Improved Fault Diagnosis for Reversible Circuits

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

***Abstract*—Reversible circuits rely on an entirely different computing paradigm allowing to perform computations not only from the primary inputs to the primary outputs but also vice versa. Recently, first physical realizations based on this paradigm have been introduced in the domain of quantum computation and low-power circuits. This puts key test challenges for the future on the table. While first steps towards testing such circuits have been made (e.g. fault models and appropriate ATPG methods have been introduced), fault diagnosis has hardly been considered so far.**

In this paper, we consider the application of fault diagnosis methods for reversible circuits. In particular, we propose a new fault diagnosis approach which explicitly exploits the advantageous properties of reversible circuits. Experiments show that even though conventional methods can be applied to reversible circuits, improvements of more than one order of magnitude are achieved if reversibility is explicitly exploited.

I. Introduction

New technologies will emerge in the near future. While some of them are advancements of existing approaches, other technologies are tending towards entirely different computing paradigms. Reversible circuits [1], [2] are one of the latter developments. Here, computations are performed in an invertible manner, i.e. not only from the primary inputs to the primary outputs but also vice versa. This new paradigm is required by several emerging technologies where, currently, applications within the domain of quantum computation [3], [4], [1] and low-power design [5], [6] are seen as the most promising ones.

While reversible logic has been considered for decades (see e.g. [5], [7], [3]), recently it got a fresh start by the introduction of first physical realizations of computing machines based on this paradigm. For example, in [8] a first (small) quantum circuit (which inherently is reversible) was introduced which is able to solve the factorization problem in polynomial time; for conventional circuits only exponential solutions are available. In [6], a first reversible circuit based on conventional CMOS technology has been presented.

Even if all this still is basic research, these advancements put key test challenges for the future on the table. Accordingly, first fault models [9] as well as methods for *Automatic Test Pattern Generation* (ATPG) [10], [11], [12] have been introduced in the recent years. In comparison to conventional circuits, the new computation paradigm causes thereby certain difficulties, but also enables simplifications. For example, fanout and feedback are not directly allowed in reversible logic. This makes the design of reversible circuits harder and requires alternative design methods (e.g. [13], [14], [15], [16]). In contrast, controllability and observability of the respective circuits usually are very good which generally makes test pattern generation for reversible circuit easy.

However, while much progress has been made in the last years in the development of appropriate design and test methodologies, *fault diagnosis* of reversible circuits has hardly been considered so far. To the best of our knowledge, only preliminary approaches and results are available [17], [18]. Here, either simulation-based methods or so called *fault tables* are applied. In both cases, all possible assignments to the circuit under test are considered in order to detect the reason for a faulty behavior. However, since the number of simulations or the size of these fault tables increases exponentially, these methods are only applicable for very small circuits. Beyond that, first approaches for error debugging have been introduced [19], [20]. Although this is related to diagnosis, a different problem is addressed there. In contrast, efficient approaches for diagnosis of conventional circuits have been introduced in the past [21], [22], [23], [24]. But these have neither been applied to reversible circuits before nor have these been optimized for this kind of circuits.

In this work, we consider the application of conventional fault diagnosis methods for reversible circuits. We show that already this leads to a fault diagnosis flow which is applicable to larger circuits. Furthermore, we propose a new fault diagnosis approach which explicitly exploits the advantageous properties of reversible circuits. Therefore, satisfiability solvers and a new structural analysis is utilized. Our experimental results show that by exploiting reversibility, diagnosis can be performed significantly faster than by the pure application of the conventional methods. In the best case, run-time improvements of more than one order of magnitude can be achieved.

The remainder of this paper is structured as follows: Section II briefly introduces the required basics on reversible circuits, testing of reversible circuits, as well as the utilized core techniques. Afterwards, conventional fault diagnosis is reviewed and already illustrated by means of reversible circuits in Section III. Improvements exploiting reversibility during the fault diagnosis are presented in Section IV. Finally, experimental results are documented in Section V and conclusions are drawn in Section VI, respectively.

II. Background

A. Reversible Circuits

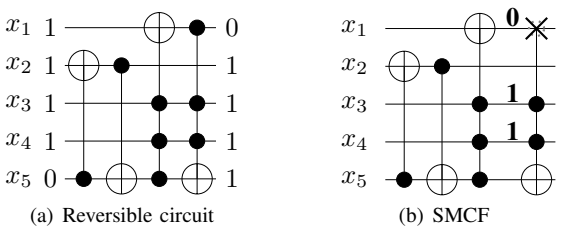
Reversible logic realizes bijective functions $f : \mathbb{B}^n \to \mathbb{B}^n$ using *reversible circuits*. A reversible circuit G is a cascade of reversible gates g_i, i.e. $G = g_1 g_2 \dots g_d$. The number d of gates is considered as the *circuit size*. In this work, we consider *Multiple Control Toffoli gates*, in the following called *Toffoli gate* for short.

Definition 1. *A Toffoli gate over the set of circuit lines $X = \{x_1, \dots, x_n\}$ has the form $g(C, x_t)$, where $C \subset X$ 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*

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

That is, if all control line variables x_c are assigned to 1, the target line x_t is inverted. Under this assignment the gate is called *activated*. All other input values x_k with $x_k \in X \setminus \{x_t\}$ pass the gate unaltered. Note that the set of control lines may be empty.

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, gates g_2 and g_3 are activated.*



(a) Reversible circuit (b) SMCF

Fig. 1. Reversible circuit with single missing control fault

B. Testing of Reversible Circuits

As it is the case for conventional circuits, physical faults of reversible circuits are abstracted by means of different fault models. In the past, several fault models considering different physical realizations have been introduced (see e.g. [9]). Most of them require to set a certain assignment to respective gates in order to detect a fault. The precise nature of the assignment differs with respect to the considered fault model. Due to page limitation, we only consider *Single Missing Control Faults* (SMCFs) in the following. However, approaches proposed in this work can easily be extended to further single fault models just by adjusting the required assignment to the respective gates.

Definition 2. *Let $g(C, x_t)$ be a Toffoli gate of a circuit G. Then, a single missing control fault occurs if instead of g, a gate $g^{'}(C^{'}, x_t)$ with $C^{'} = C \setminus \{x_i\}$, $x_i \in C$, and $x_i \neq x_t$ is executed, i.e. an SMCF occurs if a control line is missing.*

In order to detect a fault, the respective gates have to be activated so that the faulty behavior can be observed at the outputs of the circuit. In case of an SMCF, this requires an input assignment defined as follows.

Definition 3. *Let $g(C, x_t)$ be a Toffoli gate of a circuit G. In order to to detect an SMCF in gate g, all control lines in C (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.*

Example 2. *Fig. 1(b) illustrates an SMCF, which can occur in the reversible circuit previously introduced in Fig. 1(a). The respective assignment needed to detect this fault is also given.*

C. SAT and PBO

Solvers for *Boolean Satsifiability* (SAT) and *Pseudo-Boolean Optimization* (PBO) are core technologies utilized in this work. The underlying problems are defined as follows.

Definition 4. *Let $\Phi : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean function. Then, the SAT problem is to determine an assignment to all variables of Φ such that Φ evaluates to 1 or to prove that no such assignment exists. The function Φ is thereby given in* Conjunctive Normal Form *(CNF). Each CNF is a set of clauses where each clause is a set of literals and each literal is a propositional variable or its negation.*

Definition 5. *The* pseudo-Boolean optimization problem *determines a satisfying solution for a pseudo-Boolean function $\Psi : \{0,1\}^n \rightarrow \{0,1\}$ which, at the same time, minimizes an objective function $\mathcal{O}$. The* pseudo-Boolean function *Ψ is thereby a conjunction of constraints defined by $\sum_{i=1}^{n} c_i \dot{x}_i \geq c_n$, where $c_1 \ldots, c_n \in \mathbb{Z}$ and $\dot{x}_i$ either is a positive or a negative literal. The* objective function *$\mathcal{O}$ is defined by $\mathcal{O}(x_1, \ldots, x_n) = \sum_{i=1}^{n} m_i \dot{x}_i$ with $m_1, \ldots, m_n \in \mathbb{Z}$.*

Example 3. *Let $\Phi = (x_1 + x_2 + \overline{x}_3)(\overline{x}_1 + x_3)(\overline{x}_2 + x_3)$. Then, $x_1 = 1, x_2 = 1$, and $x_3 = 1$ is a satisfying assignment solving the SAT problem.*

Example 4. *Let $\Psi = (2x_1 + 3x_2 + \overline{x}_3 \geq 3)(2x_1 + x_2 \geq 2)$ and $\mathcal{O} = x_1 + x_2 + x_3$. Then, $x_1 = 1, x_2 = 0$, and $x_3 = 0$ is a solution to the PBO problem, satisfying Ψ and, at the same time, minimizing $\mathcal{O}$.*

Both, SAT and PBO, are well investigated problems. In the past efficient solving algorithms (so called *SAT solvers* or *PBO solvers*, respectively) have been proposed (see e.g. [25], [26]). Instead of simply traversing the complete space of assignments, intelligent decision heuristics, powerful learning schemes, and efficient implication methods are thereby applied. In the following, we apply these techniques as black boxes delivering the solution for the proposed problem formulations.

III. Applying Conventional Fault Diagnosis to Reversible Circuits

If a given circuit is faulty, test engineers are interested in the particular reason for that fault. Therefore, first a fault model is assumed. Afterwards, it is tried to narrow down the reason for the faulty behaviour to a certain fault location based on the fault model. From that fault location, the corresponding physical fault can be isolated. In order to narrow down the fault, *fault diagnosis* is applied beforehand, i.e. all available test patterns are applied to the circuit under test. Based on the respective responses in the presence of the fault (the *fault signatures*), fault locations not responsible for the faulty behaviour can be excluded. This process is continued until a unique fault location is determined. Therefore, sometimes new patterns distinguishing two faults may be needed or faults have to be classified to be equivalent. This is done using methods for *Diagnostic Test Pattern Generation* (DTPG) and *fault equivalence checking*. In the following, the resulting flow (based on [21]) is reviewed and, for the first time, applied to reversible circuits.

The general idea is thereby as follows: For each test pattern of a given test set, the corresponding fault signatures for each possible fault are determined and stored in a *fault dictionary*. In order to determine the respective signatures, fault simulation is applied for each fault as well as for each test pattern, respectively. Afterwards, the same test patterns are applied to the circuit under test. The responses are compared to each entry in the fault dictionary. All faults that lead to different signatures are excluded from further consideration.

Example 5. *Fig. 2 shows a reversible circuit with eight possible faults under the SMC fault model. In order to denote the respective faults, the notation (j, i) is used whereby j denotes the faulty gate and i is the position of the missing control line. The top of Fig. 3 shows a corresponding fault dictionary obtained by using a test set composed of the two test patterns 1010 and 0100. The dictionary is thereby constructed as a diagnostic tree. Faults detected by the same signatures are grouped together.*

Initially, all possible faults are under consideration. However, already applying the test pattern 1010 to the circuit under test and examining the respective responses leads to five subgroups narrowing down the reason for the faulty behavior. For example, in case the input assignment 1010 leads to the output response 1111, $(4, 2)$ is the sought fault since all other faults would lead to a different response. In contrast, if the output response 1010 is observed, three faults (namely $(1, 1)$, $(2, 3)$, and $(5, 1)$) still need to be considered. In order to further refine these results, different test patterns have to be applied.

A problem that may arise during fault diagnosis is that the given set of test patterns is not sufficient in order to distinguish all faults. Then, further test patterns have to be generated. Therefore, dedicated methods for DTPG are applied which determine not an arbitrary test pattern, but a pattern which

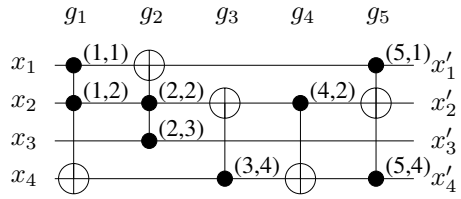


Fig. 2. A example reversible circuit

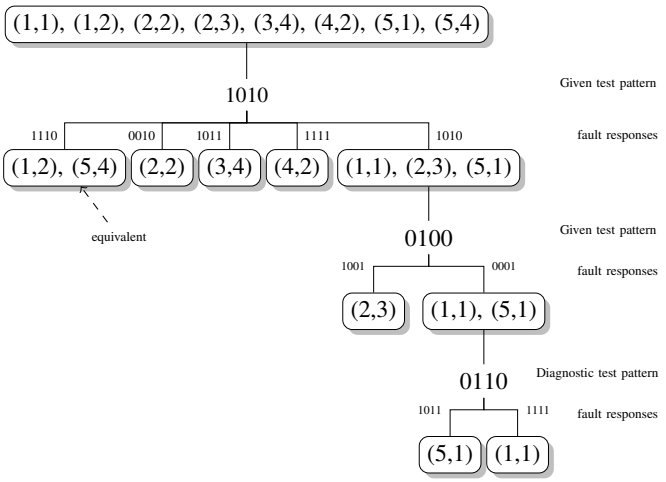


Fig. 3. Diagnostic tree

distinguishes a fault from another. Most of these methods are based on common ATPG approaches where additionally diagnostic capabilities are considered.

Example 6. *Consider again the circuit and the fault dictionary from Fig. 2 and Fig. 3, respectively. Applying the two given test patterns 1010 and 0100 is not sufficient to distinguish the faults* $(1,1)$ *and* $(5,1)$*. Thus, another test pattern (namely 0110) is obtained by means of DTPG methods. Using this test pattern, both faults can be distinguished.*

Sometimes not all faults are distinguishable from each other. In fact, two faults are said to be *equivalent*, iff there does not exist a pattern which leads to different output responses for the respective faults. All equivalent faults are grouped together to an *equivalent fault class*. Being able to prove that there is no test pattern which distinguishes between two faults is crucial. From such a result it can be concluded that, in case a faulty behavior has been narrowed down to such a fault, all faults in the equivalence class need to be considered. Structural analysis [21] often is applied for this purpose. However, these approaches are incomplete. Thus, functional analysis [22], [23], [24] additionally have to be applied which might lead to exponential run-time in the worst case.

Example 7. *Consider again the circuit and the fault dictionary from Fig. 2 and Fig. 3, respectively. It can be proven that the faults* $(1,2)$ *and* $(5,4)$ *are equivalent, i.e. there does not exist a test pattern which distinguishes these two faults. That is, if the input assignment 1010 leads to the output response 1110, no further refinement is possible and both faults need to be considered in order to detect the reason for the faulty behaviour.*

IV. Improved Fault Diagnosis for Reversible Circuits

While all the approaches introduced in the previous section are fully applicable to reversible circuits (as illustrated in the example), they do not exploit the advantageous properties of reversible circuits. As a consequence, there is room for improvement if fault diagnosis is considered for reversible circuits. This section introduces new methods for DTPG and the fault equivalence checking which make use of these possibilities. Afterwards, these methods are combined leading to an improved fault diagnosis flow for reversible circuits.

A. Improved Diagnostic Test Pattern Generation

One of the most advantageous properties of reversible circuits with respect to test purposes is their very good *controllability* and *observability*. In fact, generating a test pattern detecting a certain fault is very easy. Therefore, just an assignment activating the considered fault has to be applied to the respective gate (as e.g. done in Example 2 in Section II). Then, the respective test pattern (response) can be derived by simulating this assignment backwards to the primary inputs of the circuit (forwards to the primary outputs of the circuit)[1].

This improved controllability/observability can also be exploited if a test pattern should be created which distinguishes faults in a given group. Therefore, again assignments activating certain faults are considered. More precisely, all possible assignments activating at least one of the faults in a certain group are considered (without explicitly enumerating them). Based on them, a test pattern is determined which detects at least one, but as few as possible other faults of the considered group. This can be formulated as a PBO instance.

Therefore, two main aspects have to be encoded, namely

- the functionality of the given circuit and
- the desired diagnostic conditions.

The circuit can be encoded as introduced in [11] for the purpose of (non-diagnostic) SAT-based ATPG. Here, for a circuit G with n lines and d gates, variables $\vec{x}^j = x_n^j, x_{n-1}^j \dots x_1^j$ with $1 \leq j \leq d+1$ are introduced representing the assignment to the primary inputs (for $j = 1$), the primary outputs (for $j = d+1$), as well as the inputs and outputs of the gates (for $2 \leq j \leq d$), respectively. Furthermore, constraints ensuring the functionality of the respective gates of G are added. For example, the constraint

$$x_i^{j+1} = x_i^j \oplus \bigwedge_{x_c \in C_j} x_c$$

is added in order to ensure the correct input/output mapping of the target line of a gate $g_j(C_j, x_i)$. Similar constraints are added to encode the remaining cases and, if applicable, additional constraints. For more details on the encoding of the circuit, we refer to [11].

Having an encoding for the circuit's functionality, further variables and constraints are introduced in order to encode the diagnostic conditions. Therefore, all faults in the considered group are represented by new variables $F = \{f_1, \dots, f_k\}$, whereby k denotes the number of faults in that group. Furthermore, constraints are added ensuring that each variable $f_l \in F$ is set to 1 if an input assignment is applied activating the corresponding fault; otherwise, f_l is set to 0. For example, a missing control fault at the m^{th} line in a gate $g_j(C_j, x_i)$ with $m \neq i$ is represented by a variable f_l and the following constraint:

$$f_l = (x_m^j = 0) \wedge \Big(\bigwedge_{x_c \in C_j \setminus \{x_m^j\}} x_c = 1 \Big)$$

[1]If additional constraints (e.g. constant inputs) need to be considered, alternatives to simulation are available [11].

Functionality of the circuit:
$x_1^2 = x_1^1$, $x_2^2 = x_2^1$, $x_3^2 = x_3^1$, $x_4^2 = x_4^1 \oplus (x_1^1 \wedge x_2^1)$
...
$x_1^6 = x_1^5$, $x_3^6 = x_3^5$, $x_4^6 = x_4^5$, $x_2^6 = x_2^5 \oplus (x_1^5 \wedge x_4^5)$
Activation of faults $(1,1)$ *and* $(5,1)$*:*
$f_1 = \overline{x_1^1} \wedge x_2^1$ *for* $(1,1)$
$f_2 = \overline{x_1^5} \wedge x_4^5$ *for* $(5,1)$
At least one fault should be detected: $f_1 \vee f_2 = 1$
Objective function: $\mathcal{O} = f_1 + f_2$

Fig. 4. Encoding of the DTPG problem for the the circuit from Fig. 2

Corresponding encodings are possible for other fault models.

Using the variables and constraints introduced so far, the PBO solver will return arbitrary assignments to all variables $\vec{x}^j$ with $1 \leq j \leq d+1$ representing valid signal assignments of the circuit. However, by further restricting the assignment to the variables $f_l \in F$, the determination of such signal assignments can explicitly be controlled. In the following, this is done in order to generate a diagnostic test pattern.

First, only assignments should be allowed which detect at least one of the considered faults, i.e.

$$(\bigvee_{f_l \in F} f_l) = 1$$

is added. Second, the assignment to be determined should detect as few as possible of the other faults, i.e. the following objective function should be minimized:

$$\mathcal{O} = \sum_{l=1}^{k} f_l$$

Example 8. *In order to encode the DTPG problem for the circuit from Fig. 2 with the two faults* $(1,1)$ *and* $(5,1)$*, variables* $x_1^1, \ldots, x_4^1, \ldots, \ldots, x_1^6, \ldots, x_4^6$ *representing the assignment to the respective circuit signals as well as variables* f_1 *and* f_2 *representing the activation of faults are introduced. The respective constraints for this instance are partially shown in Fig. 4.*

The resulting instance is passed to a PBO solver which, afterwards, determines a satisfying assignment if a test pattern can be generated detecting at least one but as few as possible other faults. In this case, the respective test pattern can be obtained from the assignment to all variables $x_n^1, \ldots, x_1^1$. If no satisfying solution can be determined (i.e. if the PBO solver returns unsatisfiable), all the faults in F have been proven to be untestable (usually sorted out prior to fault diagnosis).

Example 9. *Consider again the instance from Fig. 4. One satisfying solution to this instance leads to the test pattern 0110 (obtained from the assignment* $x_1^1 = 0$*,* $x_2^1 = 1$*,* $x_3^1 = 1$*, and* $x_4^1 = 0$*). This test pattern detects the fault* $(1,1)$*, but not the fault* $(5,1)$ *in the circuit of Fig. 2.*

While this encoding offers an improved alternative for DTPG of reversible circuits, it does not entirely solve the fault equivalence checking problem. Therefore, a further check is necessary which is introduced in the next section.

B. Improved Fault Equivalence Checking

In most of the cases, the DTPG approach introduced in the last section generates a test pattern which detects at least one but as few as possible other faults of a considered group. However, sometimes only a test pattern can be determined which detects *all* faults. Then, it remains unclear whether this test pattern distinguishes at least one fault or not. In fact, a single test pattern still can distinguish two faults if it leads to different responses at the primary output[2].

[2]This is also the reason why in the proposed DTPG approach a test pattern is determined which detects as few as possible other faults and not a test pattern which does not detect at least one of the other faults.

Whether the test pattern obtained by DTPG distinguishes at least one fault or not can easily be checked by fault simulation. If this simulation leads to the same responses for all faults, another test pattern distinguishing the faults might exist. The proposed fault equivalence checking method either determines such a test pattern (showing that the faults are not equivalent and providing a new diagnostic test pattern) or proves that no such test pattern exists (showing that the faults are equivalent). Therefore, we apply an adjusted structural analysis which also exploits the good observability of reversible circuits. The general idea is based on the following observations.

Without loss of generality, assume that just two missing control faults f_1 (occurring in gate g_i) and f_2 (occurring in gate g_j with $i < j$) have to be distinguished. Furthermore, no test pattern exists which solely detects either f_1 or f_2 (such a test pattern would have been determined during DTPG). In case of fault f_1, the faulty behavior is introduced in the target line of gate g_i and propagated towards the primary outputs. The faulty behavior can only be detected by a flip of at least one primary output value (compared to the expected fault-free value). In order to distinguish fault f_1 from f_2, it must be possible to distinctively retrace at least one of the "output-flips" to f_1. This is only possible, if fault f_1 is propagated to at least one circuit line which neither influences the target line nor any of the control lines of g_j. Otherwise, the "output-flips" could also be caused by f_2.

Example 10. *Consider again the circuit from Fig. 2 as well as the two faults* $f_1 = (1,2)$ *in gate* g_1 *and* $f_2 = (5,4)$ *in gate* g_5*. The DTPG approach already confirmed that no test pattern exists which solely detects one of these faults. Furthermore, a structural analysis shows that, in case of fault* f_1*, the faulty behavior is always propagated through the forth circuit line (here, the faulty behavior is introduced) and the second circuit line (propagated by gate* g_3*), i.e. "output-flips" can only be detected at these two lines.*

However, the second circuit line is the target line of g_5 *where also the faulty behavior of* f_2 *would be introduced. Furthermore, the forth circuit line is a control line of* g_5*. Since* f_1 *and* f_2 *are always detected by the same test patterns and, additionally, those test patterns always require a certain assignment to the control lines of* g_5 *in order to activate* f_2*, also the forth line leads to an indistinguishable response. Thus, it is impossible to distinguish whether the faulty behavior is caused by* f_1 *or* f_2*. Both faults are equivalent.*

Based on these observations, two faults f_1 and f_2 can easily be classified to be fault equivalent. Therefore, just a simple structural analysis has to be performed. If this analysis shows that there is at least one line propagating the fault f_1 and neither influencing the target line nor any of the control lines of gate g_j, then it has to be checked if a test pattern can be determined which enables to propagate the faulty behavior to this line. This is illustrated by the following example.

Example 11. *Consider the circuit shown in Fig. 5 and the two faults* $f_1 = (1,3)$ *in gate* g_1 *and* $f_2 = (5,5)$ *in gate* g_5*. The DTPG approach already confirmed that no test pattern exists which solely detects one of these faults. The structural analysis shows that there is one circuit line, namely the fourth line, which propagates the fault* f_1*, but does not influence the target line nor any of the control lines of gate* g_5*. However, in order to propagate that fault through the fourth circuit line, gate* g_2 *needs to be activated. Therefore, a test pattern needs to be determined which detects* f_1 *and additionally sets all control lines of* g_2 *to one.*

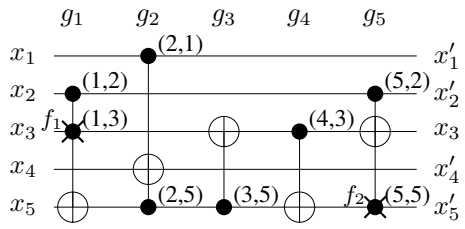


Fig. 5. Example circuit for proof of fault equivalence

In order to generate such a test pattern, a SAT solver is utilized. Therefore, an instance as described in the previous section is generated (of course, without the objective function $\mathcal{O}$). Additionally, the constraint

$$\bigwedge_{x_c \in C_{act}} x_c = 1$$

with $g(C_{act}, t)$ being the gate to be activated is added ensuring that g indeed propagates the faulty behavior. If the SAT solver returns a satisfying assignment, then a test pattern can be obtained which distinguishes f_1 and f_2. If the instance is unsatisfiable, it has to be checked if further gates and circuit lines, respectively, exist through which the faulty behavior of f_1 can be propagated. If for all these cases no test pattern could be obtained, f_1 and f_2 are proven to be fault equivalent.

The structural analysis is thereby very efficient. In fact, only linear time with respect to the number of gates the circuit is composed of is needed. In contrast, the additional ATPG runs may require some time. However, these checks are hardly necessary. In almost all cases, either DTPG or the structural analysis lead to a result beforehand. That is, combining the improved DTPG and fault equivalence checking methods, an improved fault diagnosis flow results as summarized in the following and experimentally evaluated in Section V.

C. Resulting Fault Diagnosis Flow

Fig. 6 shows the proposed fault diagnosis flow for reversible circuits incorporating the methods introduced in the previous sections. First, the test engineer provides a fault model as well as an initial test set (a). If this test set already distinguishes all faults based on the assumed model, a fault dictionary is returned and the fault diagnosis terminates (b). Otherwise, the proposed DTPG method from Section IV-A is applied to a group of faults which are still not distinguished (c). If a (diagnostic) test pattern distinguishing at least one fault returns, it is added to the test set (d). Otherwise, fault equivalence checking as proposed in Section IV-B is invoked. That is, the structural analysis (e) checks whether the respective faults can be classified to be equivalent or not. If they are equivalent, they are added to the fault equivalence class (f). Otherwise, it is checked whether a test pattern can be determined which satisfies the restrictions as discussed in Section IV-B (g). If this was successful, the resulting (diagnostic) test pattern is added to the test set (d). Otherwise, the faults have been proven to be equivalent and, thus, they are added to the fault equivalence class (f). This process continues until all faults either have been distinguished or classified to be fault equivalent.

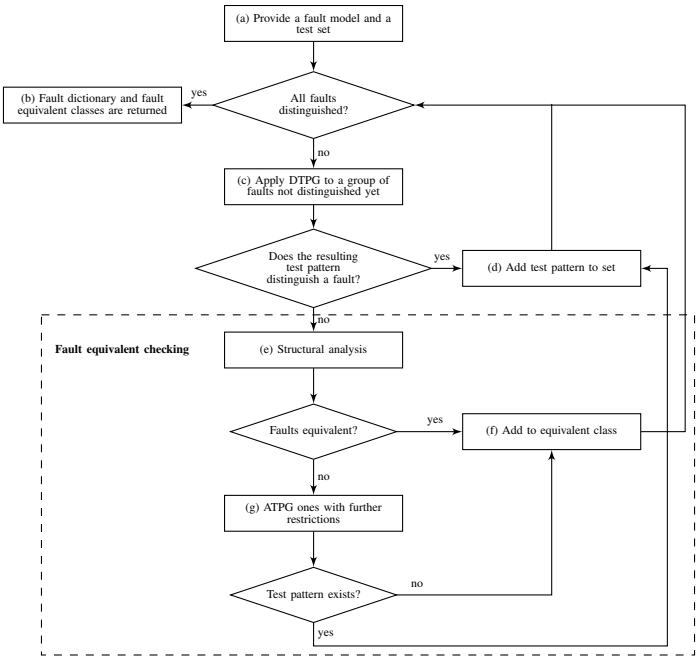


Fig. 6. The proposed fault diagnosis flow

V. Experimental Results

The proposed approaches have been implemented in C++ on top of RevKit [27]. As underlying solving engines, *clasp* [26] (for DTPG) and *MiniSAT* [25] (for the additional ATPG runs during fault equivalence checking) are applied. The performance of the approach proposed in Section IV has been evaluated on a set of benchmarks circuits taken from RevLib [28]. For comparison, these circuits have additionally been diagnosed using conventional methods based on the concepts introduced in [24] and reviewed in Section III. Due to page limitation, only single missing control faults are considered in the following. However, as briefly discussed in Section II-B, the approaches proposed in this work can easily be extended to further fault models just by adjustments of certain assignment conditions. All experiments have been carried out on an AMD Opteron $\times 4$ processor with 8GB main memory running Linux. The timeout was set to 5 000 CPU seconds.

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 d of gates, (3) the number n of circuit lines, and (4) the number c of constant inputs. The column $|\mathcal{F}|$ denotes the number of faults to be considered. Afterwards, the number of test patterns in the provided test set is given (*Test Set*). In our evaluation, we used complete test sets obtained by the ATPG method proposed in [11]. The following columns present the results obtained by applying the conventional diagnosis flow (from Section III) and the proposed diagnosis flow (from Section IV), respectively. More precisely, the number of obtained diagnostic test patterns (denoted by *DTP*), the number of equivalent faults (denoted by *EF*), and the run-time in CPU seconds (denoted by *Time*) is given. Finally, the improvement of the proposed diagnosis flow with respect to the run-time is provided in the last column.

First of all, it can be concluded that already the application of conventional approaches leads to a fault diagnosis flow which is applicable to larger circuits. Furthermore, applying the proposed diagnosis flow does not lead to a significant decrease of the quality of the results. In fact, the number of generated diagnostic test patterns is almost equal to the patterns generated by the conventional methods; sometimes (e.g. for *0410184_169* and *hwb9_123*) the number is even slightly better.

TABLE I
EXPERIMENTAL RESULTS

Circuit	d	n	c	$\lvert\mathcal{F}\rvert$	Test Set [11]	Conventional Diagnosis [24] DTP	*EF*	Time(s)	Proposed Diagnosis DTP	*EF*	Time(s)	Time Impr (%)
4gt4-v0_78	13	5	1	18	6	8	0	0.03	8	0	0.01	66.67
4_49_16	16	4	0	24	5	6	0	0.03	6	0	0.02	33.33
mini-alu_84	20	10	6	27	4	8	0	0.07	8	0	0.06	14.29
rd84_142	28	15	7	49	8	14	0	0.17	14	0	0.15	11.76
sym6_63	29	14	8	43	7	8	0	0.50	8	0	0.05	90.00
0410184_169	46	14	0	49	3	7	0	0.19	4	0	0.07	63.16
hwb5_13	88	28	23	131	7	10	0	0.35	10	0	0.31	11.43
ham15_107	132	15	0	352	16	53	28	43.51	70	28	8.4	80.69
hwb6_14	159	46	40	241	7	10	0	0.93	10	0	0.84	9.68
sym9_148	210	10	1	756	14	100	0	60.46	102	0	52.38	13.36
hwb8_113	637	8	0	2214	44	68	99	204.45	65	99	203.05	0.68
ex5p	647	206	198	904	18	24	0	49.95	24	0	17.63	64.70
hwb9_119	1544	9	0	5812	83	107	433	2667.42	110	433	2271.47	14.84
spla	1709	489	473	2711	19	>54	–	>5000.00	105	4	796.73	>87.69
hwb9_123	1959	9	0	3596	80	100	12	752.88	98	12	657.92	12.61
table3	1988	554	540	2997	23	>34	–	>5000.00	40	4	281.92	>94.36
alu4	2186	541	527	3390	18	>40	–	>5000.00	47	2	463.84	>93.84
ex1010	2982	670	660	4543	25	29	0	2092.82	29	0	386.06	68.32

Circuit: name of the circuit d: number of gates n: number of circuit lines c: number of constant inputs $|\mathcal{F}|$: number of faults to be tested
Test Set: number of test patterns in the provided test set DTP: number of diagnostic test patterns obtained by the respective approaches
EF: number of equivalent faults Run time: run-time in CPU seconds Time Impr: improvement of the proposed diagnosis flow w.r.t. the run-time

In contrast, there are significant differences regarding the run-time. For all benchmarks, the proposed diagnosis flow clearly outperforms the application of the conventional methods. In the best case, improvements of more than one order of magnitude can be achieved. Additionally, the proposed approach scales better for larger circuits. Thus, fault diagnosis of reversible circuits clearly profits from approaches that explicitly exploit the inherent reversibility of this kind of circuits.

VI. Conclusion

In this paper, approaches for fault diagnosis have been introduced. We showed that already the application of a conventional approach leads to a fault diagnosis flow which is applicable to larger circuits. Furthermore, an improved new fault diagnosis approach has been introduced which explicitly exploits the advantageous properties of reversible circuits. Experimental results showed that this enables run-time improvements of more than one order of magnitude in the best case.

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).

References

[1] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.
[2] R. Wille and R. Drechsler, *Towards a Design Flow for Reversible Logic*. Springer, 2010.
[3] A. Peres, "Reversible logic and quantum computers," *Phys. Rev. A*, no. 32, pp. 3266–3276, 1985.
[4] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," *Foundations of Computer Science*, pp. 124–134, 1994.
[5] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Res. Dev.*, vol. 5, p. 183, 1961.
[6] B. Desoete and A. D. Vos, "A reversible carry-look-ahead adder using control gates," *INTEGRATION, the VLSI Jour.*, vol. 33, no. 1-2, pp. 89–104, 2002.
[7] T. Toffoli, "Reversible computing," in *Automata, Languages and Programming*, W. de Bakker and J. van Leeuwen, Eds. Springer, 1980, p. 632, technical Memo MIT/LCS/TM-151, MIT Lab. for Comput. Sci.
[8] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, vol. 414, p. 883, 2001.
[9] I. Polian, T. Fiehn, B. Becker, and J. P. Hayes, "A family of logical fault models for reversible circuits," in *Asian Test Symp.*, 2005, pp. 422–427.
[10] J. P. Hayes, I. Polian, and B. Becker, "Testing for missing-gate faults in reversible circuits," in *Asian Test Symp.*, 2004, pp. 100–105.
[11] R. Wille, H. Zhang, and R. Drechsler, "ATPG for reversible circuits using simulation, Boolean satisfiability, and pseudo Boolean optimization," in *IEEE Annual Symposium on VLSI*, 2011, pp. 120–125.
[12] A. Paler, A. Alaghi, I. Polian, and J. P. Hayes, "Tomographic testing and validation of probabilistic circuits," in *European Test Symp.*, 2011, pp. 63–68.
[13] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Trans. on CAD*, vol. 22, no. 6, pp. 710–722, 2003.
[14] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Trans. on CAD*, vol. 25, no. 11, pp. 2317–2330, 2006.
[15] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic for large functions," in *Design Automation Conf.*, 2009, pp. 270–275.
[16] R. Wille, S. Offermann, and R. Drechsler, "SyReC: A programming language for synthesis of reversible circuits," in *Forum on Specification and Design Languages*, 2010, pp. 184–189.
[17] K. Ramasamy, R. Tagare, E. Perkins, and M. Perkowski, "Fault localization in reversible circuits is easier than for classical circuits," in *Workshop on Logic and Synthesis*, 2004.
[18] D. Pierce, J. Biamonte, and M. Perkowski, "Test set generation and fault localization software for reversible circuits," in *Int'l Symp. on Representations and Methodologies for Emergent Computing Technologies*, September 2005.
[19] R. Wille, D. Große, S. Frehse, G. W. Dueck, and R. Drechsler, "Debugging of Toffoli networks," in *Design, Automation and Test in Europe*, 2009, pp. 1284–1289.
[20] J. C. Jung, S. Frehse, R. Wille, and R. Drechsler, "Enhancing debugging of multiple missing control errors in reversible logic," in *Great Lakes Symp. VLSI*, 2010, pp. 465–470.
[21] M. Abramovici, M. Breuer, and A. Friedman, *Digital Systems Testing and Testable Design*. IEEE Press, 1990.
[22] M. E. Amyeen, W. K. Fuchs, I. Pomeranz, and V. Boppana, "Fault equivalence identication using redundancy information and static and dynamic extraction," in *IEEE VLSI Test Symposium*, 2001.
[23] V. D. Agrawal, D. H. Baik, Y. C. Kim, and K. K. Saluja, "Exclusive test and its applications to fault diagnosis," in *VLSI Design*, 2003, pp. 143–148.
[24] A. Veneris, R. Chang, M. S. Abadir, and M. Amiri, "Fault equivalence and diagnostic test generation using ATPG," in *IEEE International Symposium on Circuits and Systems*, 2004.
[25] N. Eén and N. Sörensson, "An extensible SAT solver," in *SAT 2003*, ser. LNCS, vol. 2919, 2004, pp. 502–518.
[26] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub, "Conflict-driven answer set solving," in *Int'l Joint Conference on Artificial Intelligence*, 2007, pp. 386–392.
[27] M. Soeken, S. Frehse, R. Wille, and R. Drechsler, "RevKit: A toolkit for reversible circuit design," in *Workshop on Reversible Computation*, 2010, pp. 69–72, RevKit is available at http://www.revkit.org.
[28] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: an online resource for reversible functions and reversible circuits," in *Int'l Symp. on Multi-Valued Logic*, 2008, pp. 220–225, RevLib is available at http://www.revlib.org.