Determining Minimal Testsets for Reversible Circuits Using Boolean Satisfiability

Hongyan Zhang  Stefan Frehse  Robert Wille  Rolf Drechsler
Institute of Computer Science, University of Bremen
28359 Bremen, Germany
{zhang,sfrehse,rwille,drechsle}@informatik.uni-bremen.de

Abstract—Reversible circuits are an attractive computation model as they theoretically enable computations with close to zero power consumption. Furthermore, reversible circuits found significant attention in the domain of quantum computation. With the emergence of first physical realizations for this kind of circuits, also testing issues become of interest. Accordingly, first approaches for automatic test pattern generation have been introduced. However, they suffer either from their limited scalability or do not generate a minimal testset. In this paper, a SAT-based algorithm for the determination of minimal complete testsets is proposed. An experimental evaluation of the proposed method shows that the algorithm is applicable to reversible circuits with more than 2 000 gates.

I. INTRODUCTION

Reversible circuits are \( n \)-inputs, \( n \)-outputs circuits in which every input pattern maps to a unique output pattern. As a result, computations in reversible circuits can be performed in both directions, i.e. from the inputs to the outputs and vice versa.

The reversibility of computation has intensely been studied as it provides an alternative to conventional circuits where power dissipation can theoretically be reduced or even eliminated [1], [2]. While in conventional logic energy amounting to \( kT \cdot \ln 2 \) is dissipated for each lost bit of information (where \( k \) is the Boltzmann’s constant and \( T \) is the temperature), reversible circuits are not affected by this. This makes reversible circuits interesting for low-power applications in the future.

Besides that, reversible circuits received significant attention in the domain of quantum computation [3]. Here, qubits instead of bits are applied which cannot only store the conventional Boolean values 1 and 0, but any superposition of them. Because of this, quantum circuits can solve many practical relevant problems significantly faster than their conventional counterparts. Since every quantum computation inherently is reversible, reversible circuits are of interest in this domain.

Motivated by these applications, researchers started to develop respective design methods for this new computation model (see e.g. [4], [5], [6], [7], [8]). With the emergence of first physical realizations for this kind of circuits, also testing issues become of interest. In this context, researchers studied different fault models and the respective methods for Automatic Test Pattern Generation (ATPG).

While conventional fault models (like the stuck-at fault model) have been considered at the beginning [9], new models addressing physical realizations of reversible circuits have been introduced later [10]. Among them are the missing gate fault model and the missing control line fault model (also known as the partial missing gate model). These fault models remain to be computationally tractable, while at the same time being applicable to different kinds of technologies.

Along with the fault models, researchers also started to develop test pattern generation methods. A major goal is thereby to keep the size of the testset (i.e. the number of test patterns needed to detect all considered faults in a circuit) as small as possible. Different approaches based on greedy and branch-and-bound methods [10], ILP formulations [11] as well as SAT-based approaches [12] and PBO-based methods [13] have been introduced for this purpose. However, they suffer either from their limited scalability (i.e. they are only applicable to circuits with a small number of gates [10], [11]) or do not generate a minimal testset [12], [13].

In this paper, we present an approach which determines a minimal testset for a given reversible circuit. The general idea is to iteratively check, whether for a given circuit and a given fault list a testset detecting all faults with only \( k \) patterns exists. By starting these checks with \( k = 1 \) and iteratively increasing \( k \) by one, minimality is ensured. The respective checks are thereby conducted by solvers for Boolean satisfiability. Experiments demonstrate that using the proposed approach, minimal testsets for reversible circuits can efficiently be generated.

The remainder of this paper is structured as follows. The next section introduces the basics on reversible circuits as well as on Boolean satisfiability. Section III introduces the fault models considered in this paper and defines the term of a minimal testset. Afterwards, the proposed approach is described in Section IV. Finally, experiments results are provided in Section V, while Section VI concludes the paper.
II. BACKGROUND

In this section, reversible circuits and the basics of Boolean satisfiability are briefly reviewed.

A. Reversible Circuits

A reversible function is a function $f: \mathbb{B}^n \to \mathbb{B}^m$ over inputs $X = \{x_1, ..., x_n\}$ with two properties: (1) its number of inputs is equal to its number of outputs (i.e. $n = m$) and (2) it maps each input pattern to a unique output pattern. A reversible circuit is a realization of a reversible function. Accordingly, reversible circuits also have $n$-inputs, $n$-outputs, and map each input pattern to a unique output pattern. Because of that, the output assignment can be obtained from the input assignment and vice versa. In comparison to a conventional circuits, fanout and feedback are not allowed in reversible circuits [3]. As a result, every reversible circuit $G$ is composed of a cascade of reversible gates $g$, i.e. $G = g_1g_2...g_d$. In this work, we consider the most widely used reversible gate, the Toffoli gate [14]. A Toffoli gate is defined as follows:

**Definition 1:** A Toffoli gate over the set of inputs $X = \{x_1, ..., x_n\}$ has the form $g(C, x_t)$, where $C \subseteq X$ is the set of control lines and $x_t \in X \setminus C$ is the target line. A single Toffoli gate $g(C, x_t)$ realizes the bijective function

$$(x_1, ..., x_n) \mapsto (x_1, ..., x_{t-1}, x_t \oplus \bigwedge_{x \in C} x_c, x_{t+1}, ..., x_n).$$

That is, the target line $x_t$ is inverted if (1) all control line variables $x_c \in C$ are assigned to 1 or (2) the set of control lines is empty, i.e. $C = \emptyset$. In these cases, the gate is called activated. All other values of $x_c$ with $x_k \in X \setminus \{x_i\}$ always pass the gate unaltered.

**Example 1:** Fig. 1(a) shows an example of a reversible circuit which is composed of Toffoli gates. This circuit has five circuit lines and four Toffoli gates, i.e. $n = 5$ and $d = 4$. Control lines are denoted by a $\bullet$, while the target line is denoted by an $\oplus$. The annotated values demonstrate the computation of the respective gates for a certain input pattern. In this case, the gates $g_2$ and $g_3$ are activated.

B. Boolean Satisfiability

The Boolean satisfiability (SAT) problem is defined as follows:

**Definition 2:** Let $h$ be a Boolean function. Then, the SAT problem is to determine an assignment to all variables of $h$ such that $h$ evaluates to 1 or to prove that no such assignment exists. In the case a satisfying assignment exists, the respective instance is called satisfiable (SAT); otherwise the instance is called unsatisfiable (UNSAT). Usually, the Boolean formula is thereby given in Conjunctive Normal Form (CNF), i.e. in a product-of-sum representation. A CNF consists of a conjunction of clauses. A clause is a disjunction of literals and each literal is a propositional variable or its negation.

**Example 2:** Let $h$ be a Boolean function in CNF with $h = (x_1 + x_2 + \overline{x}_3)(\overline{x}_1 + x_3)(\overline{x}_2 + x_3)$. Then, $x_1 = 1$, $x_2 = 1$, and $x_3 = 1$ is a satisfying assignment for $h$. The values of $x_1$ and $x_2$ ensure that the first clause becomes satisfied while $x_3$ ensures this for the remaining two clauses.

Because of its significance in both, theoretical research and practical applications, the SAT problem is one of the intensely studied $NP$-complete problems. The $NP$-completeness of the SAT problem has been proven by Cook in 1971 [15]. Despite this complexity, very efficient algorithms and techniques have been developed in order to solve this kind of problems. Besides learning [16] and efficient implication strategies [17] also very powerful term re-writing techniques [18] are applied today. Because of that, nowadays instances including hundreds of thousands of variables and clauses, respectively, can be solved in short time.

III. TEST OF REVERSIBLE CIRCUITS

For a given circuit $G$, the goal of Automatic Test Pattern Generation (ATPG) is to create a testset $T_F$, i.e. a set of stimulus patterns, which detects faults provided in a fault list $F$. The fault list $F$ is composed of all possible faults that may occur in the circuit according to a given fault model.

A. Fault Models

In this paper, we explicitly consider the fault models introduced in [11] and defined as follows:

**Definition 3:** Let $g(C, x_t)$ be a Toffoli gate of a circuit $G$. Then,

1) a Single Missing Control Fault (SMCF) occurs if instead of $g$ a gate $g'(C', x_t)$ with $C' = C \setminus \{x_i\}$ is executed
of the circuit. Depending on the considered fault, this requires
activated so that the faulty behavior shows up at the outputs
Partial Missing Gate Fault Model.

In order to detect a fault, the respective gates have to be
kept as small as possible. In fact, a
minimal
This obviously reduces the size of the testset.

which are additionally detected are removed from the fault list.
each newly obtained test pattern is simulated and further faults
can be performed after each test pattern generation. That is,
test pattern might cover more than one fault, fault simulation
complete testset can be determined. Moreover, since a single
faulty behavior is always shown on at least one primary output.
observability is also always ensured, i.e. in case of a fault, the

towards the primary inputs. Since reversible circuits have full
input assignments given in Definition 4 to the gate with
computed in polynomial time with respect to the the number
inputs as shown in [13]), determining a complete testset is
easy for reversible circuits, since a complete testset can be

determined is much harder than deter-
mining a complete testset. In fact, for certain fault models
it has been proven that minimal testset generation is even
NP-hard (see e.g. [20]). In this paper, we propose an ap-
proach that uses Boolean satisfiability techniques to generate
a minimal testset for the SMCF and the SMGF models.

IV. Determining a Minimal Testset

In this section, we show how a minimal complete testset
for a given circuit \( G \) and a fault list \( \mathcal{F} \) can be determined
using techniques for Boolean satisfiability. The general flow
is presented first. Afterwards, details on the structure of the
proposed SAT instance as well as on the actual encoding are
provided.

A. General Flow

In order to determine a minimal testset, an iterative approach
is proposed. The basic idea of the flow is as follows: Given
a circuit \( G \) and a fault list \( \mathcal{F} \), first it is checked whether a
complete testset exists that consists of \( k = 1 \) test pattern only.
If no such test pattern can be determined, \( k \) is increased by
one. This procedure is repeated until a testset results which
detects all faults provided in \( \mathcal{F} \). By iteratively increasing \( k \) by
one, minimality is ensured.

This procedure is formalized in Algorithm 1. The function
\( \text{minATPG}(G, \mathcal{F}) \) gets the circuit \( G \) as well as \( \mathcal{F} \) and initializes
\( k = 1 \). Then, in each iteration the question “Does there exist a
testset consisting of \( k \) patterns and detecting all faults \( f \in \mathcal{F} \in \mathcal{F} \)
in the circuit \( G \)?” is encoded as a SAT instance (Line 4) and
passed to a solver (Line 5). If the solver returns UNSAT, no

\begin{algorithm}
\caption{\textsc{minATPG}(G, \mathcal{F}): Algorithm Determining a Minimal Testset.}
\begin{algorithmic}[1]
\Statex \textbf{Input}: \( G \) a reversible circuit, \( \mathcal{F} \) a fault list
\Statex \textbf{Output}: A minimal complete testset \( T_{\mathcal{F}} \)
\Statex \begin{array}{c}
\textbf{k} = 1; \\
\textbf{while} \text{ true} \textbf{ do} \\
\quad \text{enc} = \text{encode}(\text{ATPG}(G, \mathcal{F}, k)); \\
\quad \text{if} \text{ solve(enc)} = \text{UNSAT} \text{ then} \\
\quad \quad \textbf{k} = \textbf{k} + 1; \\
\quad \text{else} \\
\quad \quad \text{return} \ T_{\mathcal{F}} \text{ obtained from the satisf. assgmt.} \\
\textbf{end}
\end{array}
\end{algorithmic}
\end{algorithm}

1\textsuperscript{1}Note that in the literature (e.g. in [11]), the SMCF model is also called
Partial Missing Gate Fault Model.
such testset exists and \( k \) is increased by one (Line 6). Otherwise (i.e. if the solver returns SAT), the complete minimal testset can be obtained from the satisfying assignment of the respective variables. If in contrast the SAT solver returns unsatisfiable, it has been proven that no complete testset with \( k \) patterns exists.

C. Encoding of the Instance

In order to encode the instance presented in the last section, a formulation based on the SAT-based ATPG approach proposed in [12] is applied. The encoding of the respective circuit copies as well as the encoding of the respective fault constraints are introduced in the following.

1) Encoding the Circuit Copies \( G^i \): First, the encoding of the circuit copies is presented. Given a reversible circuit \( G \) with \( n \) lines and \( d \) gates, \( k \) circuit copies need to be encoded. For each copy \( G^i \) with \( 1 \leq i < k \), variables \( x_{i,j}^{g,j} = x_{i,n}^{g,j}, x_{i,n-1}^{g,j} \ldots x_{i,1}^{g,j} \) for \( \mu \in \{1, \ldots, d + 1\} \) are introduced representing the assignment to the primary inputs (for \( \mu = 1 \)), the primary output (for \( \mu = d + 1 \)) as well as the inputs and outputs of the gates (for \( 2 \leq \mu \leq d \)), respectively. Fig. 3 shows those variables for one copy \( G^i \) of the circuit from Fig. 1(a).

In order to model the functionality of the circuit copy, the following constraints are added to the SAT instance for each circuit copy, i.e. for each \( i \in \{1, \ldots, k\} \):

\[
\bigwedge_{j=1}^{d} \bigwedge_{l=1}^{n} x_{i,l}^{j+1} =
\begin{cases}
  x_{i,l}^{j}, & \text{if } x_{i,l}^{j+1} \text{ represents a control line of gate } g_{j} \\
  x_{i,l}^{j} \oplus \bigwedge_{x_{j} \in C_{j}} x_{c}, & \text{if } x_{i,l}^{j} \text{ represents the target line of gate } g_{j} \\
  x_{i,l}^{j}, & \text{else (i.e. if } x_{i,l}^{j} \text{ represents neither a control line nor a target line of gate } g_{k})
\end{cases}
\]

That is, for each gate \( g_{j} \) in the circuit copy, the respective input/output mapping is constrained (depending on the position of the control and target lines). In other words, the values of all lines (except the target line) are passed through (first and third case), while the output value for the target line is determined depending on the input values of the control and the target line, respectively (second case). The bottom-left part of Fig. 3 shows the respective constraints for one copy \( G^i \) of the circuit from Fig. 1(a).

2) Encoding the Faults Variables \( f^{G^i} \): Finally, the encoding of the constraints for the \( f^{G^i} \)-variables is presented. As described above, \( f^{G^i} \) is supposed to evaluate to 1 (0) if a test pattern is applied to \( G^i \) that detects the fault \( f \) (that does not detect the fault \( f \)). Consequently, just the respective input
assignments for the given fault as provided in Definition 4 has to be applied to the corresponding gate.

More precisely, let $f \in \mathcal{F}$ be an SMCF in a gate $g_j(C_j,t_j)$ with a missing control $x_m \in C_j$. Then, the corresponding $f^{G_i}$-variable is constrained as follows:

$$f^{G_i} = \left( \bigwedge_{x \in C \setminus \{x_m\}} x^{ij}_{c} \right) \land x^{ij}_{m}$$

Similarly, let $f \in \mathcal{F}$ be an SMGF in a gate $g_j(C_j,t_j)$. Then, the corresponding $f^{G_i}$-variable is constrained as follows:

$$f^{G_i} = \bigwedge_{x_i \in C_j} x^{ij}_{i}$$

The bottom-right part of Fig. 3 shows the respective constraints for an SMCF and an SMGF.

V. EXPERIMENTAL EVALUATION

The proposed approach has been implemented in C++ on top of RevKit [21]. Boolector [18] was used as the satisfiability solver. Circuits taken from RevLib [22] were evaluated with respect to both considered fault models. The experiments have been carried out on an AMD Opteron $\times$4 processor with 3GHz and 32GB main memory running Linux. The timeout was set to 3 600 CPU seconds.

The results have been compared to previously introduced ATPG approaches for reversible circuits, namely the SAT-based approach introduced in [12] and the PBO-based approach introduced in [13], respectively. Both approaches are not intended to compute minimal testsets. However, the difference between the size of the testsets obtained by these approaches and the size of the (minimal) testsets obtained by the proposed approach is investigated.

The results are presented in Table I. The first four columns describe the characteristics of the evaluated circuits, namely (1) the name of the circuit, (2) the number of gates, (3) the number of lines, and (4) the number of constant inputs. Column $|\mathcal{F}|$ denotes the number of faults to be considered for the respective fault model. Afterwards, the results obtained by the previously introduced methods are reported, i.e. the size of the testsets obtained by the SAT-based approach and the size of the testsets obtained by the PBO-based approach, respectively. Finally, the size of the testsets determined by the proposed approach is given in column MINATPG and the required run time is given in column TIMES. Timeouts are reported by T.O., whereas the $>$ denotes the considered testset size before the timeout occurs. This value provides a lower bound.

As can be seen from the results, minimal testsets can efficiently be obtained for circuits including approx. 100 gates. In fact, less than one second is needed for this purpose. Besides that, also larger circuits can be handled. Hence, for the first time, it was possible to generate a minimal testset of a circuit composed of more than 2 000 gates. So far, minimal testsets have been presented only for significantly smaller circuits (e.g. in [11]).

Using these results, also conclusions on the quality of other ATPG methods for reversible circuits can be made. While the size of the test patterns obtained by the SAT-based approach still is way beyond the minimum (e.g. for ham15_107, 26 test patterns are generated for the SMGF; in fact, six would be sufficient), the PBO-based approach leads to testsets which are very compact (in fact, the size of these testsets only slightly differs from the minimum).

VI. CONCLUSION

In this paper, an approach to determine minimal complete testsets for the single missing control fault model and the single missing gate fault model are introduced, respectively. The approach iteratively checks for a testset with a certain size. If there is no testset with the considered number of test pattern, the approach continues with one more test pattern. Finally, by iteratively incrementing the number of test patterns, a minimal complete testset is eventually be determined.

The experimental evaluation shows that the approach is able to handle circuits with more than 2 000 gates for both considered fault models. If the approach aborts due to limited computational resources, a lower bound for the size of the minimal complete testset is provided.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their helpful comments. This work was supported by the German Research Foundation (DFG) (DR 287/20-1).
## Table I

### Result for Minimal Complete Testsets for SMCF and SMGF

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4gtl-v0_78</td>
<td>13</td>
<td>5</td>
<td>1</td>
<td>18</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>0.91</td>
<td>13</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>0.39</td>
</tr>
<tr>
<td>4gtl-v0_86</td>
<td>14</td>
<td>5</td>
<td>1</td>
<td>20</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>0.88</td>
<td>14</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.46</td>
</tr>
<tr>
<td>dec24-enable_32</td>
<td>14</td>
<td>9</td>
<td>6</td>
<td>17</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0.13</td>
<td>14</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.47</td>
</tr>
<tr>
<td>mod51_16</td>
<td>15</td>
<td>8</td>
<td>3</td>
<td>19</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>0.77</td>
<td>15</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0.28</td>
</tr>
<tr>
<td>4_49_16</td>
<td>16</td>
<td>4</td>
<td>0</td>
<td>24</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>0.77</td>
<td>16</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>0.28</td>
</tr>
<tr>
<td>miller_5</td>
<td>16</td>
<td>8</td>
<td>5</td>
<td>24</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>0.7</td>
<td>16</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1.14</td>
</tr>
<tr>
<td>1_17_6</td>
<td>17</td>
<td>7</td>
<td>4</td>
<td>20</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>0.74</td>
<td>17</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>0.53</td>
</tr>
<tr>
<td>mini-alu</td>
<td>20</td>
<td>10</td>
<td>6</td>
<td>27</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>0.74</td>
<td>20</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>0.63</td>
</tr>
<tr>
<td>rd53_131</td>
<td>28</td>
<td>7</td>
<td>2</td>
<td>24</td>
<td>11</td>
<td>11</td>
<td>9</td>
<td>1.85</td>
<td>28</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>0.44</td>
</tr>
<tr>
<td>rd84_142</td>
<td>28</td>
<td>15</td>
<td>7</td>
<td>49</td>
<td>16</td>
<td>8</td>
<td>7</td>
<td>0.95</td>
<td>28</td>
<td>9</td>
<td>4</td>
<td>3</td>
<td>0.55</td>
</tr>
<tr>
<td>sym6_63</td>
<td>29</td>
<td>14</td>
<td>8</td>
<td>43</td>
<td>11</td>
<td>7</td>
<td>6</td>
<td>0.42</td>
<td>29</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>0.42</td>
</tr>
<tr>
<td>4_49_7</td>
<td>42</td>
<td>15</td>
<td>11</td>
<td>61</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>0.83</td>
<td>42</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>0.13</td>
</tr>
<tr>
<td>ham15_106</td>
<td>70</td>
<td>15</td>
<td>0</td>
<td>125</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>1.17</td>
<td>70</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>0.98</td>
</tr>
<tr>
<td>hwb5_13</td>
<td>88</td>
<td>28</td>
<td>23</td>
<td>131</td>
<td>11</td>
<td>7</td>
<td>4</td>
<td>0.46</td>
<td>88</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>1.10</td>
</tr>
<tr>
<td>ham15_109</td>
<td>109</td>
<td>15</td>
<td>0</td>
<td>126</td>
<td>8</td>
<td>9</td>
<td>6</td>
<td>0.35</td>
<td>109</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>0.52</td>
</tr>
<tr>
<td>ham15_107</td>
<td>132</td>
<td>15</td>
<td>0</td>
<td>352</td>
<td>25</td>
<td>16</td>
<td>12</td>
<td>760.63</td>
<td>132</td>
<td>26</td>
<td>7</td>
<td>6</td>
<td>0.59</td>
</tr>
<tr>
<td>hwb6_14</td>
<td>159</td>
<td>46</td>
<td>40</td>
<td>241</td>
<td>13</td>
<td>7</td>
<td>6</td>
<td>0.71</td>
<td>159</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>0.48</td>
</tr>
<tr>
<td>ex5p</td>
<td>647</td>
<td>206</td>
<td>198</td>
<td>904</td>
<td>19</td>
<td>18</td>
<td>12</td>
<td>T.O.</td>
<td>647</td>
<td>24</td>
<td>11</td>
<td>9</td>
<td>22.85</td>
</tr>
<tr>
<td>spu</td>
<td>1709</td>
<td>489</td>
<td>473</td>
<td>2711</td>
<td>42</td>
<td>19</td>
<td>13</td>
<td>T.O.</td>
<td>1709</td>
<td>34</td>
<td>18</td>
<td>12</td>
<td>2088.79</td>
</tr>
<tr>
<td>alu</td>
<td>2186</td>
<td>541</td>
<td>527</td>
<td>3390</td>
<td>38</td>
<td>18</td>
<td>12</td>
<td>1978.12</td>
<td>2186</td>
<td>40</td>
<td>12</td>
<td>10</td>
<td>1802.56</td>
</tr>
</tbody>
</table>

**CIRCUIT**: name of the circuit  
**d**: number of gates  
**n**: number of lines  
**c**: number of constant inputs  
**$\lvert F \rvert$**: number of faults to be tested  
**SAT**: number of test patterns obtained by SAT-based ATPG  
**PBO**: number of test patterns obtained by PBO-based ATPG  
**MINATPG**: number of test patterns obtained by proposed approach  
**TIME**: required run-time in CPU seconds for the proposed approach

## References