(2) Let C be a circuit obtained from a n-wire well-formed reversible circuit C' and a $k \times k$ reversible gate G ($k \leq n$) by connecting the input wires of G with k of output wires of C', say $i_1, i_2, \ldots, i_k$. Then, C is also n-wire well-formed reversible circuit, and is denoted by $C = C', G[i_1, i_2, \ldots, i_k]$. The input wires of C are those of C', and the output ones of C are the k output wires of G together with the $n - k$ wires not connected with G. See Fig. 1(b).

(3) An n-wire well-formed reversible circuit is only constructed by (1) and (2).

An well-formed reversible circuit is an n-wire well-formed reversible circuit for some n. In what follows, a reversible circuit means a well-formed reversible circuit unless stated otherwise.

2.3 Complete Test Sets

Let C be an n-wire reversible circuit. Any $T \subseteq \{0,1\}^n$ is called a test set for C. T is said to be *complete* for C if T can detect all possible stuck-at faults on the wires in C. The problem to be considered in this paper is the following:

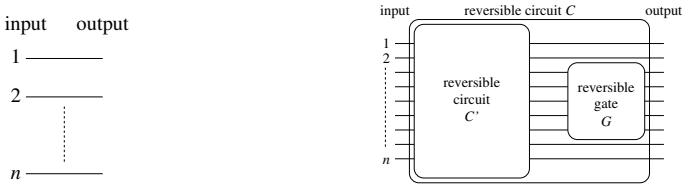


Fig. 1 Reversible Circuit

MIN-CTS
Instance: Reversible Circuit C
Question: Find a minimum test set T such that T is complete for C.

The following problem is a decision version of MIN-CTS.

CTS
Instance: Reversible Circuit C and positive integer k
Question: Is there a test set T with $|T| \leq k$ such that T is complete for C?

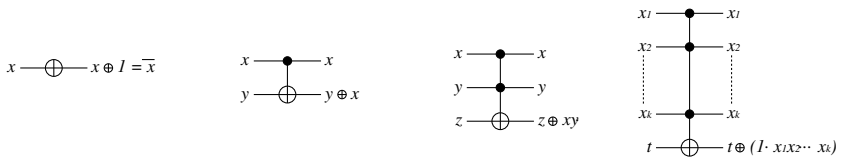
A wire w in C is *controllable* by T if $w(\boldsymbol{x}) = 0$ and $w(\boldsymbol{x}') = 1$ for some $\boldsymbol{x}, \boldsymbol{x}' \in T$, where $w(\boldsymbol{x})$ denotes the value of a wire w when $\boldsymbol{x}$ is given to C as an input. A set of wires in C, denoted by S, is controllable by T if every wire in S is controllable by T. C is controllable by T if every wire in C is controllable by T. The following theorem is shown in [7]:

Theorem I [7] T is complete for C if and only if C is controllable by T. □

2.4 CNOT Gates and CNOT Circuits

One of most known reversible gates is a CNOT gate. For any non-negative integer k, a *k-CNOT gate* is a $(k+1) \times (k+1)$ reversible gate such that if $(x_1, x_2, \ldots, x_k, t) \in \{0,1\}^{k+1}$ is given as an input then the gate outputs $(x_1, x_2, \ldots, x_k, t \oplus (1 \cdot x_1 x_2 \cdots x_k))$, where $\oplus$ is the exclusive OR operation. Each of $x_1, x_2, \ldots, x_k$ is called a control bit of the gate, and t is the target bit. Fig. 2

shows 0-CNOT, 1-CNOT, 2-CNOT, and k-CNOT gates. In this figure, the input[output] wires of the gate is the left[right] ones. A CNOT gate is a k-CNOT gate for some k, and denoted by Cnot.



(a) (b) (c) (d)

Fig. 2 (a) 0-CNOT gate, (b) 1-CNOT gate, (c) 2-CNOT gate, and (d)k-CNOT gate

A CNOT circuit is a reversible circuit consisting of only CNOT gates. A k-CNOT circuit is a CNOT circuit consisting of only k'-CNOT gates with $k' \leq k$. A 2-CNOT circuit is also called a NCT circuit. Fig. 3 illustrates a 4-wire 3-CNOT circuit, denoted by Cnot$[;2]$, Cnot$[1,2,3;4]$, Cnot$[4;2]$, Cnot$[3;1]$, Cnot$[1,4;3]$.

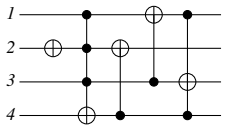


Fig. 3 Example of CNOT Circuit

3. Minimum Test Sets for 1-target CNOT Circuits

A 1-target CNOT circuit is a CNOT circuit in which the target bit of every k-CNOT gate with $k \neq 0$ is on the same wire in the circuit. The wire is called the target bit of the 1-target CNOT circuit. The other wires are called the control bits of the circuit. For example, see Fig. 4. Note that any combinatorial circuit can be expressed with a 1-target CNOT circuit because of the expression of Reed Muller.

Tayu *et al.* [9,10] proved the following theorem:

Theorem II [9,10] CTS is NP-complete even for 1-target 6-CNOT circuits.

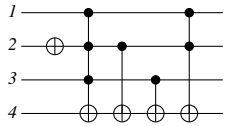


Fig. 4 Example of 1-target CNOT Circuit

□

This paper shows that CTS is NP-complete even for 1-target 3-CNOT circuits. The proof is by the relation between CTS for 1-target CNOT circuits and Not-all-Equal SAT, which is described as follows.

Let U be the set of Boolean variables u_i $(i \in \{1,2,\ldots,n\})$. For any $i \in \{1,2,\ldots,n\}$, u_i and $\overline{u_i}$ are called the positive and negative literals of u_i, respectively. A clause is a set of literals, and a clause with k literals is called a k-clause.

A truth-assignment of U is a mapping $t : U \to \{0,1\}$. If $t(u_i) = 0[t(u_i) = 1]$ then $t(\overline{u_i}) = 1[t(\overline{u_i}) = 0]$. A literal x with $t(x) = 1[t(x) = 0]$ is called a true[false] literal.

Not-All-Equal SAT
Instance: Set U of Boolean variables, collection C of clauses over U
Question: Is there a truth assigment of U such that each clause in C has at least one true literal and at least false literal?

It is easy to see the following two lemmas.

Lemma 1 Let C be a CNOT circuit. If T is complete for C then $|T| \geq 2$.

Lemma 2 Let C be an n-wire 1-target CNOT circuit, and let $T = \{(0,0,\ldots,0),(0,\ldots,0,1),(1,1,\ldots,1)\}$. Then, T is complete for C. □

From Lemmas 1 and 2, it is important to decide whether a given CNOT circuit C has a comlete test set T with $|T| = 2$ or not, and to find such an test if any.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ be an input of a circuit C. If C is a 1-target CNOT circuit and w is the input wire of a control bit of a CNOT gate in C then $w(\boldsymbol{x})$ is either x_j or $\overline{x_j}$ for some j. For any i, let Cnot$_i$ be the i-th CNOT circuit, and let $w_1, w_2, \ldots, w_k$ be the input wires of the control bits of Cnot$_i$. Then, if

x_j is viewed as a Boolean variable for every $j \in \{1, 2, \ldots, n\}$ then $w_l(\boldsymbol{x})$ can be viewed as a literal of some variable x_{j_l} for any $l \in \{1, 2, \ldots, k\}$. Let L_i be the set of such literals for any $i \in \{1, 2, \ldots, g\}$, where g is the number of CNOT gates in C. In Fig. 4,

$$L_1 = \emptyset, \quad L_2 = \{x_1, \overline{x_2}, x_3\}, \quad L_3 = \{\overline{x_2}\}, \quad L_4 = \{x_3\}, \quad L_5 = \{x_1, \overline{x_2}\}.$$

Lemma 3 Let C be an n-wire 1-target CNOT circuit. For any $T \subseteq \{0,1\}^n$ with $|T| = 2$, T is complete for C if and only if the following two conditions are satisfied: (1) $T = \{\boldsymbol{\alpha}, \bar{\boldsymbol{\alpha}}\}$ for some $\boldsymbol{\alpha} \in \{0,1\}^n$, and (2) given $\boldsymbol{\alpha}$ as an input to C, for any $i \in \{1, 2, \ldots, g\}$, L_i has at least one true literal and false one if $L_i \neq \emptyset$.

Proof: (Sufficiency) By (1), every wire of each control bit in G is controlable. Since, by (2), the control bits of each k-CNOT gate with $k \geq 1$ contain input 0 in either case of inputs, we conclude by (1) that every wire of the target bit in G is controlable. Hence, T is complete for G.

(Necessity) The proof is by contradiction. If (1) does not hold then the input wire of some bit is not controrable, which is a contradiction. So, assume that (2) does not hold, that is L_i has either all true literals or false ones in L_i under input $\boldsymbol{\alpha}$ for some $i \in \{1, 2, \ldots, n\}$. If the input wire of the target bit of the i-th CNOT gate is controlable then the inputs of the CNOT gate are either

(1) $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$ or (2) $(0, \ldots, 0, 1)$ and $(1, \ldots, 1, 0)$.

So, the output wire of the target bit of the i-th CNOT gate is not controlable in either case, and hence T is not complete for G, which is a contradiction. □

Lemma 3 shows the relation between CTS and Not-All-Equal SAT. When an instance of CTS is restricted to 1-target CNOT circuits, it is trivial to give a polynomial-time reduction from CTS to Not-All-Equal SAT by Lemma 3. It is also easy to give a polynomial-time reduction from Not-All-Equal SAT to CTS. For example, let us be given a collection of clauses

$$\{\{u_1, \overline{u_2}, \overline{u_3}\}, \{\overline{u_1}, u_4\}, \{u_2, \overline{u_4}, u_5\}\}$$

over $U = \{u_1, u_2, \ldots, u_5\}$. Then, we transform each clause into a CNOT subcircuit as follows:

$$\{u_1, \overline{u_2}, \overline{u_3}\} \rightarrow \mathrm{Cnot}[;2], \mathrm{Cnot}[;3], \mathrm{Cnot}[1,2,3;6], \mathrm{Cnot}[;3], \mathrm{Cnot}[;2]$$
$$\{\overline{u_1}, u_4\} \rightarrow \mathrm{Cnot}[;1], \mathrm{Cnot}[1,4;6], \mathrm{Cnot}[;1]$$
$$\{u_2, \overline{u_4}, u_5\} \rightarrow \mathrm{Cnot}[;4], \mathrm{Cnot}[2,4,5;6], \mathrm{Cnot}[;4].$$

An instance of CTS is obtained by concatenating the subcircuits (See Fig. 5). This transformation is a polynomial-time reduction from Not-All-Equal SAT to CTS.

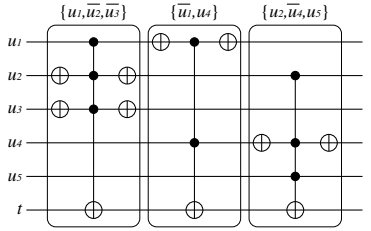


Fig. 5 Reduction from Not-All-Equal SAT to CTS

The following theorems are proved on Not-All-Equal SAT (For example, see [3]).

Theorem III Not-All-Equal SAT is NP-complete even if every clause is 3-clause. □

Theorem IV Not-All-Equal SAT is solvable in polynomial time if every clause is 2- or 1-clause. In addition, we have a polynomial time algorithm for finding a truth assigment of the boolean variables such that each clause has at least one true literal and at least false literal. □

These theorems present the following theorems:

Theorem 1 CTS is NP-complete even for 1-target 3-CNOT circuits. □

Theorem 2 CTS, and CTS-MIN, are solvable in polynomial time for 1-target 2-CNOT circuits. □

4. NP-completeness of CTS for NCT Circuits

In the previous section, CTS is proved to be NP-complete even for 1-target 3-CNOT circuits, and so even for (general) 3-CNOT circuits. In this section, it is proved that CTS is NP-complete even for NCT circuits, that is 2-CNOT circuits, by giving a polynomial-time reduction from 3-Colorability.

Let H be a graph, and let f be a mapping from $V(H)$ to $\mathbb{Z}^+$, where $\mathbb{Z}^+$ is the set of positive integers. f is called a coloring of H if $f(u) \neq f(v)$ for every edge

$e = (u, v) \in E(H)$. A k-coloring of H is a coloring of H with $\max f(V(H)) \leq k$.

3-COLORABILITY
Instance: Connected graph H
Question: Is there a 3-coloring of H?

3-COLORABILITY is a well-known NP-complete problem (For example, see [3]).

Theorem V 3-COLORABILITY is NP-complete. □

4.1 Preliminaries

Let $G_1[1,2] = \mathrm{Cnot}[1;2], \mathrm{Cnot}[1;2]$, as in Fig. 6. It is easy to see the following lemma:

Lemma 4 $T \subseteq \{0,1\}^2$ is complete for G_1 if and only if $|T| \geq 3$. □

Let $G_2[1,2,3] = \mathrm{Cnot}[1,2;3], \mathrm{Cnot}[3;1], \mathrm{Cnot}[3;2], \mathrm{Cnot}[3;2], \mathrm{Cnot}[3;1], \mathrm{Cnot}[1,2;3]$, as in Fig. 7.

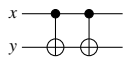


Fig. 6 Circuit G_1

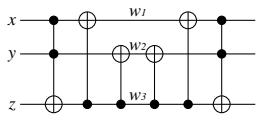


Fig. 7 Circuit G_2

Lemma 5 Let T be a subset of $\{0,1\}^3$ with $|T| = 3$ and $|T'| = 3$, where $T' = \{(x,y) : (x,y,z) \in T\}$. If T is complete for G_2 then $(0,0) \in T'$.

Proof: Assume that $(0,0) \notin T'$, that is $T' = \{(0,1),(1,0),(1,1)\}$. Then, $T = \{(0,1,z_1),(1,0,z_2),(1,1,z_3)\}$ for some $z_1, z_2, z_3 \in \{0,1\}$. Then, the values of wires w_1, w_2, and w_3 in Fig. 7 are as in Table 1 Therefore, if these three wires

Table 1 Values of w_1, w_2, w_3 in G_2

inputs			wires		
x	y	z	w_1	w_2	w_3
0	1	z_1	z_1	$\overline{z_1}$	z_1
1	0	z_2	$\overline{z_2}$	z_2	z_2
1	1	z_3	z_3	z_3	$\overline{z_3}$

are controlable then we conclude that $z_1 = z_2 = z_3$, which is a contradiction to the fact that the third input wire is controlable. Then $(0,0) \in T'$. □

Lemma 6 Let $T' = \{(0,0),(x_2,y_2),(x_3,y_3)\}$ be a subset of $\{0,1\}^2$ with $|T'| = 3$. Then, there exist $z_1, z_2, z_3 \in \{0,1\}$ such that $T = \{(0,0,z_1), (x_2,y_2,z_2),(x_3,y_3,z_3)\}$ is complete for G_2.

Proof: It follows the lemma that the following three sets T_1, T_2, T_3 are complete for G_2:

- $T_1 = \{(0,0,0),(0,1,1),(1,0,1)\}$;
- $T_2 = \{(0,0,0),(0,1,1),(1,1,1)\}$;
- $T_3 = \{(0,0,0),(1,0,1),(1,1,1)\}$.

□

Let $G_3[1,2,3,4,5] = G_1[1,2], G_2[1,2,3], \mathrm{Cnot}[;2], G_2[1,2,4], \mathrm{Cnot}[;2], \mathrm{Cnot}[;3], G_2[1,2,5], \mathrm{Cnot}[;3]$, as in Fig. 8. From Lemmas 4, 5, and 6, we have the following two lemmas.

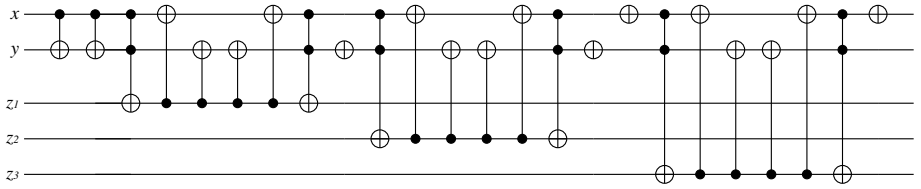


Fig. 8 Circuit G_3

Lemma 7 Let T be a subset of $\{0,1\}^5$ with $|T| = 3$, and let $T' = \{(x,y) : (x,y,z_1,z_2,z_3) \in T\}$. If T is complete for G_3 then $T' = \{(0,0),(0,1),(1,0)\}$.

Lemma 8 There exist (z_{11},z_{12},z_{13}), (z_{21},z_{22},z_{23}), and $(z_{31},z_{32},z_{33}) \in \{0,1\}^3$ such that $T = \{(0,0,z_{11},z_{12},z_{13}),(0,1,z_{21},z_{22},z_{23}),(1,0,z_{31},z_{32},z_{33})\}$ is complete for G_3.

4.2 Tranformation

Let H be a connected graph with $V(H) = \{1,2,\ldots,n\}$. Let $e_j = (u_j, v_j)$ denote the j-th edge in H for any $j \in \{1,2,\ldots,m\}$, where m is the number of

edges in H. Then, G_H is the $(n+3m)$-wire NCT circuit defined as

$$G_H = G_3[u_1, v_1, a_{11}, a_{12}, a_{13}], G_3[u_2, v_2, a_{21}, a_{22}, a_{23}], \ldots, G_3[u_m, v_m, a_{m1}, a_{m2}, a_{m3}],$$

where $a_{jk} = n+3(j-1)+k$ for any $j \in \{1, 2, \ldots, m\}$ and $k \in \{1, 2, 3\}$. It is easy to see the following lemma.

Lemma 9 G_H can be constructed from H in a polynomial time. □

Lemma 10 If H is 3-colorable then there exists some T with $|T| = 3$ such that T is complete for G_H.

Proof: Let f be a 3-coloring of H. Then,

$$T = \{(x_{11}, x_{12}, \ldots, x_{1(n+3m)}), (x_{21}, x_{22}, \ldots, x_{2(n+3m)}), (x_{31}, x_{32}, \ldots, x_{3(n+3m)})\}$$

is defined as follows. For any $i \in \{1, 2, 3\}$ and $v \in \{1, 2, \ldots, n\}$, let

$$x_{iv} = \begin{cases} 1 & \text{if } f(v) = i, \\ 0 & \text{if } f(v) \neq i \end{cases}$$

Fix any $j \in \{1, 2, \ldots, m\}$. Let

$$T_j = \{(x_{1u_j}, x_{1v_j}, x_{1a_{j1}}, x_{1a_{j2}}, x_{1a_{j3}}), (x_{2u_j}, x_{2v_j}, x_{2a_{j1}}, x_{2a_{j2}}, x_{2a_{j3}}), (x_{3u_j}, x_{3v_j}, x_{3a_{j1}}, x_{3a_{j2}}, x_{3a_{j3}})\},$$

and let

$$T'_j = \{(x_{1u_j}, x_{1v_j}), (x_{2u_j}, x_{2v_j}), (x_{3u_j}, x_{3v_j})\}.$$

Consider the j-th edge $e_j = (u_j, v_j)$, Since $f(u_j) \neq f(v_j)$, we obtain that $T'_j = \{(0,0), (0,1), (1,0)\}$. As proved in Lemma 8, we can define $(x_{1a_{j1}}, x_{1a_{j2}}, x_{1a_{j3}}), (x_{2a_{j1}}, x_{2a_{j2}}, x_{2a_{j3}}), (x_{3a_{j1}}, x_{3a_{j2}}, x_{3a_{j3}}) \in \{0,1\}^3$ so that T_j is complete for $G_3[u_j, v_j, a_{j1}, a_{j2}, a_{j3}]$. Hence, there exists some T with $|T| = 3$ such that T is complete for G_H. □

Lemma 11 If there exists some T with $|T| = 3$ such that T is complete for G_H then H is 3-colorable.

Proof: Let

$$T = \{(x_{11}, x_{12}, \ldots, x_{1(n+3m)}), (x_{21}, x_{22}, \ldots, x_{2(n+3m)}), (x_{31}, x_{32}, \ldots, x_{3(n+3m)})\}$$

Since H is connected and T is complete for G_H, we conclude by Lemma 7 that exactly two of x_{1v}, x_{2v}, x_{3v} is 0 and the other is 1 for every $v \in \{1, 2, \ldots, n\}$. So, we define $f(v) = i$ for any $v \in \{1, 2, \ldots, n\}$, where $x_{iv} = 1$. Consider the j-th edge $e_j = (u_j, v_j)$ in H for any $i \in \{1, 2, \ldots, m\}$. Let

$$T_j = \{(x_{1u_j}, x_{1v_j}, x_{1a_{j1}}, x_{1a_{j2}}, x_{1a_{j3}}), (x_{2u_j}, x_{2v_j}, x_{2a_{j1}}, x_{2a_{j2}}, x_{2a_{j3}}), (x_{3u_j}, x_{3v_j}, x_{3a_{j1}}, x_{3a_{j2}}, x_{3a_{j3}})\},$$

and let

$$T'_j = \{(x_{1u_j}, x_{1v_j}), (x_{2u_j}, x_{2v_j}), (x_{3u_j}, x_{3v_j})\}.$$

Since T is complete for G_H, T_j is complete for $G_3[u_j, v_j, a_{j1}, a_{j2}, a_{j3}]$, so $T'_j = \{(0,0), (0,1), (1,0)\}$ by Lemma 7, which implies that $f(u_j) \neq f(v_j)$. Hence, we conclude that f is a 3-coloring of H. □

From Theorem V and Lemmas 9, 10, and 11, we have the following theorem.

Theorem 3 CTS is NP-complete even for NCT circuits.

5. Randomized Algorithm for Smaller Test Sets of CNOT Circuits

This section gives a randomized algorithm for finding a complete test set for CNOT circuits. This algorithm can be easily extended to one for reversible circuits, and the results in the section can be also generalized for reversible circuits.

By Theorem 3 in the previous section, it seems to be quite difficult, or even impossible, to design a polynomial time algorithm for finding a minimum test set for a given CNOT circuit, so we propose an approximation algorithm. Let w be any wire in an n-wire CNOT circuit C, let $\boldsymbol{x} \in \{0,1\}^n$ be an input of C. If $w(\boldsymbol{x}) = b$ for some $b \in \{0,1\}$ then $\boldsymbol{x}$ is said to cover (w, b). For any $S \subseteq \{0,1\}^n$, wire w in C, and $b \in \{0,1\}$, S covers (w, b) if some $\boldsymbol{x} \in S$ covers (w, b). Our algorithm is based on the following fact shown in [7]:

Fact 1 [7] Let C be an n-wire CNOT circuit. For each wire w in C and value $b \in \{0,1\}$, there exist exactly 2^{n-1} inputs each of which covers (w, b). □

The following key lemma is obtained from the fact:

Lemma 12 Let C be an n-wire CNOT circuit, and $S \subset \{0,1\}^n$. Then, there exist at least 2^{n-2} inputs $\boldsymbol{x} \in \{0,1\}^n$ such that $\boldsymbol{x}$ covers at least $1/3$ of the pairs (w, b) not covered by S.

Proof: Assume for that the lemma does not hold. Let N be the number of inputs $\boldsymbol{x}$ such that $\boldsymbol{x}$ covers at least $1/3$ of pairs (w, b) not covered by S. Then, $N < 2^{n-2}$. From Fact 1, the following inequality must hold:

$$m2^{n-1} < mN + \frac{1}{3}m(2^n - N) = \frac{1}{3}m(2^n + 2N), \quad (1)$$

where m is the number of pairs not covered by S. However, since $4N < 2^{n-2}$, the right-hand side is less than $m2^{n-1}$, which is a contradiction. Hence, the lemma must hold. □

Lemma 12 presents a simple randomized algorithm, as shown in Fig. 9, for finding a test set of any given CNOT circuit.

Algorithm Random(C)

Input: n-wire CNOT circuit C;

Step 0: $S \leftarrow \emptyset$;
Step 1: If S covers all pairs of wires in C and binary values then output S and halt;
Step 2: Select $\boldsymbol{x} \in \{0,1\}^n$ randomly until $\boldsymbol{x}$ satisfies the condition in Lemma 12;
Step 3: $S \leftarrow S \cup \{\boldsymbol{x}\}$, and goto Step 1;

Fig. 9 Randomized Algorithm for Finding a Test Set of a CNOT circuit

5.1 Theoretical Analysis

Theorem 4 Given an n-wire CNOT circuit C, let S_C be the output of Algorithm Random in Fig. 9. Then,

$$|S_C| \leq O(\log(n+g)),$$

where g is the number of gates in C.

Theorem 5 For an n-wire CNOT circuit with g CNOT gates, Algorithm Random in Fig. 9 runs in $O((n+kg)\log(n+g))$ expected time, where k is the maximum number of control bits over all CNOT gates in the circuit.

Proof: By Lemma 12, the expected number of times $\boldsymbol{x}$ is selected is at most 4 in Step 2 for each S. It takes $O(n+kg)$ time to check whether $\boldsymbol{x}$ covers at least 1/3 of the pairs not covered by S. So, each repetition runs in $O(n+kg)$ expected time. By Theorem 4 and the linearity of expectation, Algorithm Random in Fig. 9 runs in $O((n+kg)\log(n+g))$ expected time. □

5.2 Experimental Results

We have implemented the proposed algorithm and the algorithm by Patel *et al.* [7] in C programming language. All the experiments are performed on a PC machine with Intel Core 2 Duo SU9300 processor and 2GB memory, whose OS is Ubuntu Linux 9.04. GLPK 4.29 is used as a C library to solve integer programming in the algorithm by Patel *et al.* [7].

In the experiments, we generated randomly at least 50 k-wire NCT circuits with g gates for each $k \in \{16, 64\}$ and $g \in \{100, 1000, 10000, 100000\}$. Table 2 shows the average results on executed time and the size of test sets obtained, where "—" means that the results cannot be obtained for lack of memory. In this table, RA and IP represents the algorithms proposed by the paper and Patel *et al.* [7], respectively, and "+compact." means that the compaction operation proposed by Patel *et al.* [7] is performed as post-processing.

As Table 2 shows, RA and RA+compact. are much faster than IP and IP+compact. Furthermore, RA requires less memory space than the other algorithms. In fact, RA can run even for 1024-wire CNOT circuits with 1 million gates while the other cannot for 16-wire CNOT circuits with 100,000 gates. On the other hand, the size of the test set obtained by RA(+compact.) is almost same as that by IP(+compact.). Compact operation can be used to reduce the size of obtained test set obtained by RA or IP. However, compaction operation needs much more memory space, and hence cannot be used for large scale reversible circuits.

6. Conclusion

This paper considered the problem of generating a minimum complete test set for stuck-at faults in a given reversible circuit. First of all, the paper proved that the problem is NP-hard even for 3-CNOT circuits, and that any 1-target CNOT circuit has a complete test set with size 3. Next, in this paper, the problem is shown to be NP-hard even for NCT circuits, that is, circuits consisting of only k-CNOT gates with $k \leq 2$. Finally, the paper proposed a simple randomized algorithm for this problem, and showed the performance of the algorithm both by theoretical analysis and by experiments.

The following two problems are still open:

Table 2 Comparison of Algorithms RA(proposed) and IP(by Patel *et al.* [7])
(a) 16-Wire NCT Circuits

#gates	100		1000		10000		100000	
	#Tests	Time[s]	#Tests	Time[s]	#Tests	Time[s]	#Tests	Time[s]
RA	7.1	8.8×10^{-5}	10.2	6.6×10^{-4}	13.5	8.1×10^{-3}	16.8	9.8×10^{-2}
RA+Compact.	5.9	8.9×10^{-4}	8.5	8.0×10^{-3}	11.6	1.4×10^{-1}	—	—
IP	6.2	9.8	10.4	9.9	13.4	13.0	—	—
IP+Compact.	5.6	10.7	8.5	10.9	11.6	13.3	—	—

(b) 64-Wire NCT Circuits

#gates	100		1000		10000		100000	
	#Tests	Time[s]	#Tests	Time[s]	#Tests	Time[s]	#Tests	Time[s]
RA	7.8	8.4×10^{-5}	10.3	6.8×10^{-4}	13.6	8.2×10^{-3}	16.9	9.7×10^{-2}
RA+Compact.	6.4	1.2×10^{-3}	8.7	9.9×10^{-3}	11.6	2.3×10^{-1}	—	—
IP	5.5	18.4	9.9	19.8	13.3	22.8	—	—
IP+Compact.	4.8	18.5	8.6	21.1	11.7	27.5	—	—

(1) The algorithm proposed in the paper guarantees an upper bound on the size of the obtained test set. However, the approximation ratio of the algorithm is $O(\log(n+g))$, where n and g are the numbers of wires and gates in the circuit, respectively. Does there exist a polynomial-time constant-approximable algorithm for this problem?

(2) Does there exists a polynomial-time algorithm to generate a minimum complete test set for stuck-at faults when a given reversible circuit consists of only 0-CNOT and 1-CNOT circuits?

References

1) C.Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol.17, pp.525–532, 1971.
2) E.Fredkin and T.Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol.21, pp.219–253, 1982.
3) M.Garey and D.Johnson, *Computers and Intractability: A Guide to the Teory of NP-Completeness*. W. H. Freeman and Company, 1979.
4) R.Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol.3, pp.183–191, 1961.
5) R.Merkle, "Two types of mechanical reversible logic," *Nanotechnology*, vol.4, pp. 114–131, Feb. 1993.
6) M.Nielsen and I.Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
7) K. Patel, J. Hayes, and I. Markov, "Fault testing for reversible circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.23, no.8, pp.1220–1230, 2004.
8) V. Shende, A. Prasad, I. Markov, and J. Hayes, "Synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.22, pp.710–722, June 2003.
9) S.Tayu, S.Ito, and S.Ueno, "On the fault testing for reversible circuits," *Lecture Notes in Computer Science*, vol.4835, pp.812–821, 2007.
10) S.Tayu, S.Ito, and S.Ueno, "On fault testing for reversible circuits," *IEICE Transactions on Information and Systems*, vol.E91-D, no.12, pp.2770–2775, 2008.