Test Pattern Generation Effort Evaluation of Reversible Circuits

Abhoy Kole[1], Robert Wille[2], Kamalika Datta[3], and Indranil Sengupta[4](✉)

[1] B.P. Poddar Institute of Management and Technology, Kolkata, India
abhoy.kole@gmail.com
[2] Johannes Kepler University Linz, Linz, Austria
robert.wille@jku.at
[3] National Institute of Technology Meghalaya, Shillong, India
kdatta@nitm.ac.in
[4] Indian Institute of Technology Kharagpur, Kharagpur, India
isg@iitkgp.ac.in

Abstract. The problem of synthesis and optimization of reversible and quantum circuits have drawn the attention of researchers for more than one decade. With physical technologies for realizing the quantum bits (qubits) being announced, the problem of testing such circuits is also becoming important. There have been several works for identifying fault models for reversible circuits, and test generation algorithms for the same. In this work, we aim to show that the problem of testing reversible circuits with respect to recent fault models (like missing gate, missing control, reduced control, etc.) is easy, and it is not really worth to spend time and effort for generating better test patterns. To establish this point, test generators using two extreme scenarios have been implemented: a naive test generator that is very fast but does not guarantee optimality and a SAT-based test generator that is slow but guarantees smallest test sets. Experiments have been carried out on reversible benchmark circuits, which establish the fact that the size of the test patterns does not drastically differ across the spectrum.

Keywords: Reversible circuit · ATPG · SAT · Optimization

1 Introduction

A circuit is said to be reversible if it provides a bijective mapping between the input and output lines, which implies that the number of input and output lines are equal. A reversible circuit is composed as a cascade of simple reversible gates, without any fanout or feedback connections. Reversible circuits have been studied extensively in the literature as an alternate computing paradigm with some potential for low power design (see e.g. [2,8] or, more recently, [3]) or quantum computation [11].

Particular for the latter domain, how to implement corresponding circuits has intensely been considered. To this end, various methods for synthesis and

I. Phillips and H. Rahaman (Eds.): RC 2017, LNCS 10301, pp. 162–175, 2017.
DOI: 10.1007/978-3-319-59936-6_13

optimization have been proposed (see e.g. [4,14]). For their physical realization, various quantum gates and methods of decomposing reversible circuits to circuits composed of the corresponding gate library have been reported. Here, particular the NCV library [9] or more recently the Clifford+T [1] library received attention. In this regard, it has to be considered that, in a quantum circuit, information is represented in terms of qubits, which can not only be in the states 0 and 1, but also any superposition of them. Reversible circuits however provide a good basis for this, since every quantum gate operation is also reversible in nature.

With various technologies beginning to emerge that are able to implement and manipulate qubits, researchers have also been looking at the various fault effects and models that such circuits can be subjected to. In conventional gate level circuits, where each gate is physically implemented and a signal moves from one gate to the next, wire-level fault models like stuck-at or bridging faults have become popular. In contrast, a quantum circuits consists of a set of qubits whose interactions are controlled by applying a sequence of control pulses; in other words, the same set of qubits perform the gate operations sequentially. Therefore, wire-level fault models are not relevant to quantum circuits, and newer fault models like missing-gate or missing-control have been proposed.

There have been several reported works that target the testing of faults in reversible circuits – and, by this, conduct *Automatic Test Pattern Generation* (ATPG). Initial works [12] used the stuck-at fault models for reversible circuits; however, subsequent works relied on more realistic fault models addressing physical realization constraints [5]. The various test pattern generation methods that have been reported can be broadly categorized as: (a) branch-and-bound methods [5], (b) methods based on *Integer Linear Programming* (ILP) [13], SAT-based and PBO-based methods [17–19], etc. In addition, there has been several works on *design for testability* (DFT), where by adding some extra gates or controls, the faults can be tested using very few test patterns [10].

Most of these methods suffer from scalability problems as they often aim to find the best possible, i.e. smallest possible, solution and rely on tools like ILP, SAT, or PBO solvers that do not scale well with problem size. In this work, we are questioning whether these efforts are really worth it. In fact, it is evident that the problem of testing reversible circuits is much simpler as compared to that for conventional circuits – since the problems of controllability and observability are naturally solved by the bijective mappings for every gate operation. Because of this, it is rather easy to implement an ATPG tool that would run very fast and, at the same time, still would generate the desired set of test patterns which is of moderate size.

Motivated by that, the main objectives of the proposed work is as follows:

(a) Evaluate test generation methods from an effort-quality tradeoff point of view. Specifically, how naive ATPG algorithms perform as compared to the optimum ATPG approaches.
(b) Establish the fact that it does not make much sense to spend time and effort in minimizing the number of test patterns in reversible circuits. Because of its inherent properties, such circuits in any case do not require too many patterns for testing.

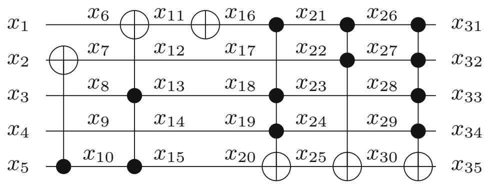


Fig. 1. Example reversible circuit

The rest of the paper is organized as follows. Section 1 provides a brief literature survey covering reversible circuits and gates, the fault models, and the notable test generation works reported in the literature. Section 2 provides the details for our proposed evaluation, where we discuss a naive test generation approach that is fast but non-optimal, and a SAT-based approach that is slow but optimal. Results of the experimental evaluation will be presented and discussed in Sect. 3. Finally, Sect. 4 summarizes the work with concluding remarks and some directions for future work.

In this section, we briefly review the basics of reversible circuits, the fault models, and the test generation approaches that have been proposed for reversible circuits.

1.1 Reversible Circuits and Gates

In our work, we only consider reversible circuits that are composed of multiple-control Toffoli (MCT) gates, also known as k-CNOT gates. A k-CNOT gate has $k + 1$ inputs and outputs, with k control connections $(c_1, c_2, \ldots, c_k)$ and one target line t. The logic value of the target line t gets inverted only when all the lines with control connections are at logic 1, while the logic values of all the other lines remain unchanged. In other words, the new value of t becomes $t_{new} = (c_1.c_2 \cdots c_k) \oplus t$.

Figure 1 shows a reversible circuit with 6 gates, comprising of one 0-CNOT gate, one 1-CNOT gate, two 2-CNOT gate, one 3-CNOT gate and one 4-CNOT gate.

Since the state of the target line of a k-CNOT gate is computed using the XOR operation, it is clear that every gate is reversible; if the output vector is applied to the output of the gate, we get back the previous input. When we generate test vectors to detect faults in the gates, this property results in the following unique characteristics.

(a) For any state vector S_i applied to the input of any gate g_i, it will map to a unique state vector in the primary outputs (PO). Any bit change(s) in S_i will result in a different unique state vector in PO. This is due to the bijective property of reversible circuits.
(b) For any state vector S_i applied to the input of any gate g_i, it is always possible to get a unique state vector at the primary inputs (PI). This can be

achieved by back-tracing from gate g_i, and evaluating every gate encountered in the reverse direction.

These characteristics help to solve the controllability and observability problems during testing of reversible circuits, which is known to be one of the biggest obstacles in the testing of conventional circuits.

1.2 Reversible Circuit Fault Models

It is known that reversible gates can be decomposed into quantum gates using some quantum gate library (e.g. NCV). It has been mentioned in [11] that quantum gates can be implemented using various nanotechnologies. Some of these approaches use the quantum states of sub-atomic particles (like spin-up and spin-down) to represent the qubits. The qubit states are modified by applying very short-width electromagnetic pulses to implement the quantum gate functions. In trapped-ion technology, for instance, individual atoms can represent the qubits. The atomic states are altered by applying precise laser pulses of specified frequency and duration. A sequence of such pulses has to be applied in time to execute the gates that comprise a reversible or quantum circuit.

In some earlier works, classical stuck-at or bridging fault models at the reversible circuit level was considered. However, because of the dynamic nature of evaluation of the quantum gates by application of pulses, the applicability of such wire-oriented classical fault models is doubtful. As discussed in [5], a suitable fault model in the quantum domain should largely be technology independent, and based on errors with regards to the application of evaluation pulses. The following fault scenarios can result.

(a) *Missing Gate Fault*, where due to absence of a pulse, a gate might not evaluate at all.
(b) *Repeated Gate Fault*, where due to multiple pulses being generated instead of just one, a gate might be evaluated multiple (say, k) times. Since the effect of two identical gate operations cancel each other, when k is even, this also reduces to the *Missing Gate Fault*. And when k is odd, the fault is undetectable.
(c) *Missing Control Fault* (also known as *Partial Missing Gate Fault*) where a gate gets evaluated even when some subset of the control lines is active, because of partially misaligned or mistuned gate pulses.
(d) *Additional Control Fault*, where some control connection gets added in addition to the already existing connections.

Earlier works have considered single missing gate fault (SMGF), and single missing control fault (SMCF/1-PMGF) for test generation and analysis. However, SMCF can be generalized to *Partial Missing Gate Fault (PMGF)*, where multiple missing control faults within a gate can also occur. Inclusion of one additional control line leads to single additional control fault (SACF). Besides that, the considerations conducted in this work can similarity be applied to any other fault model for reversible circuits (even those which are about to be proposed in the future).

1.3 Existing ATPG Solutions for Reversible Circuits

Various works on *Automatic Test Pattern Generation* (ATPG) for reversible circuits have been reported in the literature. The problem of test set generation and reduction of test set has been addressed using ILP in [12,13]. Exact approaches that generate smallest test set has also been considered previously in [18,19]. The authors in [6,7] proposed approaches to derive test sets for detecting multiple missing-gate faults in reversible circuits. Besides exact approaches (using SAT and PBO), a simulation based approach has also been considered in [17].

2 Proposed Work

In the present paper, we aim to evaluate how the effort in generating tests for reversible circuits correlate with the quality of test vectors generated. For the purpose of evaluation, we have considered two extremes of the spectrum with respect to test generation in a reversible circuit:

(a) A naive approach that directly generates a test for an undetected fault that is expected to require larger number of test vectors.
(b) An exact (minimal) approach that would generate the smallest possible test set.

In the following subsections, we evaluate and compare the number of test vectors for these two extreme scenarios.

2.1 Naive Test Pattern Generation

Consider a reversible circuit consisting of p gates $\{G_1, G_2, \ldots, g_p\}$. For every gate G_i, $i = 1$ to p, we do the following:

(i) Generate the fault list F consisting of all faults according to a given fault model.
(ii) Generate a set of vectors $\{V_i\}$ at the input of gate G_i that can detect all faults $f \in F$ in G_i.
(iii) Repeat the following steps for all vectors $v \in \{V_i\}$:
   (a) Back propagate v to obtain the corresponding input test vector, say T. Since every gate is reversible, for a given v, T will be unique.
   (b) Carry out fault simulation with test vector T to determine the faults in F that get detected.
   (c) Remove the detected faults from F (*fault dropping*).
(iv) Continue with Step i. The process terminates as soon as the fault list F becomes empty.

To detect an SMCF in G_i, all control lines (except the missing one) have to be assigned to 1, while the missing control line has to be assigned to 0. The assignment of the remaining lines can be chosen arbitrarily. Similarly, to detect an SMGF in G_i, all control lines have to be assigned to 1, while the remaining lines can be arbitrarily assigned. Again, to detect single additional control fault (SACF) in G_i, all control lines except the additional one have to be assigned to 1, and the additional line has to be assigned to 0. The assignment of the remaining lines can be arbitrarily chosen.

Three alternate methods for filling up the remaining lines (i.e. the don't care bits) have been studied:

(a) *0-filling*: where a don't care line is set to logic value 0.
(b) *1-filling*: where a don't care line is set to logic value 1.
(c) *Random filling*: where a don't care line is randomly set to 0 or 1.

The naive test generation approach with the three alternatives for don't care filling have been studied for the SMGF and SMCF models. The overall runtime of this naive approach is $\mathcal{O}(n^2)$ (For a reversible circuit with n gate, the number of generated test vectors encompassing all fault models is Cn, where C is a constant, and back propagation and fault dropping for each test vector on average is $\frac{n}{2}+n$.).

2.2 Exact (Minimal) Test Pattern Generation

In this subsection we discuss an approach that uses Boolean satisfiability to generate minimum test patterns for detecting faults in a reversible circuit. To this end, we utilize a SAT formulation which is similar to the one proposed in [18]. We state below the SAT formulation for SMGF and for SMCF. We also discuss the SAT formulation of the combined SMGF+SMCF model and the PMGF model which have not been considered earlier. By this we show that this solution (and, hence, the considerations conducted here) can be applied to ATPG of reversible circuits in general and does not rely on a particular fault model.

(a) **SMGF**: The SAT based formulation for detecting the presence of SMGF in a circuit C is:

$$\bigwedge_{i=1}^{n} I_i \wedge \bigwedge_{g_k \in C} \left(\bigvee_{i=1}^{n} f_i(g_k) \right) \tag{1}$$

where for the k-th C^mNOT gate, g_k $(T(\{x_1^i, x_2^i, \ldots, x_m^i\}; x_{m+1}^i))$ from i-th circuit instance $f_i(g_k) = x_1^i \wedge x_2^i \wedge \cdots \wedge x_m^i$ for all missing gate, $g_k \in C$.

(b) **SMCF**: The SAT based formulation for detecting the presence of SMCF in a circuit C is:

$$\bigwedge_{i=1}^{n} I_i \wedge \bigwedge_{g_k \in C} \left(\bigvee_{i=1}^{n} f_i^{'}(g_k) \right) \tag{2}$$

where for the k-th C^mNOT gate, g_k $(T(\{x_1^i, x_2^i, \ldots, x_m^i\}; x_{m+1}^i))$ from i-th circuit instance $f_i^{'}(g_k) = x_1^i \wedge x_2^i \wedge \cdots \wedge \overline{x_j^i} \wedge \cdots \wedge x_m^i$ for all missing control line, $x_j^i \in \{x_1^i, x_2^i, \ldots, x_m^i\}$.

(c) **SMGF+SMCF**: We can also combine the SMGF and SMCF fault models in a single unified formulation. The SAT formulation for the combined fault model for a given circuit C is:

$$\bigwedge_{i=1}^{n} I_i \wedge \overbrace{\bigwedge_{g_k \in C} \left(\bigvee_{i=1}^{n} f_i(g_k) \right)}^{SMGF} \wedge \overbrace{\bigwedge_{g_k \in C} \left(\bigvee_{i=1}^{n} f_i^{'}(g_k) \right)}^{SMCF} \tag{3}$$

where for the k-th $C^m NOT$ gate, g_k $(T(\{x_1^i, x_2^i, \ldots, x_m^i\}; x_{m+1}^i))$ from i-th circuit instance $f_i(g_k) = x_1^i \wedge x_2^i \wedge \cdots \wedge x_j^i \wedge \cdots \wedge x_m^i$ for all missing gate, $g_k \in C$ and $f_i^{'}(g_k) = x_1^i \wedge x_2^i \wedge \cdots \wedge \overline{x_j^i} \wedge \cdots \wedge x_m^i$ for all missing control line, $x_j^i \in \{x_1^i, x_2^i, dots, x_m^i\}$.

(d) **PMGF**: The PMGF fault model is a superset of the SMCF fault model. The SAT based formulation for detecting PMGFs in a circuit C is:

$$\bigwedge_{i=1}^{n} I_i \wedge \bigwedge_{g_k \in C} \left(\bigvee_{i=1}^{n} f_i^1(g_k) \right) \wedge \ldots \left(\bigvee_{i=1}^{n} f_i^n(g_k) \right) \tag{4}$$

where for the k-th $C^m NOT$ gate, g_k $(T(\{x_1^i, x_2^i, \ldots, x_m^i\}; x_{m+1}^i))$ from i-th circuit instance

$$\begin{aligned} f_i^1(g_k) &= x_1^i \wedge \cdots \wedge \overline{x_j^i} \wedge \cdots \wedge x_m^i, \\ f_i^2(g_k) &= x_1^i \wedge \cdots \wedge \overline{x_j^i} \wedge \cdots \wedge \overline{x_k^i} \wedge \cdots \wedge x_m^i, \\ \ldots &= \ldots, \\ f_i^n(g_k) &= \overline{x_1^i} \wedge \overline{x_2^i} \wedge \cdots \wedge \overline{x_m^i}. \end{aligned}$$

Here the term $f_i^1(g_k)$ is identical to the term $f_i^{'}(g_k)$ of SMCF.

(e) **SMGF+PMGF**: We can also combine the SMGF and PMGF fault models in a single unified formulation. The SAT formulation for the combined fault model for a given circuit C is:

$$\bigwedge_{i=1}^{n} I_i \wedge \overbrace{\bigwedge_{g_k \in C} \left(\bigvee_{i=1}^{n} f_i(g_k) \right)}^{SMGF} \wedge \overbrace{\bigwedge_{g_k \in C} \left(\bigvee_{i=1}^{n} f_i^1(g_k) \right) \wedge \ldots \left(\bigvee_{i=1}^{n} f_i^n(g_k) \right)}^{PMGF} \tag{5}$$

where for the k-th $C^m NOT$ gate, g_k $(T(\{x_1^i, x_2^i, \ldots, x_m^i\}; x_{m+1}^i))$ from i-th circuit instance

$$\begin{aligned} f_i(g_k) &= x_1^i \wedge x_2^i \wedge \cdots \wedge x_j^i \wedge \cdots \wedge x_m^i, \\ f_i^1(g_k) &= x_1^i \wedge x_2^i \wedge \cdots \wedge \overline{x_j^i} \wedge \cdots \wedge x_m^i, \\ f_i^2(g_k) &= x_1^i \wedge \cdots \wedge \overline{x_j^i} \wedge \cdots \wedge \overline{x_k^i} \wedge \cdots \wedge x_m^i, \\ \ldots &= \ldots, \\ f_i^n(g_k) &= \overline{x_1^i} \wedge \overline{x_2^i} \wedge \cdots \wedge \overline{x_m^i}. \end{aligned}$$

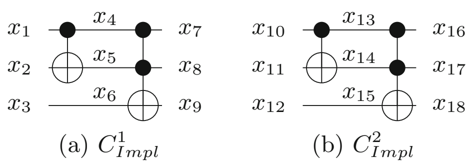


Fig. 2. An example circuit with multiple instances

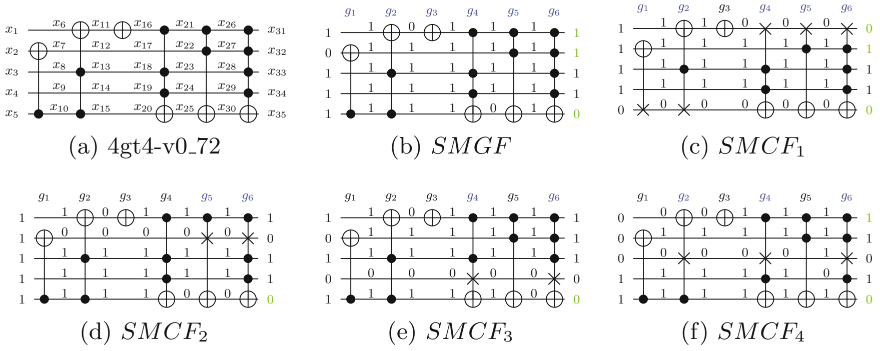


Fig. 3. Detecting *SMGF* and *SMCF* faults for (a) the benchmark 4gt4-v0_72 with variables assigned before and after each gate operation, (b) input pattern detects all possible SMGFs that may produce affected output and (c)–(f) undesired output and corresponding specific SMCFs detected by different input patterns

The basic idea behind the generation of minimal test sets for reversible circuit using SAT based approach is illustrated with the help of an example illustrated in Fig. 2. If the number of test vectors required is t, then we need to have t instances of the circuit with all the lines distinctly labeled. The figure shows two instances C^1_{Impl} and C^2_{Impl} for $t = 2$.

For the instance C^1_{Impl} (see Fig. 2a) the equations representing the circuit behavior are formed as:

$$\begin{aligned} I_1 = (x_4 = x_1) \wedge (x_5 = x_2 \oplus x_1) \wedge (x_6 = x_3) \wedge (x_7 = x_4) \\ \wedge (x_8 = x_5) \wedge (x_9 = x_6 \oplus x_4 x_5) \end{aligned}$$

Similarly, for the instance C^2_{Impl} shown in Fig. 2b, the equations are formed as:

$$\begin{aligned} I_2 = (x_{13} = x_{10}) \wedge (x_{14} = x_{11} \oplus x_{10}) \wedge (x_{15} = x_{12}) \\ \wedge (x_{16} = x_{13}) \wedge (x_{17} = x_{14}) \wedge \ (x_{18} = x_{15} \oplus x_{13} x_{14}) \end{aligned}$$

The SAT formulation for SMCF as per Eq. (2) will be

$$I_1 \wedge I_2 \wedge (\overline{x_1} \vee \overline{x_{10}}) \wedge (\overline{x_4} x_5 \vee \overline{x_{13}} x_{14}) \wedge (x_4 \overline{x_5} \vee x_{13} \overline{x_{14}})$$

The SAT formulation for SMGF as per Eq. (1) will be

$$I_1 \wedge I_2 \wedge (x_1 \vee x_{10}) \wedge (x_4 x_5 \vee x_{13} x_{14})$$

Given these formula as inputs, a SAT solver will provide a set of test vectors as output.

2.3 Test Generation for Several Fault Models Using SAT Solver

If we combine the equations for SMCF and SMGF into a single set of equations, and feed the same to a SAT solver, we shall get the test patterns required to test both single missing-control and also single missing-gate faults. The total number of test patterns is expected to be less in the combined approach.

We illustrate the idea with the help of a benchmark circuit, viz. 4gt4-v0_72. Figure 3a shows the reversible logic implementation. We show the test patterns generated under the following three fault model scenario.

(a) **SMGF**: The SAT solver returns the following single test vector covering all SMGFs of the circuit shown in Fig. 3a:
   (i) $[x_1, x_2, x_3, x_4, x_5] = [1, 0, 1, 1, 1]$, which detects SMGFs for all the gates as shown in Fig. 3b.

(b) **SMCF**: The SAT solver returns the following 4 test vectors covering all SMCFs of the circuit shown in Fig. 3a:
   (i) $[x_1, x_2, x_3, x_4, x_5] = [1, 1, 1, 1, 0]$, which detects SMCFs for the gates g_1, g_2, g_4, g_5 and g_6 as shown in Fig. 3c.
   (ii) $[x_1, x_2, x_3, x_4, x_5] = [1, 1, 1, 1, 1]$, which detects SMCFs for the gates g_5 and g_6 as shown in Fig. 3d.
   (iii) $[x_1, x_2, x_3, x_4, x_5] = [1, 0, 1, 0, 1]$, which detects SMCFs for the gates g_4 and g_6 as shown in Fig. 3e.
   (iv) $[x_1, x_2, x_3, x_4, x_5] = [0, 0, 0, 1, 1]$, which detects SMCFs for the gates g_2, g_4 and g_6 as shown in Fig. 3f.

(c) **Combined (SMGF+SMCF)**: For the combined formulation, the SAT solver returns the following 5 test vectors covering all SMGFs and SMCFs of the circuit shown in Fig. 3a:
   (i) $[x_1, x_2, x_3, x_4, x_5] = [1, 1, 1, 1, 1]$, which detects SMCFs for the gates g_5 and g_6, and SMGFs for the gates g_1, g_2, g_3 and g_4 as shown in Fig. 3d.
   (ii) $[x_1, x_2, x_3, x_4, x_5] = [0, 0, 0, 1, 1]$, which detects SMCFs for the gates g_2, g_4 and g_6, and SMGFs for the gates g_1, g_3 and g_5 as shown in Fig. 3f.
   (iii) $[x_1, x_2, x_3, x_4, x_5] = [1, 0, 1, 1, 1]$, which detects SMGFs for the gates g_1, g_2, g_3, g_4, g_5 and g_6 as shown in Fig. 3b.
   (iv) $[x_1, x_2, x_3, x_4, x_5] = [1, 1, 1, 1, 0]$, which detects SMCFs for the gates g_1, g_2, g_4, g_5 and g_6 as shown in Fig. 3c.
   (v) $[x_1, x_2, x_3, x_4, x_5] = [1, 0, 1, 0, 1]$, which detects SMCFs for the gates g_4 and g_6, and SMGFs for the gates g_1, g_2, g_3 and g_5 as shown in Fig. 3e.

(d) **PMGF**: The SAT solver returns 15 test vectors covering all PMGFs of the circuit shown in Fig. 3a.

(e) **Combined (SMGF+PMGF)**: For the combined formulation, the SAT solver returns 16 test vectors covering all SMGFs and PMGFs of the circuit shown in Fig. 3a.

3 Experimental Evaluation

Experiments have been carried out on reversible benchmark circuits available in [16]. The naive test generator (with fault simulator) have been implemented in C and run on a core-i3 machine with 4GB RAM, running Ubuntu v16.04. The test generators using SAT solvers have been implemented on the RevKit [15] platform, using C++ and Python, and run on the same core-i3 machine.

Using these implementations, we conducted the evaluations motivated in Sect. 1. Tables 1 and 2 provide a summary of some of the obtained numbers. More precisely, Table 1 shows the results of the SAT-based ATPG implementation with combined fault models and also the naive ATPG implementation. Results for 0-filling, 1-filling and random filling (best out of 5 runs) are also shown. Table 2 show the results for larger benchmark circuits using the naive ATPG tool with 0-filling, 1-filling and random-filling. Since those benchmarks cannot be handled by the SAT-based exact approach (due to run-time limitations), no corresponding numbers for this solutions are provided here.

Based on these numbers (as well as further case studies for which we cannot present all numbers due to page limitations), the following conclusions can be drawn:

Table 1. Combined fault model and comparison with naive approach

Benchmarks				SAT based ATPG						Naive ATPG (SMGF+SMCF)			
Circuit	d	n	c	Separate		SMGF+SMCF		SMGF+PMGF		0-fill	1-fill	r-fill	T
				F_{tot}	P_{tot}	P	T	P	T	P	P	P	
4gt4-v0_78	13	5	1	31	8	6	0.01	16	0.04	9	7	7	0.00
4gt12-v0_86	14	5	1	34	6	5	0.01	16	0.05	10	7	8	0.00
decod24-enable_32	14	9	6	31	3	3	0.01	4	0.01	6	5	3	0.00
mod5d1_16	15	8	3	34	3	3	0.01	4	0.01	9	7	5	0.00
4_49_16	16	4	0	40	8	5	0.01	8	0.02	8	5	7	0.00
miller_5	16	8	5	40	5	5	0.02	6	0.02	8	7	7	0.00
3_17_6	17	7	4	37	4	3	0.02	4	0.01	7	5	6	0.00
mini-alu_84	20	10	6	47	3	3	0.03	4	0.01	9	6	8	0.00
rd53_131	28	7	2	52	15	10	0.04	16	0.08	14	13	14	0.00
rd84_142	28	15	7	77	7	5	0.09	5	0.04	29	10	9	0.00
sym6_63	29	14	8	72	5	5	0.05	6	0.04	17	11	9	0.00
4_49_7	42	15	11	103	5	5	0.06	6	0.06	14	11	8	0.00
ham15_108	70	15	0	195	17	11	0.17	16	0.36	12	12	11	0.00
hwb5_13	88	28	23	219	5	5	0.30	6	0.20	29	17	10	0.00
ham15_109	109	15	0	235	11	7	0.23	16	0.39	15	10	9	0.00
ham15_107	132	15	0	484	20	16	78.59	–	–	37	24	23	0.04
hwb6_14	159	46	40	400	6	5	0.51	6	0.57	51	30	13	0.06
ex5p	647	206	198	1551	–	–	–			172	117	17	3.55

d: number of gates, n: number of lines, c: number of constant lines
F_{tot}: total number of faults, T: Time in seconds
P_{tot}: total number of test patterns (when run separately)
P: number of test patterns, r: random filling

Table 2. ATPG test patterns for larger benchmarks

Benchmarks				Faults		ATPG			
Circuit	d	n	c	F_1	F_2	0-fill	1-fill	r-fill	Time (s)
9symml_195	129	10	1	474	129	222	193	194	0.2
add6_196	229	19	7	853	229	210	174	141	0.68
alu2_199	157	16	6	567	157	211	198	202	0.33
alu4_201	1063	22	8	5535	1063	1735	1549	1573	120.77
bw_291	307	87	82	432	307	49	37	13	0.26
clip_206	174	14	5	653	174	175	163	160	0.34
dist_223	185	13	5	727	185	155	144	147	0.32
e64-bdd_295	387	195	130	454	387	135	129	14	0.9
f51m_233	663	22	8	3296	663	1252	1091	1127	33.67
frg1_234	212	31	3	1343	212	928	447	697	2.78
ham15_298	153	45	30	157	153	33	26	8	0.04
hwb7_302	281	73	66	426	281	76	40	14	0.27
hwb7_62	331	7	7	582	331	66	52	47	0.29
hwb8_116	749	8	8	1317	749	126	85	77	2.62
hwb8_303	449	112	104	686	449	126	53	14	1.05
hwb9_123	1959	9	9	3596	1959	275	168	150	38.59
hwb9_304	699	170	161	1068	699	184	73	17	3.55
in0_235	338	26	11	2107	338	423	406	408	3.37
in2_236	405	29	10	2475	405	506	454	451	5.26
life_238	107	10	1	387	107	138	136	129	0.1
max46_240	107	10	1	371	107	189	172	171	0.13
mlp4_245	131	16	8	480	131	138	92	95	0.15
plus127mod8192_162	910	13	13	5704	910	1072	105	311	27.61
plus63mod8192_164	492	13	13	3064	492	765	121	286	6.65
rd84_253	111	12	4	315	111	116	116	107	0.09
sym10_262	194	11	1	818	194	307	290	313	0.71
sym9_148	210	10	1	756	210	210	10	29	0.17
sym9_193	129	10	1	474	129	222	193	195	0.21
table3_264	1012	28	14	8002	1012	978	912	946	53.52
tial_265	1041	22	8	5517	1041	1767	1625	1614	119.85
urf1_150	1517	9	9	6077	1517	259	149	148	20.56
urf1_151	1487	9	9	5878	1487	256	149	144	19.68

d: number of gates, n: number of lines, c: number of constant lines
F_1: number of SMCFs, F_2: number of SMGFs

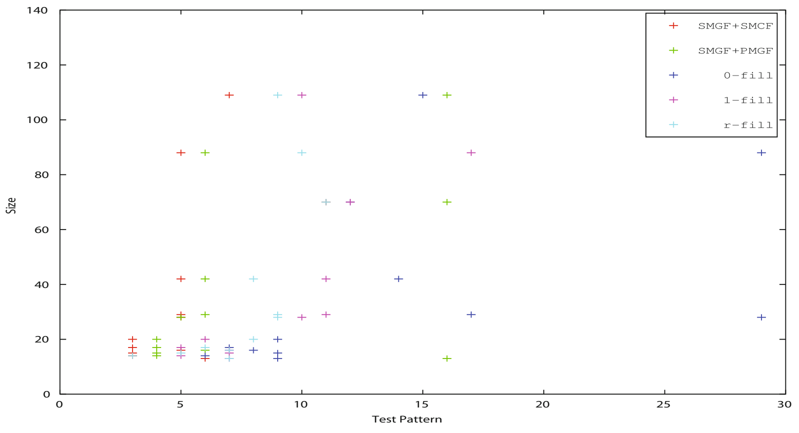
- Exact test pattern generation using the SAT-based exact approach is time-consuming and not scalable. While small circuits can indeed be handled in some seconds, no results can be obtained for circuits composed of some dozens of circuit lines and hundreds of gates. This is not very surprising considering the exponential complexity of guaranteeing minimal test sets.
- Despite the efforts spent on guaranteeing minimality, determining the minimal test set often yields only moderate improvements compared to the naive approach. In fact, the size of the test sets obtained by the naive approach is often only a few patterns bigger than the test sets obtained by the SAT-based minimal approach.

These evaluations confirm that, from an effort-quality tradeoff point of view, there is no real need to spent much effort into the optimization of test pattern generation for reversible circuits. In fact, naive solutions as sketched in this work already yield results which are close to the optimum. In contrast, further improving them towards minimality often comes with an increase in the run-time and a substantially reduced scalability so that it is often not worthwhile to spent these efforts (for a relatively small gain).

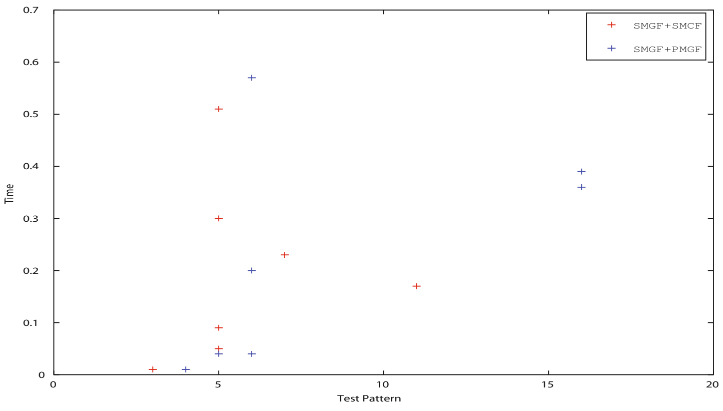
4 Conclusion

In this paper we have carried out an evaluation of the various alternate test generation techniques for reversible circuits, and how it impacts the quality of the test. To observe the entire spectrum of variability, we have implemented test generators touching the two extremes. Firstly, a naive test generator has been implemented that is very fast and uses a greedy approach to generate test patterns. Secondly, a SAT based test generator has been implemented that generates the smallest test set but requires large run times. The variation in the number of test patterns, in spite of a very large variation in run times, is not significantly large. This summarizes the main finding of the work, namely, test generation for reversible circuit is easy and naive solutions are often sufficient. Vice versa, it is not worthwhile to spent much efforts on the development of more sophisticated solutions since the possible gain will be moderate.

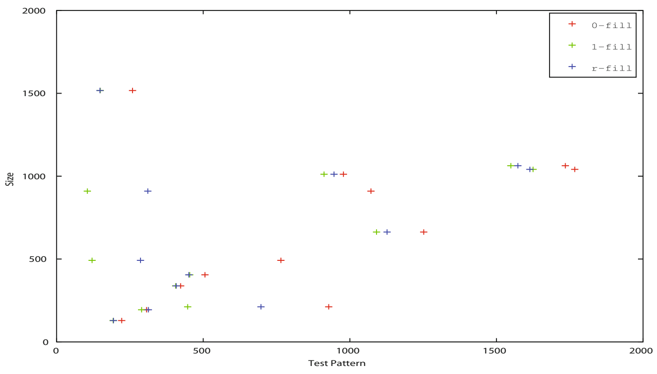
Although SAT-based exact approach produces smaller number of test patterns compared to naive approach as shown in Fig. 4a, it takes longer duration even for circuits with small number of gates, as shown in Fig. 4b. Here the comparison is made for the benchmarks with upto 109 gates reported in Table 1. Figure 4c shows 0-filling produces larger number of test patterns than 1-filling and r-filling as presented for larger benchmarks in Table 2.



(a) Size



(b) Time



(c) Larger Benchmark

Fig. 4. Evaluation of approaches (a) comparison of number of test patterns generated using SAT-based and naive approaches for smaller size benchmarks from Table 1, (b) time taken by various SAT-based approaches and corresponding test pattern generated for small size benchmark from Table 1 compared to naive approach and (c) number of test patterns generated by different naive approaches for the benchmarks from Table 2

References

1. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Trans. CAD **32**(6), 818–830 (2013)
2. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**(6), 525–532 (1973)
3. Berut, A., Arakelyan, A., Petrosyan, A., Ciliberto, S., Dillenschneider, R., Lutz, E.: Experimental verification of Landauer's principle linking information and thermodynamics. Nature **483**, 187–189 (2012)
4. Drechsler, R., Wille, R.: From truth tables to programming languages: progress in the design of reversible circuits. In: International Symposium on Multi-valued Logic (2011)
5. Hayes, J.P., Polian, I., Becker, B.: Testing for missing-gate faults in reversible circuits. In: Asian Test Symposium, pp. 100–105 (2004)
6. Kole, D.K., Rahaman, H., Das, D.K., Bhattacharya, B.B.: Derivation of automatic test set for detection of missing gate faults in reversible circuits. In: International Symposium on Electronic System Design (ISED), pp. 200–205, December 2011
7. Kole, D.K., Rahaman, H., Das, D.K., Bhattacharya, B.B.: Derivation of test set for detecting multiple missing-gate faults in reversible circuits. Comput. Electr. Eng. **39**(2), 225–236 (2013)
8. Landauer, R.: Irreversibility and heat generation in computing process. IBM J. Res. Dev. **5**(3), 183–191 (1961)
9. Miller, D.M., Wille, R., Sasanian, Z.: Elementary quantum gate realizations for multiple-control Toffolli gates. In: International Symposium on Multi-valued Logic (2011)
10. Mondal, J., Das, D.K., Kole, D.K., Rahaman, H.: A design for testability technique for quantum reversible circuits. In: East-West Design & Test Symposium (EWDTS 2013) (2012)
11. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge University Press, New York (2000)
12. Patel, K.N., Hayes, J.P., Markov, I.L.: Fault testing for reversible circuits. IEEE Trans. CAD **23**(8), 1220–1230 (2004)
13. Polian, F., T., Becker, B., Hayes, J.P.: A family of logical fault models for reversible circuits. In: Asian Test Symposium, pp. 422–427 (2004)
14. Saeedi, M., Markov, I.L.: Synthesis and optimization of reversible circuits - a survey. ACM Comput. Surv. **45**(2), 21:1–21:34 (2013)
15. Soeken, M., Frehse, S., Wille, R., Drechsler, R.: RevKit: an open source toolkit for the design of reversible circuits. In: Vos, A., Wille, R. (eds.) RC 2011. LNCS, vol. 7165, pp. 64–76. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29517-1_6
16. Wille, R., Grosse, D., Teuber, L., Dueck, G.W., Drechsler, R.: Revlib: an online resource for reversible functions and reversible circuits. In: International Symposium on Multi-valued Logic, pp. 220–225, May 2008
17. Wille, R., Zhang, H., Drechsler, R.: ATPG for reversible circuits using simulation, Boolean satisfiability, and pseudo Boolean optimization. In: IEEE Annual Symposium on VLSI, pp. 120–125, July 2011
18. Zhang, H., Frehse, S., Wille, R., Drechsler, R.: Determining minimal testsets for reversible circuits using Boolean satisfiability. In: AFRICON, pp. 1–6 (2011)
19. Zhang, H., Wille, R., Drechsler, R.: SAT-based ATPG for reversible circuits. In: International Design and Test Workshop, pp. 149–154, December 2010