

Fault Testing for Reversible Circuits*

Ketan N. Patel, John P. Hayes and Igor L. Markov

University of Michigan, Ann Arbor 48109-2122

{knpatel, jhayes, imarkov}@eecs.umich.edu

Abstract

Applications of reversible circuits can be found in the fields of low-power computation, cryptography, communications, digital signal processing, and the emerging field of quantum computation. Furthermore, prototype circuits for low-power applications are already being fabricated in CMOS. Regardless of the eventual technology adopted, testing is sure to be an important component in any robust implementation.

We consider the test set generation problem. Reversibility affects the testing problem in fundamental ways, making it significantly simpler than for the irreversible case. For example, we show that any test set that detects all single stuck-at faults in a reversible circuit also detects all multiple stuck-at faults. We present efficient test set constructions for the standard stuck-at fault model as well as the usually intractable cell-fault model. We also give a practical test set generation algorithm, based on an integer linear programming formulation, that yields test sets approximately half the size of those produced by conventional ATPG.

*This work was supported by the DARPA QuIST program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies of endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA) or the U.S. Government. A preliminary version of this paper was presented at the VLSI Test Symposium, Napa, CA in April 2003.

1 Introduction

A primary motivation for the study of reversible circuits is the possibility of nearly energy-free computation. Landauer [18] showed that traditional irreversible circuits necessarily dissipate energy due to the erasure of information. On the other hand, in principle, reversible computation can be performed with arbitrarily small energy dissipation [2, 11]. Though the fraction of the power consumption in current VLSI circuits attributable to information loss is negligible, this is expected to change as increasing packing densities force the power consumption per gate operation to decrease [24], making reversible computation an attractive alternative.

Many applications in the fields of cryptography, communications and digital signal processing require computations that transform the data without erasing any of the original information. These applications are particularly well-suited to a reversible circuit implementation. However, the applicability of reversible circuits is not limited to inherently reversible applications. Conventional irreversible computation can be implemented reversibly using limited overhead [19, 5]. Furthermore, reversible circuits are not just a theoretical area of study: DeVos et al. [9] have built reversible CMOS circuits that are powered entirely from the input pins without the assistance of additional power supplies.

A major new motivation for the study of reversible circuits is provided by the emerging field of quantum computation [21]. In a quantum circuit the operations are performed on quantum states or qubits rather than bits. Since quantum evolution is inherently reversible, the resulting quantum computation is as well. Classical reversible circuits form an important subclass of these quantum circuits.

While logic synthesis, and hardware implementations for reversible circuits have been studied in previous work, very little research has considered reversibility in the context of testing. One exception is research at Montpellier, where reversibility was used to synthesize on-line test structures for irreversible circuits [3, 4]. In contrast, our focus is on testing inherently reversible circuits, particularly, generating efficient test sets for these circuits. Though this is

a hard problem for conventional irreversible circuits, it can be significantly simplified in our case. Agrawal [1] has shown that fault detection probability is greatest when the information output of a circuit is maximized. This suggests that it may be easier to detect faults in reversible circuits, which are information lossless, than in irreversible ones. While this previous work focused on probabilistic testing, here we are concerned with complete deterministic testing. We show that surprisingly few test vectors are necessary to fully test a reversible circuit under the multiple stuck-at fault model, with the number growing at most logarithmically both in the number of inputs and the number of gates. This provides additional motivation for studying reversible circuits, namely they may be much easier to test than their irreversible counterparts.

In Section 2 we give some basic background on reversible circuits. We then present some theoretical results on complete test sets for reversible circuits in Section 3, and results on such sets for worst-case circuits in Section 4. In Section 5 we give a practical algorithm for generating efficient complete test sets, present simulation results, and make comparisons to conventional ATPG (automatic test pattern generation). We extend our results to the more general cell-fault model in Section 6, and conclude in Section 7.

2 Reversible Circuits

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 numbers of input and output wires are equal. If it has k inputs (and outputs), we call it a *reversible $k \times k$ gate*. Three commonly used gates, composing the *NCT*-gate library, are shown in Figure 1. The NOT gate inverts the input, the C-NOT gate passes the first input through and inverts the second if the first is 1, and the Toffoli gate passes the first two inputs through and inverts the third if the first two are both 1.

A *well-formed reversible circuit* is constructed by starting with n wires, forming the basic circuit, and iteratively concatenating reversible gates to some subset of the output wires of the previous circuit. The outputs of each reversible gate replace the wires at its input. This iterative

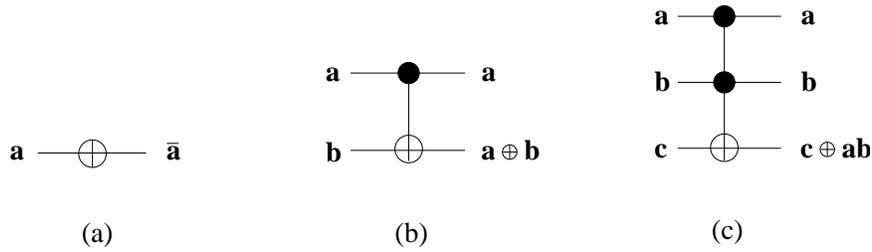


Figure 1: Examples of reversible logic gates: (a) NOT, (b) C-NOT, and (c) Toffoli.

construction naturally gives us the notion of *levels* in the circuit; the inputs to the circuit are at level 0, and the outputs of any gate are at one plus the highest level of any of its inputs. For convenience in cases where a wire at the input of a gate is at level i and the outputs are at level $j > i + 1$, we say the input is at all levels between i and $j - 1$ inclusively. This gives us n wires at each level. Figure 2 shows an example of a reversible circuit with the levels denoted by dotted lines. The propagation of an input vector through the circuit is shown to illustrate the circuit's operation. The *depth* d of the circuit is the maximum level, which can be no larger than the number of gates in the circuit. We will often find it convenient to use an n -bit vector to refer to the values of the wires at a given level in the circuit. A binary vector has *weight* k if it contains exactly k 1's, and we denote the all-0's and all-1's vectors by $\underline{0}$ and $\underline{1}$, respectively.

The foregoing iterative construction also leads to the notion of a *sub-circuit*, the part of the original circuit between levels i and j , or more specifically, the circuit formed by the gates with outputs at level greater than i and less than $j + 1$. We denote the function computed by the sub-circuit as $f_{i,j}$ and its inverse as $f_{i,j}^{-1}$. If we omit the first subscript i it should be assumed to be 0. The function of the entire circuit is then f_d .

We say a reversible circuit is *L-constructible*, if it can be formed using the L -gate library. Some important gate libraries used here are the *NCT*-gate library mentioned above, the C-NOT gate library consisting of only C-NOT gates, and the universal or *U*-gate library which consists of all possible reversible $n \times n$ gates. These three gate libraries compute the set of even permutations, the set of linear reversible functions, and the set of all permutations respectively [22].

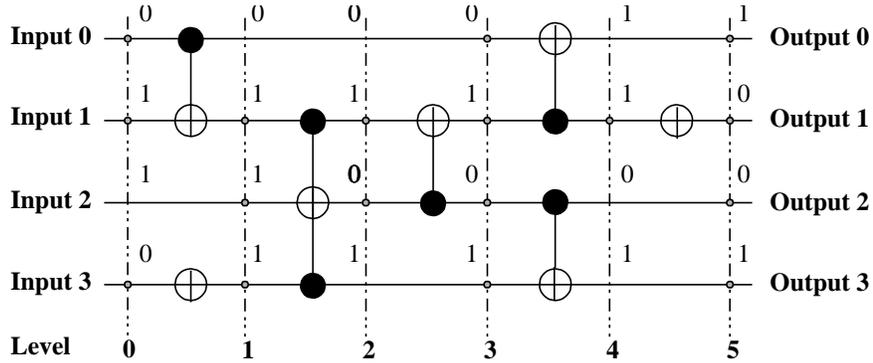


Figure 2: Reversible circuit example. The dotted lines represent the levels in the circuit, and the small open dots represent possible stuck-at fault sites. The diagram also shows the propagation of input vector 0110 through the circuit.

In order to compute any function that is not an even permutation, at least one gate that spans all n wires is required. Unlike the U -library, practical gate libraries are unlikely to contain such large gates. The NCT -gate library has been well studied [23, 22] and computes essentially all functions that are practically realizable. Consequently, we will focus on it for most of our work.

3 Complete Test Sets

Given a reversible circuit C and a fault set F , we want to generate a set of test vectors that detect all faults in F . We call such a test set *complete*. A complete test set with the fewest possible vectors is *minimal*.

Two important properties of reversibility simplify the test set generation problem. The first is *controllability*: there is a test vector that generates any given desired state on the wires at any given level. The second is *observability*: any single fault that changes an intermediate state in the circuit also changes the output. Neither property holds, in general, for irreversible circuits.

To illustrate these two properties, consider the reversible circuit shown in Figure 2. The controllability property enables us to set the wires at any level in the circuit to any desired set of values using a unique input vector, found by reversing the action of the circuit. For

example, to find the input vector necessary to set the wires at level 2 to the vector 0101, we first backtrack through the three-input gate between levels 1 and 2. This gives us the vector 0111 at level 1. Backtracking once more gives the vector 0110 at the input. Reversibility guarantees that this backtracking is always possible and always yields a unique vector at the input. The observability property enables us to observe any intermediate change in the circuit at the output. This is because each vector at any level in the circuit corresponds to exactly one output vector. For example, only the vector 0101 at level 2 in the circuit yields the output vector 1001; any other vector at this level will yield a different output.

For most of this paper we adopt the standard stuck-at fault model used in testing conventional circuits, which includes all faults that fix the values of wires in the circuit to either 0 or 1. For reversible circuits we show that any test set that detects all single stuck-at faults, also detects any number of simultaneous faults. In Section 6 we extend our results to the more general cell-fault model, where the fault set consists of single gate failures.

3.1 General Properties

The following proposition provides a simple necessary and sufficient condition for a test set to be complete for the stuck-at fault model.

Proposition 1 *Under the single stuck-at fault model a test set is complete if and only if each wire at every level can be set to both 0 and 1 by the test set.*

Proof Assume without loss of generality that a test set does not set a wire at level i to 0. A stuck-at 1 fault at this point in the circuit is then undetectable, since the outputs from the test set are unaffected. On the other hand, if all wires at every level can be set to both 0 and 1 by the test set, then a stuck-at fault must affect at least one test vector, changing the value of the wire at that level from a 0 to a 1 or vice versa. By the observability property this change will affect the output. \square

To illustrate this proposition consider the fault site on the second wire at level 4 in the circuit in Figure 2. In order to detect a stuck-at 0 fault, the test set must be able to set this wire to 1, otherwise the fault would not have any effect on the test set, and would therefore be undetected. If a stuck-at 0 fault does occur, then a test vector that sets the wire to 1 would generate an incorrect output, namely a 1 instead of a 0 on the second wire at the output. Similarly, to detect a stuck-at 1 fault on this wire, the test set must be able to set the wire to 0.

The next proposition shows that the single stuck-at and the multiple stuck-at fault models are essentially identical for reversible circuits; specifically, a test set that is complete for one model is also complete for the other. The intuition behind this property is that in the case of multiple faults the final fault(s), i.e., those closest to the outputs, can be detected by working backwards from the outputs.

Proposition 2 *Any test set that is complete for the single stuck-at fault model is also complete for the multiple stuck-at fault model.*

Proof Suppose we have a counter-example. Then there must be a complete test set T for some reversible circuit under the single fault model, which is not complete for multiple faults. So at least one multiple fault M is undetectable by T . Since M is undetectable, the response of the circuit to T must be the same as those of the fault-free circuit. Now M is composed of faults at various levels. Let i be the deepest level containing a sub-fault of M . Since no sub-faults occur at any level greater than i , the reversible sub-circuit between level i and the outputs is identical to the corresponding sub-circuit in the fault-free circuit. Therefore, since the response to T and the reversible sub-circuit between level i and the outputs are the same as for the fault-free circuit, the values of the wires at level i must also be the same as for the fault-free circuit. Since T is complete under the single fault model, by Proposition 1 each wire at level i must take both the value 0 and 1. However this is a contradiction, since there is at least one sub-fault at level i that fixes the value of a wire. \square

This correspondence between the single and multiple stuck-at fault models allows us to restrict our attention to the conceptually simpler case of single faults. If we have an n -wire circuit with l gates of sizes k_1, \dots, k_l , then a total of $2(n + \sum_{i=1}^l k_i)$ single stuck-at faults can occur: stuck-at 0 and stuck-at 1 faults for each gate input and circuit output. Reversibility then implies the following result, which will be useful later.

Lemma 1 *Each test vector covers exactly half of the possible faults, and each fault is covered by exactly half of the possible test vectors.*

Proof Each test vector t sets the bit at each fault site to either 0 or 1, detecting either a stuck-at 1 or stuck-at 0 fault, respectively. Therefore, t detects precisely half of the possible single stuck-at faults. For a given stuck-at fault there are 2^{n-1} possible bit vectors at that level that can detect the fault, namely those that set the faulty bit to the opposite of the stuck-at value. Since the circuit is reversible, each of these can be traced back to a distinct input vector. Therefore, half of the 2^n input vectors detect the fault. \square

We can obtain some properties of a minimal test set of a circuit by decomposing the circuit into sub-circuits. For example, the size of a minimal test set for a reversible circuit is greater than or equal to that of any of its sub-circuits. On the other hand, the size of a minimal test set for a circuit formed by concatenating reversible circuits C_1, \dots, C_k is no greater than the sum of the sizes of minimal test sets for the individual C_i 's. Finally if two reversible circuits C_1 and C_2 , with minimal test sets of sizes $|T_1|$ and $|T_2|$ respectively, act on a disjoint set of input/output bits, then the size of the minimal test set of the circuit formed by concatenating C_1 and C_2 is equal to $\max\{|T_1|, |T_2|\}$. These properties can be used to bound the size of the minimal test set, and in some cases, to simplify the problem of finding a minimal test set.

3.2 Test Set Construction

The following proposition gives a number of complete test set constructions, implicitly providing upper bounds on the size of a minimal test set.

Proposition 3 *A complete test set for an n -wire reversible circuit with depth d and a total of l gates with sizes k_1, \dots, k_l is given by:*

- a. *any $2^{n-1} + 1$ distinct test vectors*
- b. *the following $d + 2$ test vectors*

$$\left\{ \underline{0}, \underline{1}, f_1^{-1}(\overline{f_1(\underline{0})}), \dots, f_d^{-1}(\overline{f_d(\underline{0})}) \right\} \quad (1)$$

- c. *some set of $\lfloor \log_2(n + \sum_{i=1}^l k_i) \rfloor + 2$ test vectors.*

Proof

(a) The value of a wire at a given level is set to 0 (or 1) by exactly 2^{n-1} input vectors. Therefore, if the test set contains $2^{n-1} + 1$ vectors, then at least one will set it to 1 (or 0). Since this is true for all fault sites, by Proposition 1 the test set is complete.

(b) The vector $f_i^{-1}(\overline{f_i(\underline{0})})$ sets the wires at level i to the bitwise inverse of the values set by the $\underline{0}$ vector. Therefore each wire at every level can be set to both 0 and 1 by the test set. By Proposition 1 the test set is complete.

(c) To prove this part we first prove that given a reversible circuit and an incomplete set of test vectors, there is a test vector that can be added that covers at least half of the remaining faults.

Let m be the number of test vectors given, F_C be the faults covered by this set, and C the size of F_C . If none of the remaining $2^n - m$ input vectors cover at least half of the remaining faults, then they must each cover more than half of the faults in F_C . By Lemma 1 every test vector covers exactly $n + \sum_{i=1}^l k_i$ faults and every fault is covered by exactly 2^{n-1} test vectors. Therefore, the number of times faults in F_C are covered by all input vectors cumulatively is $2^{n-1} \cdot C$, implying the following inequalities:

$$(2^n - m) \left(\frac{C}{2} \right) < 2^{n-1} \cdot C - m \left(n + \sum_{i=1}^l k_i \right) \quad (2)$$

$$2 \left(n + \sum_{i=1}^l k_i \right) < C \quad (3)$$

The second inequality is false since the number of faults covered cannot be larger than the total number of faults that can occur. Therefore we have a contradiction, and there must be a test vector that can be added to cover at least half of the remaining faults.

Recursively applying this observation we can eliminate all uncovered faults in no more than

$$\left\lceil \log_2 \left(n + \sum_{i=1}^l k_i \right) \right\rceil + 2 \quad (4)$$

steps (test vectors). \square

Proposition 3 limits the size of the minimal test set based on the size of the reversible circuit both in terms of its depth and the number of input/output bits. For the circuit in Figure 2, parts a-c of the proposition give upper bounds of 9, 7, and 6 test vectors, respectively. The final part of the proposition implies that a reversible circuit can be tested by a very small set of tests. As an example, a reversible circuit on 64 wires with a million 3×3 gates can be tested using no more than 23 input vectors. However, while the first two parts of Proposition 3 give practical constructions, the last one does not; consequently, it may not be easy to find such a test set.

4 (L,n)-Complete Test Sets

We say a test set is (L,n) -complete for gate library L acting on n wires, if it is complete for all circuits formed by the library. The following proposition shows that a circuit requiring such a test set exists for any gate library.

Proposition 4 *Any reversible gate library L acting on n wires has an (L,n) -complete set of test vectors that is minimal for some circuit in the set.*

Proof Let C_1, \dots, C_N be a set of circuits that computes the set of all functions computable using L , and $C = C_1 C_1^{-1} \cdots C_N C_N^{-1}$. Then any test set that is complete for C must be complete for any circuit formed by L . Therefore, a minimal test set for C is (L, n) -complete. \square

The following proposition characterizes (L, n) -complete test sets for three classes of reversible circuits: C -constructible, U -constructible, and NCT -constructible.

Proposition 5

- a. A (C, n) -complete test set must have at least $n + 1$ vectors. One such set comprises the all-0's vector and the n weight-1 vectors.*
- b. A (U, n) -complete test set must have at least $2^{n-1} + 1$ vectors, and any $2^{n-1} + 1$ test vectors will give such a set.*
- c. An (NCT, n) -complete test set must have at least $2^{n-1} + 1$ vectors, and any $2^{n-1} + 1$ test vectors will give such a set.*

Proof

(a) Any input to the circuit can be written as a linear combination of the n weight-1 vectors. Furthermore, since the gate library is linear (under the operation of bitwise XOR), the corresponding values of the wires at the i th level can be written as the same linear combination of the values for these weight-1 vectors. If any input vector sets the value of a wire at the i th level to 1, then so must at least one weight-1 vector. Since there are inputs that do, the weight-1 vectors are sufficient for setting all wires to 1. Furthermore, since the circuit is linear, the all-0's vector sets all wires at all levels to 0. Therefore, this is a (C, n) -complete test set. In general any n linearly independent vectors along with the all-0's vector forms a (C, n) -complete test set.

On the other hand, if the test set consists of only n input vectors, we have two possibilities: either the set spans the n -dimensional space or it does not. If the latter is true, a linear reversible circuit can be constructed that maps the test set into the $(n - 1)$ -dimensional subspace $0X \cdots X$, implying that the test set is not complete. If the test set spans the entire n -dimensional space,

we can construct a linear reversible circuit that maps them to the following linearly independent vectors:

$$\begin{aligned}
 v_1 &\rightarrow 1\ 0\ 0\ 0\ \cdots\ 0 \\
 v_2 &\rightarrow 1\ 1\ 0\ 0\ \cdots\ 0 \\
 v_3 &\rightarrow 1\ 1\ 1\ 0\ \cdots\ 0 \\
 &\vdots\quad \vdots\quad \vdots\quad \vdots \\
 v_n &\rightarrow 1\ 1\ 1\ 1\ \cdots\ 1
 \end{aligned}$$

Since the first wire cannot be set to 0, the test set is not complete for this circuit.

(b) Suppose we have a (U, n) -complete test set with 2^{n-1} test vectors. Because the gate library computes all permutations, we can generate a circuit mapping all 2^{n-1} test vectors to output vectors of the form $0XX \cdots X$. This test set does not set the first output bit to 1, and thus is not complete for this U -gate circuit. This implies it is not (U, n) -complete. By Proposition 3a, any $2^{n-1} + 1$ test vectors will give (U, n) -completeness.

(c) Any permutation can be composed from a series of transpositions. The NCT gate library can construct circuits computing any even permutation of the input values [22], that is, a permutation that can be composed from an even number of transpositions. Following the proof for part b, a permutation can map any 2^{n-1} test vectors to output vectors of the form $0XX \cdots X$. If this permutation is even we have shown that this is an incomplete test set, otherwise we can add a transposition that exchanges the outputs $00 \cdots 0$ and $00 \cdots 1$. This new permutation is even and still maps the test vectors to the set of outputs $0XX \cdots X$, and therefore, the test set is not complete for this NCT -circuit. By Proposition 3a, any $2^{n-1} + 1$ test vectors will give (NCT, n) -completeness. \square

Note that any two gate libraries that can compute the same set of functions are equivalent with respect to (L, n) -completeness. This is because the function of every gate of one library can be computed by the other. Therefore, if a test set is not (L, n) -complete for one library, it cannot be for the other either, implying that the two libraries share the same (L, n) -complete test sets. This means that the above result for the U -gate library is applicable to any library that

can compute all n -bit reversible functions.

C-constructible circuits are analogous to XOR-trees, since C-NOT gates are simply XOR gates with an additional output that is equal to one of the inputs. Consequently, part (a) of the above proposition can be considered the reversible analog of the well-known result that any XOR-tree can be tested for single stuck-at faults using no more than four tests [14]. We consider linear reversible circuits separately here, primarily because they can be tested with a very simple set of tests just as XOR-trees in conventional irreversible circuit testing [8].

5 ILP formulation

While Proposition 3c guarantees that an efficient test set exists for any reversible circuit, it gives no practical construction. In this section, we formulate the problem of constructing a minimal test set as an integer linear program (ILP) with binary variables. We then use this to find a practical heuristic for generating efficient test sets.

5.1 ILP Model

We can formulate the minimal test set problem as an ILP with binary decision variables t_i associated with each input vector T_i ; t_i takes a value of one if the corresponding input vector is in the test set, and zero otherwise. A fault is detected if a decision variable with value one corresponds to a vector that detects the fault. The values of the wires at the j -th level for input T_i are $f_j(T_i)$, so to detect all stuck-at 0 faults at level j the following inequalities must be satisfied

$$\sum_{i=0}^{2^n-1} f_j(T_i) \cdot t_i \geq \underline{1}$$

These inequalities guarantee that each wire at the j -th level is set to 1 by some test vector. A similar set of inequalities ensures that all stuck-at 1 faults are also detected. A total of $2n(d+1)$ linear inequality constraints guarantees completeness. We determine a minimal test

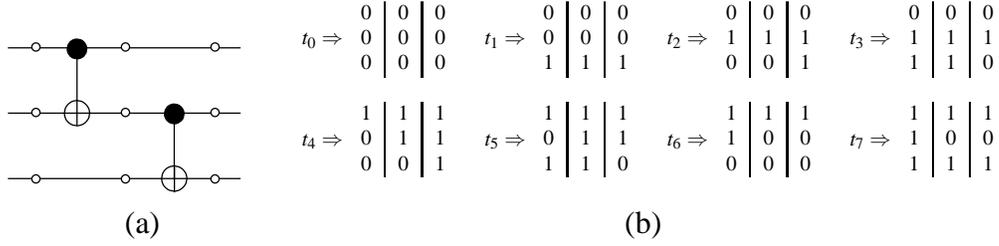


Figure 3: (a) Reversible circuit example. Possible stuck-at fault sites are represented by small open dots. (b) Propagation of each of the possible input vectors through the circuit. A complete test set must set each of the nine fault sites to both 0 and 1. An example of a complete test set for this circuit is $\{t_0, t_2, t_7\}$

set by minimizing the sum of the t_i 's subject to these constraints.

$$\text{Minimize } t_0 + t_1 + \dots + t_{2^n-1}$$

subject to the constraints

$$\sum_{i=0}^{2^n-1} f_j(T_i) \cdot t_i \geq 1$$

$$\sum_{i=0}^{2^n-1} \overline{f_j(T_i)} \cdot t_i \geq 1, \quad \text{for all } 0 \leq j \leq d$$

where $t_i \in \{0, 1\}$, $0 \leq i \leq 2^n - 1$, and

T_i is the n -bit binary expansion of integer i

Each feasible solution gives a complete test set composed of those vectors i for which $t_i = 1$. For relatively small circuits this ILP can be solved efficiently using an off-the-shelf optimization tool such as CPLEX [16].

As an example consider the circuit shown in Figure 3. The ILP formulation for the minimal test set problem is

$$\text{Minimize } t_0 + t_1 + t_2 + t_3 + t_4 + t_5 + t_6 + t_7$$

subject to the constraints

$$\begin{array}{l}
\Sigma_{i=0}^7 f_0(T_i) \cdot t_i \geq \underline{1} \\
\Sigma_{i=0}^7 f_1(T_i) \cdot t_i \geq \underline{1} \\
\Sigma_{i=0}^7 f_2(T_i) \cdot t_i \geq \underline{1}
\end{array}
\iff
\begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
t_0 \\
t_1 \\
t_2 \\
t_3 \\
t_4 \\
t_5 \\
t_6 \\
t_7
\end{bmatrix}
\geq
\begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1
\end{bmatrix}$$

$$\begin{array}{l}
\Sigma_{i=0}^7 \overline{f_0(T_i)} \cdot t_i \geq \underline{1} \\
\Sigma_{i=0}^7 \overline{f_1(T_i)} \cdot t_i \geq \underline{1} \\
\Sigma_{i=0}^7 \overline{f_2(T_i)} \cdot t_i \geq \underline{1}
\end{array}
\iff
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{bmatrix}
\cdot
\begin{bmatrix}
t_0 \\
t_1 \\
t_2 \\
t_3 \\
t_4 \\
t_5 \\
t_6 \\
t_7
\end{bmatrix}
\geq
\begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1
\end{bmatrix}$$

where $t_i \in \{0, 1\}$, $0 \leq i \leq 7$

The first set of inequalities guarantees that each of the fault sites can be set to 1 and the second set guarantees that each can be set to 0. Solving the ILP, we find that three test vectors are required to detect all stuck-at faults in the circuit. One such solution is $t_0 = t_7 = t_2 = 1$.

Using the ILP formulation and CPLEX 7.0, we obtained minimal test sets for all optimal 3-wire *NCT*-circuits. CPLEX was able to solve the ILP for each circuit in a fraction of a second on a Sun SPARC. Table 1 gives a distribution of minimum test set size with respect to the number of gates in the circuit. The optimal *NCT* implementation of a given function is not unique, and therefore the distribution in Table 1 may be dependent on the particular optimal set chosen.

As expected, the size of the minimal test set generally increases with the length of the circuit. However, there are long circuits that have smaller minimal test sets than much shorter circuits. The largest minimal test set has 4 vectors, however suboptimal circuits requiring 5 test

		Circuit Length (gates)								
		0	1	2	3	4	5	6	7	8
Test Size	2	1	6	24	67	134	155	105	21	-
	3	-	6	78	558	2641	8727	16854	10185	577
	4	-	-	-	-	5	39	90	47	-

Table 1: Minimal test set size distribution for optimal 3-wire *NCT*-circuits as a function of circuit length.

vectors can be constructed.

5.2 Circuit Decomposition Approach

Solving the ILP exactly is feasible for small circuits; however, since the number of variables increases exponentially with the number of input/output bits, it is impractical for large circuits. An alternative approach is to decompose the original circuit into smaller sub-circuits acting on fewer input/output bits, and use the ILP formulation iteratively for these sub-circuits combining the test vectors dynamically; a similar approach has been used for irreversible circuits [13]. While the resulting test set is not guaranteed to be minimal, it is generally small enough to enable efficient testing. Furthermore, it may be possible to use lower bounds to ensure the test set is not much larger than a minimal one. For example, the size of the minimal test set of a sub-circuit can be used to bound that of the larger circuit.

The algorithm shown in Figure 4 uses this decomposition approach. First the circuit is decomposed into a series of circuits acting on a smaller number of wires. One way to do this is to start at the input of the circuit and add gates to the first sub-circuit C_0 until no more can be added without having C_0 act on more than m wires. Then we continue with C_1 , and so on until the entire circuit has been decomposed. The remaining steps in the algorithm are best illustrated by an example.

Consider the decomposition of the reversible circuit in Figure 5. Though the entire circuit acts on six wires, each sub-circuit acts on no more than four. Using the ILP formulation on C_0

- 1) Partition circuit into disjoint sub-circuits C_0, \dots, C_l each acting on $\leq m$ wires
- 2) Initialize $\text{test_set} = \{\}$ and $i=0$
- 3) Generate ILP for C_i as in Section 5.1
- 4) Add constraints for each vector in test_set
- 5) Solve ILP
- 6) Incorporate new test vectors into test_set , setting any unused wires of new vectors to don't cares
- 7) Apply C_i to test_set , setting don't cares at inputs of C_i to 0
- 8) If $i < l$, $i=i+1$ and go to Step 3
- 9) Set remaining don't cares in test_set to 0
- 10) Apply C^{-1} to test_set to get complete test set

Figure 4: Algorithm for complete test set generation based on circuit decomposition.

gives test vectors:

$$\begin{array}{rcc}
 & \begin{array}{cccccc} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \end{array} & & \begin{array}{cccccc} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\
 v_0 = & \begin{array}{cccccc} \text{X} & 0 & 1 & \text{X} & 1 & 1 \end{array} & \xrightarrow{C_0} & \begin{array}{cccccc} \text{X} & 1 & 1 & \text{X} & 0 & 0 \end{array} \\
 v_1 = & \begin{array}{cccccc} \text{X} & 1 & 0 & \text{X} & 0 & 0 \end{array} & & \begin{array}{cccccc} \text{X} & 1 & 0 & \text{X} & 1 & 0 \end{array} \\
 v_2 = & \begin{array}{cccccc} \text{X} & 1 & 1 & \text{X} & 1 & 0 \end{array} & & \begin{array}{cccccc} \text{X} & 0 & 0 & \text{X} & 1 & 1 \end{array}
 \end{array}$$

where the X's represent don't cares and the left and right halves represent the test vectors at the input and output of C_0 , respectively. Sub-circuit C_1 acts on wires x_0, x_1, x_4 and x_5 . We generate the ILP for C_1 , and add the following constraints:

$$\begin{array}{rcc}
 \begin{array}{cccc} x_0 & x_1 & x_4 & x_5 \end{array} & & \begin{array}{c} \text{Constraints} \\ \hline \end{array} \\
 \begin{array}{cccc} \text{X} & 1 & 0 & 0 \end{array} & \in T \Rightarrow & t_4 + t_{12} \geq 1 \\
 \begin{array}{cccc} \text{X} & 1 & 1 & 0 \end{array} & & t_6 + t_{14} \geq 1 \\
 \begin{array}{cccc} \text{X} & 0 & 1 & 1 \end{array} & & t_3 + t_{11} \geq 1
 \end{array}$$

Solving this ILP gives the solution $t_6 = t_{11} = t_{12} = 1$. Incorporating these values into the

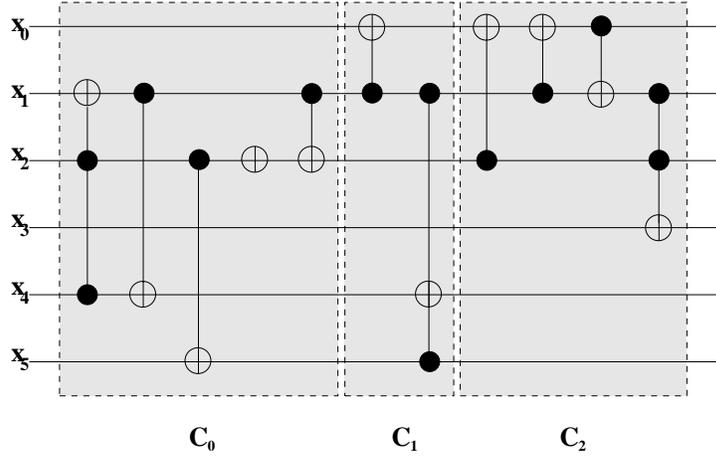


Figure 5: Circuit decomposition example.

previous test vectors we have:

$$\begin{array}{cccccc}
 x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \\
 \hline
 1 & 1 & 1 & X & 0 & 0 \\
 0 & 1 & 0 & X & 1 & 0 \\
 1 & 0 & 0 & X & 1 & 1
 \end{array}
 \xRightarrow{C_1}
 \begin{array}{cccccc}
 x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \\
 \hline
 0 & 1 & 1 & X & 0 & 0 \\
 1 & 1 & 0 & X & 1 & 0 \\
 1 & 0 & 0 & X & 1 & 1
 \end{array}$$

Sub-circuit C_2 acts on wires x_0 , x_1 , x_2 , and x_3 . We generate the ILP for this sub-circuit, and incorporate the current test set using the following constraints:

$$\begin{array}{cccc}
 x_0 & x_1 & x_4 & x_5 \\
 \hline
 0 & 1 & 1 & X \\
 1 & 1 & 0 & X \\
 1 & 0 & 0 & X
 \end{array}
 \in T \Rightarrow
 \begin{array}{c}
 \text{Constraints} \\
 \hline
 t_6 + t_7 \geq 1 \\
 t_{12} + t_{13} \geq 1 \\
 t_8 + t_9 \geq 1
 \end{array}$$

Solving this ILP gives solutions t_5 , t_7 , t_8 , and t_{12} . The last three can be incorporated into the

previous test set, however the first test vector must be added:

$$\begin{array}{cccccc}
 \underline{x_0} & \underline{x_1} & \underline{x_2} & \underline{x_3} & \underline{x_4} & \underline{x_5} & & \underline{x_0} & \underline{x_1} & \underline{x_2} & \underline{x_3} & \underline{x_4} & \underline{x_5} \\
 0 & 1 & 1 & 1 & 0 & 0 & & 0 & 1 & 1 & 0 & 0 & 0 \\
 1 & 1 & 0 & 0 & 1 & 0 & \xRightarrow{C_2} & 0 & 1 & 0 & 0 & 1 & 0 \\
 1 & 0 & 0 & 0 & 1 & 1 & & 1 & 1 & 0 & 0 & 1 & 1 \\
 0 & 1 & 0 & 1 & X & X & & 1 & 0 & 0 & 1 & X & X
 \end{array}$$

Filling the don't cares with 0's and applying C^{-1} to the test set yields a complete test set for C . While the resulting test set is not guaranteed to be minimal, in this case it is, as can be shown by applying the ILP method on the entire circuit.

5.3 Test Set Compaction

The circuit decomposition method in the previous section generally produces redundant test sets. One way to reduce this redundancy is to compact the test set, that is, find the smallest complete subset. This approach has been used previously in ATPG algorithms for conventional circuits [15, 7, 10]. The ILP formulation in Section 5.1 can be used to perform the test set compaction. We simply eliminate all test vectors that are not in the original complete test set, along with the corresponding columns in the constraint matrix. Generally, this ILP can be solved more efficiently than the ILP for the minimal test set, since it has fewer variables. Consider the example in the previous section. Since the circuit decomposition method yields a complete test set with four test vectors the ILP formulation for the test set compaction problem only requires four variable, significantly less than the 64 required in the ILP formulation for the minimal test set problem.

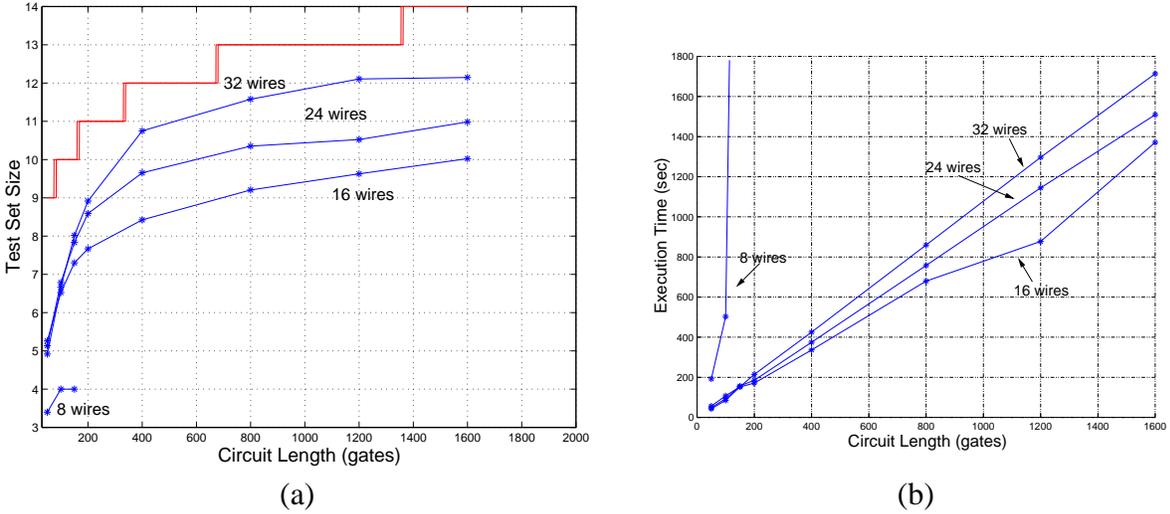


Figure 6: Simulations results for circuit decomposition algorithm limiting sub-circuit size to 8 wires. (a) Average test set size (after compaction) versus circuit length. Staircase graph represents the upper bound given in Proposition 3c. (b) Execution time versus circuit length.

5.4 Simulation Results

We conducted a set of simulations to evaluate the performance of our algorithm. We generated random *NCT*-circuits of various lengths over 8, 16, 24 and 32 wires. The circuits were generated by selecting at random from the set of all allowable NOT, C-NOT, and Toffoli gates. Each circuit was decomposed into sub-circuits acting on at most 8 wires, and our algorithm was used to find a complete test set. Figure 6a shows the average number of test vectors needed as a function of the circuit length. At least 150 circuits were generated for each data point.

The average execution time for the algorithm seems to increase linearly with circuit length and does not vary very much with the number of input/output wires, with the exception of the 8-wire case for which execution time appears to increase exponentially with circuit length (Figure 6b). This latter case is most likely because the number of constraints increases linearly with the number of gates, yielding increasingly difficult ILPs. On the other hand, for the circuits on more than 8 wires, an increase in the length of circuit does not generally lead to significantly harder individual ILPs, rather only a (linearly) larger number of them to solve.

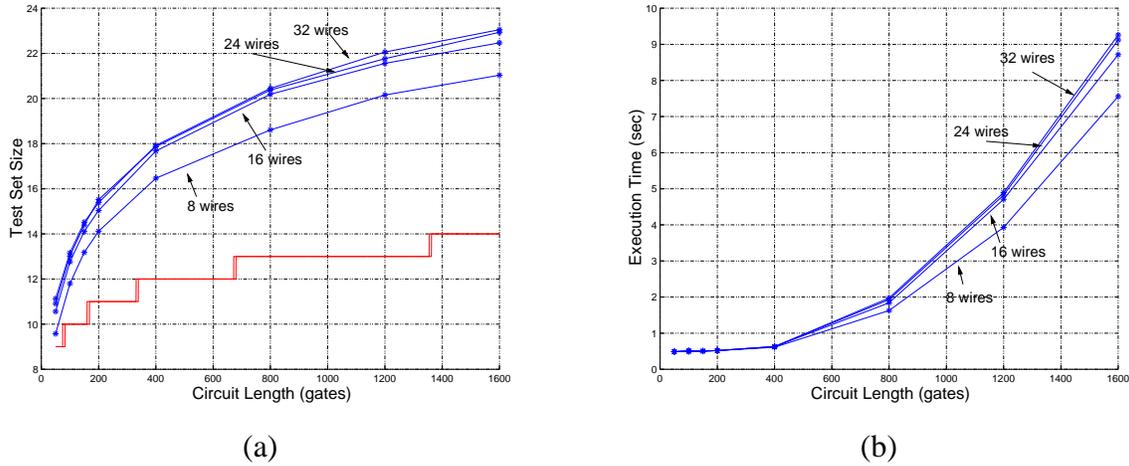


Figure 7: Simulation results for Atalanta [20]. (a) Average test set size versus circuit length. Staircase graph represents the upper bound given in Proposition 3c. (b) Execution time versus circuit length.

Test compaction, as expected, is most effective for longer circuits, eliminating an average of approximately one redundant test vector for circuits containing 800 or more gates.

5.5 Comparison to Conventional ATPG

A number of ATPG software packages are available for generating test sets for conventional combinational circuits, and some of these can be readily modified for the reversible case. Here we used the ATPG tool Atalanta [20], because of its ease of use and the availability of its source code. Since the Toffoli gate used in our reversible circuits is not a standard combinational logic gate, we had to make some minor modification to the code to handle this gate. Basically, we replaced each Toffoli gate by an equivalent combinational circuit using conventional irreversible gates, and modified the code to ignore faults in the internal nodes of these sub-circuits.

Figure 7 shows the average size of test sets generated by Atalanta as a function of the number of gates in the reversible circuit. Results are shown for 8, 16, 24 and 32 input/output wires. As the figure illustrates, the test sets given by Atalanta are, on average, almost twice as large as those given by our circuit decomposition algorithm, and their average size is greater

than the upper bound of Proposition 3c. However, Atalanta is significantly faster than our algorithm requiring an average of less than 10 seconds for circuits with 32 wires and 1600 gates; this compares to approximately 30 minutes for the circuit decomposition algorithm. However, the execution time for Atalanta appears to increase at a much faster rate with respect to the circuit length than that of the circuit decomposition algorithm.

6 Cell Fault Model

While the use of the stuck-at fault model has been very effective in conventional circuit testing, other fault models may be more appropriate for reversible circuits, especially in the quantum domain. For example, the cell fault model [17], where the function of the faulty $k \times k$ gate changes arbitrarily from the desired function, may be more realistic. In this section we extend some of our results to this model.

The following proposition provides a basic necessary and sufficient condition for a test set to be complete for the cell fault model. While this condition is also necessary for irreversible circuits, in that latter case it is not sufficient.

Proposition 6 *Under the cell fault model a test set is complete if and only if the inputs of every $k \times k$ gate in the circuit can be set to all 2^k possible values by the test set.*

Proof If a test set does not set the input wires of a gate to a particular value say \underline{a} , then it would not be able to detect a failure in this gate that only affects the output of \underline{a} . On the other hand, if the input wires of every gate in the circuit can be set to all possible values by the test set, then any single-gate failure will affect at least one test vector, changing the value at the output of the gate. By the observability property of reversible circuits, this will be reflected in a change at the output. \square

As an example, consider a circuit with a C-NOT gate. In order to detect any fault in the C-NOT gate the test set should be able to set the inputs of the gate to $\{00, 01, 10, \text{ and } 11\}$. If the

gate is faulty, it will operate incorrectly on at least one of these input values which will then be reflected in an incorrect circuit output.

Let g_1, \dots, g_l be the gates in a reversible circuit, and k_1, \dots, k_l the respective gate sizes. If we consider every possible value at the input of each gate as representing a distinct fault, the total number of faults that need to be covered is $\sum_{i=1}^l 2^{k_i}$. Under this definition, we have the following lemma.

Lemma 2 *Each input vector covers exactly l faults, and a fault associated with a $k \times k$ gate is covered by exactly 2^{n-k} input vectors.*

Proof Each input vector sets the bits at the inputs of each gate to some value. Therefore, since there are l gates, the vector can detect l faults. For a given fault associated with a $k \times k$ gate there are 2^{n-k} possible values for the n bits at that level that can detect it. Since the circuit is reversible, each of these can be traced back to a distinct input vector. \square

The following proposition, which is analogous to Proposition 3, gives upper bounds on the size of the minimal test set under the cell fault model.

Proposition 7 *A complete test set under the cell fault model for an n -wire reversible circuit with a total of l gates with sizes $k_1 \geq k_2 \geq \dots \geq k_l$ is given by*

- a. any $2^n - 2^{n-k_1} + 1$ distinct test vectors
- b. a set of $(\sum_{i=1}^l 2^{k_i}) - l + 1$ test vectors
- c. some set of at most $\sum_{i=1}^l \left\lceil \frac{2^{k_i}}{i} \right\rceil$ test vectors

Proof

(a) For any $k \times k$ gate in the circuit there are 2^{n-k} distinct inputs that yield a particular value at its input. Therefore, if the test set has $2^n - 2^{n-k_1} + 1$ vectors (implying that fewer than 2^{n-k} are not included) then it must include at least one such input. Since this is true for all gates in the circuit, by Proposition 6, the test set is complete.

(b) Any input vector will cover l faults leaving $\sum_{i=1}^l 2^{k_i} - l$. By the controllability property we can cover these with one test vector each. Therefore, all of the faults can be covered with no more than $\sum_{i=1}^l 2^{k_i} - l + 1$ test vectors.

(c) We first prove that given an incomplete set of m test vectors covering faults in the set F_C , there must be a test vector that covers at least

$$l - \left\lfloor \sum_{f \in F_C} 2^{-k(f)} \right\rfloor \quad (5)$$

of the remaining faults, where $k(f)$ is the size of the gate associated with fault f .

Suppose this is false. By Lemma 2 every test vector covers exactly l faults and a fault f is covered by exactly $2^{n-k(f)}$ input values. Therefore the number of times faults in F_C can be covered is $\sum_{f \in F_C} 2^{n-k(f)}$ and the current test set accounts for ml of these. Furthermore, each of the remaining input vectors must cover more than $\sum_{f \in F_C} 2^{-k(f)}$ of the already covered faults, otherwise our assertion would be true. Combining these we have the following inequalities.

$$(2^n - m) \left(\sum_{f \in F_C} 2^{-k(f)} \right) < \sum_{f \in F_C} 2^{n-k(f)} - m \cdot l \quad (6)$$

$$l < \sum_{f \in F_C} 2^{-k(f)} \quad (7)$$

The second inequality is false since the right side can be no larger than l . Therefore, we have a contradiction, and our proposition must be true.

Iteratively removing $l - \left\lfloor \sum_{f \in F_C} 2^{-k(f)} \right\rfloor$ faults from the set of uncovered faults eventually leaves the set empty. The number of iterations needed to do this is an upper bound on the number of test vectors needed for completeness. The floor function in the equation makes it difficult to obtain a closed form for the bound, but we can weaken the above result to do this. To cover the first 2^{k_l} faults we need at most $\lceil 2^{k_l} / l \rceil$ test vectors, since each test vector we add covers l faults. To cover the next $2^{k_{l-1}}$ faults we need at most $\lceil 2^{k_{l-1}} / (l-1) \rceil$ test vectors, and so on. Thus, we can cover all single cell faults using no more than $\sum_{i=1}^l \lceil 2^{k_i} / i \rceil$ test vectors. \square

For a reversible circuit on 64 wires with a million 3×3 gates, parts a-c of Proposition 7 give upper bounds of approximately 10^{19} , $7 \cdot 10^6$ and 10^6 test vectors, respectively. However, since part c uses the property illustrated in Equation (5) very conservatively in order to obtain a closed form, a much tighter bound can be obtained by applying this property directly. In fact, by iteratively applying this property, one can show that no more than 108 test vectors are needed for complete testing. In general we can approximate this tighter bound if we assume that the number of faults covered after any given iteration when taken mod 2^k is uniformly distributed on the integers 0 through $2^k - 1$ inclusively. The expected value of (5) is then

$$\mathbb{E} \left[l - \left\lfloor \sum_{f \in F_C} 2^{-k(f)} \right\rfloor \right] = \mathbb{E} \left[l - \left\lfloor \frac{|F_C|}{2^k} \right\rfloor \right] = l - \frac{|F_C|}{2^k} + \frac{2^k - 1}{2^{k+1}} \quad (8)$$

This then gives the following recursion on $C(n)$, the minimum number of faults that can be covered by n test vectors:

$$C(n) = C(n-1) + \left(l - \frac{C(n-1)}{2^k} + \frac{2^k - 1}{2^{k+1}} \right) \quad (9)$$

$$= \frac{2^k - 1}{2^k} \cdot C(n-1) + l + \frac{2^k - 1}{2^{k+1}} \quad (10)$$

Letting $a = (2^k - 1)/2^k$ and $b = l + (2^k - 1)/2^{k+1}$,

$$C(n) = a^2 \cdot C(n-2) + a \cdot b + b \quad (11)$$

$$= a^n \cdot C(0) + (a^{n-1} + a^{n-2} + \dots + 1) \cdot b \quad (12)$$

$$= \frac{1 - a^n}{1 - a} \cdot b = 2^k \cdot \left(1 - \left(\frac{2^k - 1}{2^k} \right)^n \right) \cdot \left(l + \frac{2^k - 1}{2^{k+1}} \right) \quad (13)$$

In order to cover all faults, $C(n)$ must be greater than or equal to the total number of faults:

$$C(n) = 2^k \cdot \left(1 - \left(\frac{2^k - 1}{2^k} \right)^n \right) \cdot \left(l + \frac{2^k - 1}{2^{k+1}} \right) \geq l \cdot 2^k \quad (14)$$

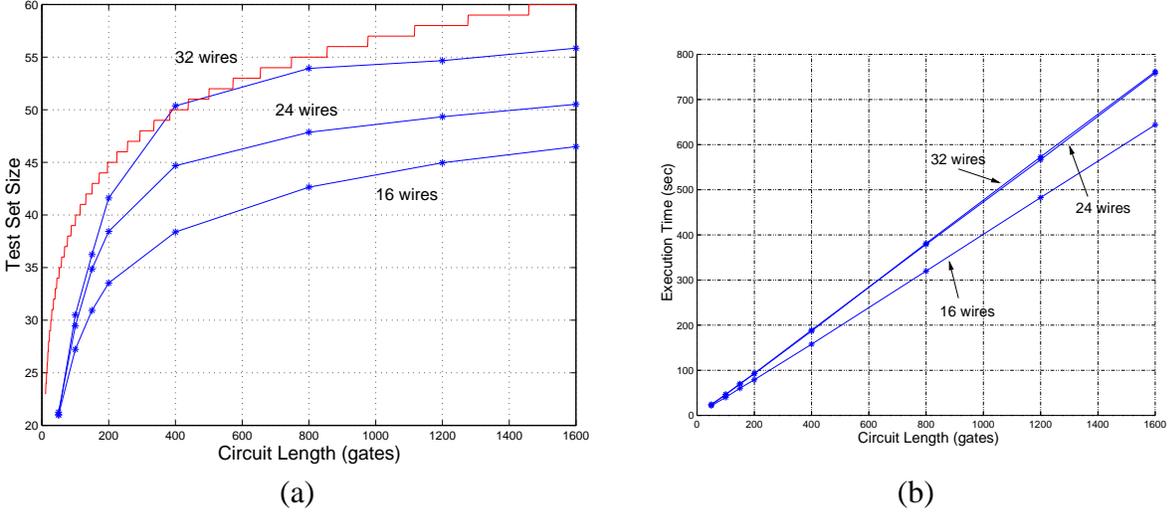


Figure 8: Simulation results for cell fault model. (a) Average test set size (after compaction) vs. circuit length for circuit decomposition algorithm using the cell fault model. The sub-circuit sizes are limited to 8 wires in the decomposition. The staircase graph represents the upper bound given by iterating the property in (5). (b) Execution time versus circuit length.

On rearranging we have

$$n \geq \frac{\log_2 \left(\frac{2^{k+1}}{2^k - 1} \cdot l + 1 \right)}{k - \log_2 (2^k - 1)} \quad (15)$$

This result strongly suggests the number of test vectors needed for completeness under the cell fault model grows logarithmically in the number of gates, just as in the stuck-at fault case.

To obtain an ILP formulation for the cell fault model only the constraints given in Section 5 need to be modified. For each $k \times k$ gate at each level we generate 2^k constraints, one for each of the possible inputs to the gate. The circuit decomposition method from Section 5.2 as well as the test set compaction method in Section 5.3 can be applied as in the stuck-at fault case. Figure 8 shows simulation results for the circuit decomposition algorithm under the cell fault model. The staircase graph represents the upper bound obtained by iterating (5). The average size of the generated test set is generally below this bound. Test set compaction results in the elimination of 4-5 test vectors on average, with more vectors eliminated for longer circuits and fewer for shorter ones.

Empirically, the algorithm requires half the execution time of the stuck-at fault case, with the exception of the 8-wire case for which we were unable to obtain results due to prohibitive execution times.

7 Conclusions

We have considered the test set generation problem for reversible circuits, and shown that the property of reversibility fundamentally simplifies the problem. For example, a test set that detects all single stuck-at faults in a reversible circuit also detects all multiple faults. We have derived test set completeness conditions under both the stuck-at and cell fault models. We have then used these to find general test set constructions that implicitly yield upper bounds on the number of test vectors needed for completeness. One bound shows that the test set size increases at most logarithmically with the length of circuit, strengthening our assertion that reversible circuits are easier to test than conventional ones. We have also given a practical algorithm for finding complete test sets. Our algorithm generates test sets that are approximately half the size of those produced by conventional ATPG.

In addition to the fault detection problem we have investigated here, we also plan to study fault diagnosis, that is, using test sets to localize faults. As with the detection problem, fault diagnosis may be easier for reversible circuits than for irreversible ones. Finally, though we have focused on testing for classical reversible circuits, we also hope to extend our work to the quantum case. The latter is likely to be very different from the former: while fault-free classical circuits are deterministic, fault-free quantum ones are inherently probabilistic. Thus, the goal for the quantum case may be to determine as efficiently as possible, and with a given degree of confidence, whether the circuit contains a fault or not. A step towards this goal may be to find a small set of test vectors that sufficiently exercises the internal gates in the circuit. Our results for the cell fault model studied in Section 6 may be particularly useful for this.

References

- [1] V. D. Agrawal. An Information Theoretic Approach to Digital Fault Testing. *IEEE Transactions on Computers*, vol. 30, pp. 582–587, August 1981.
- [2] C. H. Bennett. Logical Reversibility of Computation. *IBM Journal of Research and Development*, vol. 17, pp. 525–532, November 1973.
- [3] J. C. Bertrand, N. Giambiasi, and J. J. Mercier. Sur la Recherche de l’Inverse d’un Automate. *RAIRO*, pp. 64–87, April 1974.
- [4] J. C. Bertrand, J. J. Mercier, and N. Giambiasi. Sur la Recherche de l’Inverse d’un Circuit Combinatoire. *RAIRO*, pp. 21–44, July 1974.
- [5] H. Buhrman, J. Tromp, and P. Vitányi. Time and Space Bounds for Reversible Simulation. *Journal of Physics A: Mathematical and General*, vol. 34, pp. 6821–6830, September 2001.
- [6] M. L. Bushnell and V. D. Agrawal. *Essentials of Electronic Testing for Digital Memory & Mixed-Signal VLSI Circuits*. Kluwer Academic Publishers, Boston, 2000.
- [7] J.-S. Chang and C.-S. Lin. Test Set Compaction for Combinational Circuits. *IEEE Transactions on CAD*, vol. 14, pp. 1370–1378, November 1995.
- [8] W. H. Debany Jr., C. R. P. Hartmann and T. J. Snethen. Algorithm for Generating Optimal Tests for Exclusive-OR Networks. *IEE Proceedings E (Computers and Digital Techniques)*, vol. 138, pp. 93–96, March 1991.
- [9] B. Desoete and A. De Vos. A Reversible Carry-Look-Ahead Adder Using Control Gates. *Integration, The VLSI Journal*, vol. 33, pp. 89–104, 2002.
- [10] P. F. Flores, H. C. Neto, and J. P. Marques-Silva. On Applying Set Covering Models to Test Set Compaction. *Proceedings of GLS-VLSI*, pp. 8–11, March 1999.

- [11] E. Fredkin and T. Toffoli. Conservative Logic. *Intl. Journal of Theoretical Physics*, vol. 21, pp. 219–253, 1982.
- [12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.
- [13] P. Goel and B. C. Rosales. Test Generation & Dynamic Compaction of Test. *Digest of Papers Test Conference*, pp. 189–192, October 1979.
- [14] J. P. Hayes. On Realizations of Boolean Functions Requiring a Minimal or Near Minimal Number of Tests. *IEEE Transactions on Computers*, vol. 20, pp. 1506–1513, December 1971.
- [15] D. S. Hochbaum. An Optimal Test Compression Procedure for Combinational Circuits. *IEEE Transactions on CAD*, vol. 15, pp. 1294–1299, October 1996.
- [16] ILOG CPLEX. <http://www.ilog.com/products/cplex>.
- [17] W. H. Kautz. Testing for Faults in Cellular Logic Arrays. *Annual Symposium on Switching and Automata Theory*, pp. 161–174, 1967.
- [18] R. Landauer. Irreversibility and Heat Generation in the Computing Process. *IBM Journal of Research and Development*, vol. 3, pp. 183–191, July 1961.
- [19] M. Li, J. Tromp, and P. Vitányi. Reversible Simulation of Irreversible Computation. *Physica D*, pp. 168–176, September 1998.
- [20] H. K. Lee and D. S. Ha. On the Generation of Test Patterns for Combinational Circuits. *Technical Report No. 12_93*, Dept. of Electrical Engineering, Virginia Polytechnic Institute and State University.
- [21] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

- [22] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes. Synthesis of Reversible Logic Circuits. *IEEE Transactions on CAD*, vol. 22, pp. 710–722, June 2003.
- [23] T. Toffoli. Reversible Computing. *Automata, Languages and Programming, 7th Colloquium*, J. W. de Bakker and J. van Leeuwen (eds.), Lecture Notes in Computer Science No. 85, Springer-Verlag, pp. 632-644, 1980.
- [24] V. V. Zhirnov, R. K. Calvin III, J. A. Hutchby, and G. I. Bourianoff. Limits to Binary Logic Switch Scaling—A Gedanken Model. *Proceedings of the IEEE*, vol. 91, pp. 1934–1939, November 2003.