On the Relation Between Simulation-based and SAT-based Diagnosis

Görschwin Fey[1]   Sean Safarpour[2]

[1]Institute of Computer Science
University of Bremen
28359 Bremen, Germany
{fey,drechsle}@informatik.uni-bremen.de

Andreas Veneris[2]   Rolf Drechsler[1]

[2]Dep. of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario M5S 3G4, Canada
{sean,veneris}@eecg.toronto.edu

Abstract

The problem of diagnosis – or locating the source of an error or fault – occurs in several areas of computer aided design, such as dynamic verification, property checking, equivalence checking and production test. Manually locating errors can be a time consuming and resource-intensive process. Several automated approaches for diagnosis have been presented, among them are simulation-based and SAT-based techniques. These two approaches are found to be robust even for large circuits as well as being applicable to a broad range of diagnosis problems. An in-depth comparison of both approaches necessary to augment our knowledge of diagnosis procedures has not been addressed by previous work.

This paper provides a thorough analysis of the similarities and differences between simulation-based and SAT-based procedures for diagnosis. The relation between the basic approaches is theoretically analyzed. Issues regarding performance and diagnosis quality (resolution) are discussed. Experimental data strengthens the theoretical results. This detailed understanding of the relations between the techniques is necessary to provide further improvements to the field of diagnosis. The initial steps towards building a hybrid technique are also presented.

1 Introduction

Coping with the size and complexity of todays circuits and systems is only possible with *Computer Aided Design (CAD)* techniques. Many sophisticated algorithms and techniques exist that ensure the correctness of a design. These methods include efficient simulation and formal verification – in form of property checking and equivalence checking – as well as post-production test of a design. Although effective, these techniques only detect the existence of errors or faults. Further effort is required to locate the source of the errors. Unfortunately, manually finding or diagnosing the error locations in a design is a time consuming and therefore costly task. Automatic approaches for diagnosis have been proposed to speed up this process.

The structural approaches [12] and the BDD-based approaches [6, 8] have certain draw-backs. Structural approaches rely on similarities between the erroneous circuit – the implementation – and the specification. But such similarities may not be present, e.g. due to optimizations during synthesis. For large designs BDD-based approaches suffer from space complexity issues.

Here, diagnosis methods that use a set of test-vectors are considered. We focus on approaches for simulation-based diagnosis [10, 9, 18, 13] and diagnosis based on *Boolean Satisfiability* (SAT) [4, 3]. Both approaches have been applied to combinational and sequential diagnosis problems. Due to the underlying engines, both techniques are robust with respect to the size of the de-

sign. Simulation-based approaches can use efficient parallel simulation techniques with linear runtimes, while SAT-based approaches benefit from recent advances in SAT-solving [14, 15]. An in-depth analysis of these diagnosis methods can show directions for further improvements.

In this work simulation-based and SAT-based diagnosis are compared from a theoretical and empirical point of view for the first time. Both approaches use a set of test-vectors for diagnosis that may be provided after test-bench simulations, formal verification, or after failing a post-production test. The basic procedures of the two approaches are outlined in this work. The relationship between these procedures is explained by introducing a third approach of simulation-based diagnosis for multiple errors. Similarities and differences are analyzed using this third approach. The theoretical results are backed by experimental data based on the ISCAS89 benchmark suite. Overall, this work provides future research initiatives for improving each individual diagnosis technique as well as creating hybrid approaches that exploit the advantages of both.

2 Diagnosis Approaches

In this section the diagnosis problem is introduced and the basic diagnosis procedures for simulation-based and SAT-based diagnosis are presented. References for the advanced approaches which make use of the basic procedures are given in the corresponding sections. It is assumed that the reader is familiar with the representation of a circuit in *Conjunctive Normal Form* (CNF) [11], and with the concepts of modern SAT-solvers such as conflict-based learning [14] and efficient *Boolean Constraint Propagation* (BCP) [15]. Error location and fault diagnosing are similar problems as explained in [1]. Therefore, unless otherwise stated, terminology for error location will be used in the following.

2.1 Diagnosis Problem

Definition 1. *Let the circuit $\mathcal{I}$ be a faulty implementation of a specification. A test T is a triple (t, o, v), where*

- *t is a test-vector in terms of primary input values of $\mathcal{I}$.*
- *t causes an erroneous value at primary output o.*
- *v is the correct value for primary output o.*

A test-set $\mathcal{T}$ is a set of tests.

Definition 2. *Let the circuit $\mathcal{I}$ be an implementation of a specification and let $\mathcal{T}$ be a test-set of m tests.*

PathTrace($\mathcal{I}$, i, t, o)
(1) Simulate t to establish values of internal signals.
(2) Mark the gate with the erroneous primary output.
(3) For each marked gate g, that was not visited
   $\mathcal{C}_i := \mathcal{C}_i \cup g$
   If there are inputs with controlling value[1]
      mark one of these inputs,
   else // no input has a controlling value
      mark all inputs.
(4) If there remain marked gates, that have not been visited goto (3).
BasicSimDiagnose($\mathcal{I}$, $\mathcal{T}$)
(5) For $i = 1$ to m do PathTrace($\mathcal{I}$, i, t_i, o_i)

[1] E.g. the input-value 0 is a controlling value for an AND-gate. The input value 1 is non-controlling.

Figure 1. Basic simulation-based diagnosis

The diagnosis problem *is to determine a set of candidate gates* $\mathcal{C} = \{g_1, \ldots, g_c\}$ *where a correction can be applied to rectify the tests in* $\mathcal{T}$.

The size of a correction $\mathcal{C}$ is denoted by $|C|$. The size of a circuit $\mathcal{I}$ is denoted by $|\mathcal{I}|$.

Definition 3. *A set of candidate gates* $\mathcal{C}$ *is called a* valid correction *for a test-set* $\mathcal{T}$, *if changing the functionality of the gates in* $\mathcal{C}$ *is sufficient to rectify all tests in* $\mathcal{T}$.

Definition 4. *A valid correction* $\mathcal{C}$ *contains only* essential candidates, *iff for any* $g \in \mathcal{C}$: $\mathcal{C} \setminus \{g\}$ *is not a valid correction.*

The faulty circuit $\mathcal{I}$ contains p actual error sites e_1, $\ldots$, e_p. An error is considered to be the replacement of the function of a gate by another arbitrary Boolean function. Therefore, the number of valid corrections is in the order of $O(|\mathcal{I}|^p)$ [18]. In the following the term *effect analysis* means "determining whether changing the functionality of one or more internal circuit lines corrects the value of the erroneous output".

2.2 Simulation-Based Diagnosis

The basic procedure for simulation-based diagnosis approaches considered in this paper is *Path Tracing* (PT) that is derived from *critical PT* [2]. The overall flow for a naïve simulation-based diagnosis is shown in Figure 1. The procedure **BasicSimDiagnose** uses PT to calculate a set of candidates $\mathcal{C}_i$ for each triple (t_i, o_i, v_i) in the test-set. PT marks "candidate gates" on the sensitive paths leading to the erroneous output o_i. The basic algorithm does not check whether the inversion of a candidate's logic value for a particular test vector can really cause a logic value change at the erroneous output(s), i.e. no effect analysis is performed. In the following we refer to this basic simulation-based approach as BSIM.

The interpretation of the diagnosis result depends on the number of errors that are assumed to be contained in $\mathcal{I}$. If there is only a single error in the circuit the actual error site is contained in the intersection of all candidate sets, i.e. in $\bigcap_{i=1}^{m} \mathcal{C}_i$. If there are multiple errors a conservative approach has to be used: each marked gate has to be considered as a candidate. The number of tests that sensitized a particular gate $M(g_i) = |\{i : g \in \mathcal{C}_i\}|$ can be used to order the candidates. But there is no
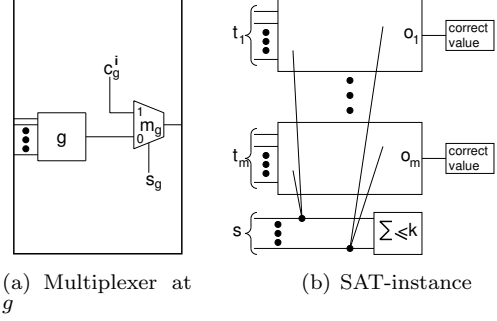

(a) Multiplexer at g (b) SAT-instance

Figure 2. SAT-based diagnosis

guarantee, that any real error site has been marked by the largest number of PT marks. Because the candidate set of each test contains at least one actual error site, at least one actual error site is marked by more than m/p tests [10], i.e. $\exists e \in \{e_1, \ldots, e_p\} : M(e_i) > m/p$. Thus, for the correction of k errors, subsets up to size k of all marked gates have to be considered. The set of marked gates is given by $\cup_1^m \mathcal{C}_i$.

This is done by the advanced simulation-based approaches relying on PT [9, 18, 13]. Multiple errors are handled by considering the corrections of size k and applying pruning techniques. E.g. in [13] the number of remaining errors is reduced by one each time in a greedy-like manner. After choosing a single correction, the candidate sets $\mathcal{C}_i$ are recalculated by calling **BasicSimDiagnose**. This effect analysis is necessary, because correcting one error may change the sensitized paths in the circuit. Then, the next single correction is chosen. But earlier decisions may have been wrong. Thus, the ability to perform a backtrack similar to the solvers for NP-complete problems is required. As a result the time complexity for the advanced simulation-based techniques drastically increases compared to BSIM.

A simulation-based approach that does not use PT has been introduced in [5]. Instead of backtracing sensitive paths an approach based on forward implications by injecting X-values was chosen for diagnosis. Therefore the core idea is similar to the approaches based on PT: The effect of changing a value at a certain position is considered.

2.3 SAT-Based Diagnosis

For SAT-based diagnosis a SAT-instance is generated that can only be satisfied, if changing a limited number of gates in the erroneous circuit produces the correct output values for all tests. This approach was first presented in [17]. The SAT-instance $\mathcal{F}$ is built as shown in Figure 2. Multiplexers are inserted at each gate g to allow for corrections (see Figure 2(a)). The output value of g is propagated when the select input s_g has the value 0. A correction is applied when the select input s_g is set to 1: the value of g is overwritten by a new unrestricted value c_g^i. Such corrections are necessary to retrieve a solution for the SAT-instance $\mathcal{F}$ shown in Figure 2(b). According to the pseudocode in Figure 3 a copy of $\mathcal{I}$ is created for each test $(t_i, o_i, v_i) \in \mathcal{T}$. Each copy is constrained to assume the primary input values of test-vector t_i and the correct output value v_i for the erroneous output o_i. The select-line s_g for multiplexers corresponding to gate g is the same in all copies

BasicSATDiagnose($\mathcal{I}$, $\mathcal{T}$, k)
(1) For each triple $(t, o, v) \in \mathcal{T}$ do
        Create an instance I of $\mathcal{I}$ in the SAT-instance.
        Constrain o to assume the correct value v.
        Constrain inputs to the values of t.
        Insert multiplexers at gates that are considered for correction.
(2) For $i = 1 \ldots k$
        Constrain the number of select-inputs with value 1 to be at most i.
        Enumerate all solutions and add a blocking clause for each solution.

Figure 3. Basic SAT-based diagnosis

of $\mathcal{I}$. Therefore the gate may be changed for all tests or for none. The injected value c_g^i may be different for different tests. Thus, gate g can be replaced by any arbitrary Boolean function. The number of gates that may be changed is bounded by constraining the number of select-lines that may assume the value 1 to be less than or equal to k.

A SAT-solver is used to solve the SAT-instance $\mathcal{F}$. Free variables in $\mathcal{F}$ are those corresponding to the select-lines s_g and to the new primary inputs for the correct values at gates c_g^i. All other variable values are determined by the constraints for the circuit's gates, the test-vectors and the correct output values. Each solution of $\mathcal{F}$ is a solution to the diagnosis problem. The select-lines that are set to 1 in a satisfying assignment for $\mathcal{F}$ determine the set of candidate gates $\mathcal{A}$ that have to be changed. In the following we refer to $\mathcal{A}$ as a solution of $\mathcal{F}$. In Figure 3 the limit is iteratively incremented in the for-loop of line (2). This guarantees, that all solutions generated by the approach only contain essential candidates, because solutions with a smaller number of candidates are blocked before increasing the limit. For this purpose an incremental SAT-solver can be used [19]. In the following we refer to this basic SAT-based approach as BSAT.

The advanced SAT-based diagnosis approach [17] applies several heuristics that improve the performance of BSAT. To reduce the search space additional clauses are added that force the free variables c_g^i to 0 when s_g is set to 0. This prevents up to $|\mathcal{I}|$ decisions of the SAT-solver. Also instead of inserting a multiplexer at each gate only dominators are selected in a first run to reduce the search space. In a second run a finer level of granularity for diagnosis can be retrieved by introducing more multiplexers in the dominated regions that may contain an error. Additionally, for a large number of tests the test-set is split into partitions to reduce the size of the SAT-instance. Finally, an all-solutions SAT-solver is used. Such a solver automatically minimizes the number of assignments in a solution. Thus, incrementally solving instances with larger limits as in the basic procedure is not necessary. These techniques do not change the solution space, but dramatically decrease the runtime. In fact, speed-up factors of more than 100 times have been observed [16]. The approach has also been applied to diagnose sequential errors efficiently [4].

3 Relation Between the Approaches

In this section the two basic diagnosis approaches BSIM and BSAT are compared from a theoretical point of view. A third approach is introduced that formally describes the application of BSIM for the diagnosis of

SCDiagnose($\mathcal{I}$, $\mathcal{T}$, k)
(1) Call **BasicSimDiagnose($\mathcal{I}$, $\mathcal{T}$, k)** to calculate $\mathcal{C}_i, 1 \leq i \leq m$.
(2) Calculate all solutions of the *set covering problem* $\mathcal{S}$: Find $\mathcal{C}^*$ such that
      (a) for each i: at least one element of $\mathcal{C}_i$ is contained in $\mathcal{C}^*$,
      (b) for any $g \in \mathcal{C}^*$: $\mathcal{C}^* \setminus g$ does not fulfill condition (a),
      (c) $|\mathcal{C}^*| \leq k$.

Figure 4. Diagnosis based on set covering

multiple errors. Using this approach the differences between the two basic techniques are explained. The discussion in Section 4 includes the advanced approaches.

The third approach is given in Figure 4. First, **BasicSimDiagnose** is called to calculate the candidate set $\mathcal{C}_i$ for each triple $(t_i, o_i, v_i) \in \mathcal{T}$. These sets form an instance $\mathcal{S}$ of the *set covering problem*. A solution $\mathcal{C}^*$ of $\mathcal{S}$ contains at least one element of each set $\mathcal{C}_i$. Thus, for each test in $\mathcal{T}$ at least one gate on a sensitized path is contained in $\mathcal{C}^*$. We refer to the approach implemented by **SCDiagnose** as COV.

Example 1. *Assume that* **SCDiagnose** *is called for $k = 2$ and a test-set with three tests. Further assume that* **BasicSimDiagnose** *returns the following candidate sets:*

$$
\begin{aligned}
\mathcal{C}_1 &= \{A, B, F, G\} \\
\mathcal{C}_2 &= \{C, D, E, F, G\} \\
\mathcal{C}_3 &= \{B, C, E, H\}
\end{aligned}
$$

Then, $\{B, D\}$ would be one possible solution returned by **SCDiagnose***. Another solution would be $\{A, D, H\}$. This simple approach does not use heuristics to bias preference to one solution over another.*

The *minimum set covering problem*, i.e. to decide if no solution with fewer elements exists, is NP-complete [7]. The relation between the set covering problem and diagnosis of multiple errors has been studied earlier, e.g. in [18].

The BSAT approach solves a very similar problem. By choosing the values of the select lines locations for corrections are determined. One difference is the simulation engine which is replaced by the BCP of the SAT-solver. Additionally, BSAT carries out an effect analysis while solving the SAT-instance: when switching a select line of a multiplexer the BCP propagates value changes dynamically. In contrast, COV does not carry out effect analysis at all. Based on these observations the following lemmas can be derived.

Lemma 1. *Let $\mathcal{I}$ be a circuit, $\mathcal{T}$ be a test-set and $k \in \mathbb{N}$. Each solution $\mathcal{A}$ of the SAT-instance $\mathcal{F}$ is a valid correction for $\mathcal{T}$.*

Proof. The construction of the SAT-instance directly implies this lemma. $\square$

Lemma 2. *Let $\mathcal{I}$ be a circuit, $\mathcal{T}$ be a test-set and $k \in \mathbb{N}$. There exist solutions for the set covering problem $\mathcal{S}$ in* **SCDiagnose($\mathcal{I}$, $\mathcal{T}$, k)** *that are not a valid correction for $\mathcal{T}$.*

Proof. Consider the circuit in Figure 5(a). A test-vector is assigned to the inputs that produces the output value 0 instead of 1. PT either marks the gates $\{A, B, D\}$ or $\{A, C, D\}$ because both inputs of D have a controlling

Table 1. Comparison of the approaches

	BSIM	COV	adv. sim.-based	BSAT	adv. SAT-based
number of candidate error sites	$O(|\mathcal{I}|)$	k, user defined (or incrementally determined)			
valid correction	not guaranteed, guides the designer		guaranteed, correct values per test are supplied		
effect analysis	none		simulation-based	inherent	
structural information	available	none for correction	available	none	exploited during CNF generation
simulation engine	efficient, circuit-based			BCP	
time complexity	$O(|\mathcal{I}| \cdot m)$	$O(|\mathcal{I}|^k)$	$O(|\mathcal{I}|^{k+1} \cdot m)$	$O(k \cdot 2^{|\mathcal{I}| \cdot m})$	$O(2^{|\mathcal{I}| \cdot m})$
size complexity	$O(|\mathcal{I}| + m)$	$O(|\mathcal{I}| \cdot m)$	$O(k \cdot |\mathcal{I}| \cdot m)$	$\Theta(|\mathcal{I}| \cdot m)$	
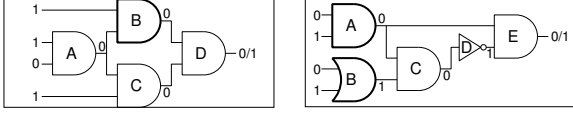
(a) COV may not provide a correction

(b) Solution for $k = 2$ by BSAT but not by COV

Figure 5. Example circuits

value. A possible solution to cover this single set of candidates is $\{B\}$ (or $\{C\}$, respectively). But the test cannot be rectified by changing only the output value of B (or C). $\qquad\square$

Lemmas 1 and 2 directly lead to Theorem 1.

Theorem 1. *Let $\mathcal{I}$ be a circuit, $\mathcal{T}$ be a test-set and $k \in \mathbb{N}$. There exist solutions calculated by* **SCDiagnose**$(\mathcal{I}, \mathcal{T}, k)$ *that are not calculated by* **BasicSATDiagnose**$(\mathcal{I}, \mathcal{T}, k)$.

Next, the capability to calculate all valid corrections is analyzed.

Lemma 3. *Let $\mathcal{I}$ be a circuit, $\mathcal{T}$ be a test-set and $k \in \mathbb{N}$.* **BasicSATDiagnose**$(\mathcal{I}, \mathcal{T}, k)$ *returns all valid corrections containing only essential candidates up to size k.*

Proof. Again, the construction of the SAT-instance directly implies this lemma. Incrementally calculating corrections of sizes 1 to k and "blocking" smaller solutions guarantees that only essential candidates are contained in each correction. $\qquad\square$

Lemma 4. *Let $\mathcal{I}$ be a circuit, $\mathcal{T}$ be a test-set and $k \in \mathbb{N}$. There are valid corrections with a size of k or less candidate gates that are not calculated by* **SCDiagnose**$(\mathcal{I}, \mathcal{T}, k)$.

Proof. Consider the circuit in Figure 5(b). Assume that only the applied test-vector shows an erroneous output value and that $k = 2$. By changing the output values of A and B, the correct output value 1 can be produced. But the single candidate set $\{A, C, D, E\}$ generated by PT does not contain B. Therefore $\{A, B\}$ is not a solution of $\mathcal{S}$. $\qquad\square$

Lemma 3 and 4 imply the following theorem.

Theorem 2. *Let $\mathcal{I}$ be a circuit, $\mathcal{T}$ be a test-set and $k \in \mathbb{N}$. There exist solutions calculated by* **BasicSATDiagnose**$(\mathcal{I}, \mathcal{T}, k)$ *that are not calculated by* **SCDiagnose**$(\mathcal{I}, \mathcal{T}, k)$.

This analysis states that neither BSIM nor COV always provide valid corrections. Furthermore these methods do not calculate the valid corrections, whereas BSAT both calculates and provides only valid corrections. This difference is critical when discussing the advanced approaches in the next section.

4 Discussion

While only the basic procedures were compared in the previous section, the following discussion also includes advanced simulation-based [9, 18, 13] and SAT-based [17] approaches. Formal aspects like the complexity of the approaches and their ability to calculate valid corrections are considered. Further issues are discussed on an informal basis. Table 1 summarizes the comparison topics and the respective results.

The *number of candidate error sites* differs between the approaches. A large number of candidates is returned by BSIM, only the number of tests that marked a particular gate may differ. In contrast the other approaches only return k candidates. The number k is small and has either to be specified by the user or is determined by automatically calculating a minimal solution. During the search subsets of the gates in $\mathcal{I}$ up to size k are considered.

When debugging the design it is important whether an approach is guaranteed to return a *valid correction*. This is not done by BSIM and COV. The solutions calculated by these basic approaches can only be used to guide the designer during error location. In contrast the advanced simulation-based approaches, the BSAT approach and the advanced SAT-based technique only return valid corrections. Additionally, with respect to each test a new value for each gate in the correction is provided. This can be exploited to determine the "correct" function of the gate.

Effect analysis guarantees that only valid corrections are calculated. The advanced simulation-based approaches rely on re-simulation while a SAT-solver inherently carries out effect analysis.

The simulation-based approaches may use *structural information* for these purposes since they are directly applied to the circuit. E.g. successor/predecessor relations, knowledge of dominators etc. can directly be exploited in the algorithms. For a SAT-based approach such information has to be encoded while generating the SAT-instance. This is not done by BSAT. But the advanced SAT-based approach in [17] uses, for example, information about structural dominators to prune the search space.

A crucial issue when considering a large number of tests is the *simulation engine*. Naturally, the simulation-based approaches can use fast engines that directly evaluate the circuit. Such an engine can also be used for what-if-analysis when carrying out effect analysis. The SAT-approach inherently uses BCP for these purposes. This may induce some overhead, when the SAT-instance is large. But due to sophisticated implementation techniques BCP is very efficient in practice [15]. Moreover, a large number of unit literals is contained in the SAT-instances. These are not further considered after the preprocessing step.

Table 2. Runtime of the basic approaches

| $\mathcal{I}$	p	m	BSIM	COV			BSAT		
				CNF	One	All	CNF	One	All
s1423	4	4	0.00	0.01	0.01	1.36	0.02	0.21	34.21
s1423	4	8	0.01	0.01	0.01	19.98	0.02	0.21	12.93
s1423	4	16	0.02	0.01	0.02	4.12	0.04	0.29	13.14
s1423	4	32	0.03	0.03	0.03	0.68	0.06	0.60	22.72
s6669	3	4	0.01	0.01	0.03	0.09	0.05	3.24	56.49
s6669	3	8	0.02	0.02	0.04	0.12	0.05	5.06	47.87
s6669	3	16	0.03	0.04	0.05	0.7	0.08	10.48	12.06
s6669	3	32	0.10	0.10	0.12	0.65	0.13	10.80	14.30
s38417	2	4	0.18	0.18	0.18	0.20	0.40	37.4	1093.76
s38417	2	8	0.25	0.25	0.25	0.27	0.42	33.64	522.62
s38417	2	16	0.45	0.45	0.45	0.47	0.49	300.86	637.18
s38417	2	32	0.90	0.90	0.90	0.92	0.60	394.47	953.98

Only BSIM has a linear *time complexity* of $O(|\mathcal{I}| \cdot m)$, where simulation and PT are carried out for each test. COV has to determine a solution to the set covering problem. A backtrack search is applied to determine subsets of size $\leq k$ of gates that cover all candidate sets $\mathcal{C}_i, 1 \leq i \leq m$, which takes $O(|\mathcal{I}|^k)$. The advanced simulation-based approaches also calculate these subsets. Additionally, for each subset simulation and PT is carried out per test. In total this takes time $O(|\mathcal{I}|^k \cdot |\mathcal{I}| m)$. For BSAT the SAT-solver searches for a satisfying solution on the k SAT-instances. Each SAT-instance contains one select input for a multiplexer per gate and per test a variable and an additional input per gate, leading to $|\mathcal{I}| + m \cdot 2|\mathcal{I}|$ variables. Additionally, l variables are used to restrict the number of select inputs with value 1. Thus, the search on one SAT-instance is carried out in $O(2^{|\mathcal{I}| \, m \, 2 + |\mathcal{I}| + l})$. Asymptotically the same result applies for the advanced SAT-based approach. But modern SAT-solvers drastically improve upon this theoretical upper bound. For example after choosing the values for the select inputs of the multiplexers the value of all other variables in the SAT-instance can be implied, which reduces the search space to the size $2^{|\mathcal{I}|}$ already. Pruning due to learning techniques further improves the search.

The *space complexity* of BSIM is also the smallest. Each test can be handled independently of the others leading to a complexity of $O(|\mathcal{I}| + m)$. COV stores the circuit, the current test and the set of candidates marked by PT for each test. In the worst case PT marks all gates of the circuit. Therefore the complexity in this case is $O(|\mathcal{I}| \cdot m)$. The advanced simulation-based approaches store the same information, but additionally re-simulation has to be done during the backtrack search. At each search level up to depth k this information has to be stored, yielding $O(k \cdot |\mathcal{I}| \cdot m)$ space requirements. The CNF generated for BSAT always contains a copy of the circuit for each test, i.e. $\Theta(|\mathcal{I}| \cdot m)$.

In summary, the basic simulation-based approaches BSIM and COV are very fast, but do not yield good diagnosis results. The other approaches have higher runtimes, but provide valid corrections. The experiments presented in the following section strengthen these theoretical observations.

5 Experimental Results

In the experiments the three basic approaches BSIM, COV and BSAT were considered. A number of 1-4 gate change errors were injected into circuits from the ISCAS89 benchmark set. The limit k was always set to the number of errors injected previously. Then, diagnosis was done for 4, 8, 16, 32 tests to show the finer resolution obtained from additional tests. In all cases a

(a) Avg.   (b) #sol.
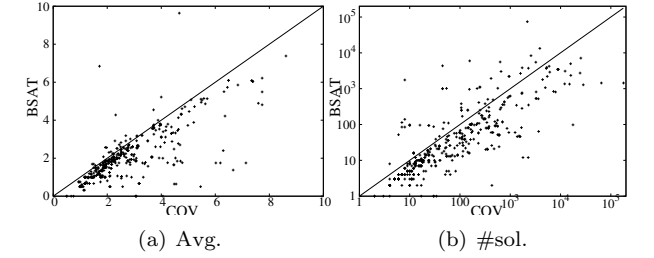
Figure 6. Quality of BSAT vs. COV

part of the same test-set has been used for an erroneous circuit.

All experiments were carried out on an AMD Athlon 3500+ (1GB, 2.2GHz, Linux). The resources were restricted to 512MB and 30 minutes of CPU-time. The SAT-solver Zchaff [15] was used. Zchaff supports incremental SAT to reuse learned clauses. The covering problem in COV was also solved using Zchaff.

The three basic approaches are compared with respect to runtime and quality. Table 2 shows the runtimes of the three approaches. Given are the name of the circuit $\mathcal{I}$, the number of errors p and the number of tests used m. For COV and BSAT runtimes to create the SAT-instance "CNF" (for COV this includes the time for BSIM), to calculate one solution "One" and to calculate all solutions "All" are reported.

Remark 1. *Note that* only BSAT *is guaranteed to return a* valid correction *since the other approaches do not carry out any effect analysis. Thus, BSAT solves a harder problem.*

Remark 2. *The* runtimes *of the basic approaches cannot be compared to that of the advanced approaches. For the SAT-based approaches heuristics have been proposed that yield a speed-up of more than 100 times (see Section 2.3). The advanced simulation-based techniques are applying a backtrack search and carry out effect analysis for each solution resulting in a drastic increase in runtimes (see Section 2.2).*

As expected BSIM is the fastest approach and takes less than 1 second of CPU time even for a large circuit as "s38417". Also COV computes corrections quite fast even when all corrections are retrieved. Due to the effect analysis BSAT needs much longer runtimes especially when all solutions are calculated. But this ensures to return only valid corrections.

Table 3 compares the quality of the approaches:
For BSIM
- The total number of gates that have been marked by PT is given ($|\cup \mathcal{C}_i|$).
- For each of these gates the distance to the nearest error was determined, i.e. the number of gates on a shortest path to any error. The average value of these distances is reported (avg_A).
- The number of gates that have been marked by the maximal number of tests is also given, i.e. $G_{max} = |\{g : \forall h \in \mathcal{I} : M(g) \geq M(h)\}|$.
- Again, the distance to the nearest error was determined for each of these gates. The minimal, maximal and average (avg_G) values of these distances are reported. If the minimal value is greater than zero, no actual error site was marked by the maximal number of tests.

Table 3. Quality of the basic approaches

$\mathcal{I}$	p	m	BSIM $\lvert \cup \mathcal{C}_i \rvert$	avg_A	G_{max}	min	max	avg_G	COV #sol	min	max	avg	SAT #sol	min	max	avg
s1423	4	4	100	3.68	4	1	4	2.75	5931	0	5.33	2.90	4239	0	4.00	2.18
s1423	4	8	115	3.78	2	3	4	3.50	28281	0	5.50	3.42	1281	0	3.50	1.78
s1423	4	16	126	3.90	1	1	1	1.00	7960	0	4.50	2.85	809	0	3.25	1.66
s1423	4	32	139	3.85	3	1	4	2.67	1716	0.33	4	2.37	767	0	3.25	1.61
s6669	3	4	90	6.89	83	0	12	7.17	415	0	7	4.18	1935	0	5.67	3.66
s6669	3	8	106	6.87	86	0	12	6.95	565	0	7	3.94	1029	0	5.67	3.72
s6669	3	16	117	6.85	69	0	12	6.94	2275	0	7.33	4.55	12	0	1	0.64
s6669	3	32	117	6.85	64	0	12	7.39	1790	0	7.33	4.48	12	0	1	0.64
s38417	2	4	52	4.75	18	0	11	4.61	156	0	11	4.67	5959	0	22.00	9.64
s38417	2	8	67	5.69	18	0	11	4.61	113	0	11	4.61	31	0	5.50	3.45
s38417	2	16	67	5.69	15	0	11	4.73	150	0	11	4.53	29	0	5.50	3.33
s38417	2	32	95	7.56	14	0	11	4.93	133	0	11	4.40	33	0	4.50	2.88

For COV and BSAT

- The number of solutions is given.
- For each gate in a solution the distance to the nearest error was determined. Per solution the average a of these distances was calculated. The minimal, maximal and average value of a over all solutions is reported.

These distance measures give an intuition up to which depth the designer has to analyze the circuit when starting from a solution returned by one of the approaches. A small value of this distance is desirable.

The table shows that BSIM alone does usually not yield a good diagnosis result. The number of gates that have the highest count from PT (G_{max}) can be quite large, see e.g. "s6669". While often an actual error site is among these gates, this cannot be guaranteed. Based on these results the designer may have to analyze a large part of the circuit before finding an error. COV considers subsets up to size k of all marked gates. Thus, the solution space is large. Using more tests may even increase the solution space, because more gates are marked by PT. Similarly, for BSAT the solution space is large. More tests may also increase this space, when additional outputs are introduced into the diagnosis problem. If no additional outputs are introduced the number of solutions is reduced. Besides the fact, that all solutions calculated by BSAT are valid corrections, also their quality is better in all cases, except for "s38417" when only 4 tests were considered. When more tests were used BSAT returned the best results. Figure 6 shows results for all benchmarks of BSAT versus COV for the value of "avg" (Figure 6(a)) and the number of solutions (Figure 6(b)). The figures show that BSAT usually returns a smaller number of solutions of a better quality. This directly implies time savings during design debugging.

In summary the approaches behave as expected from the theoretical analysis. BSAT is slower than the other approaches, but returns by far the best results. Nonetheless even the simple approaches often calculate solutions of good quality. This opens areas for future research.

6   Summary and Future Work

The relations between simulation-based and SAT-based diagnosis have been investigated in detail. Theoretically and empirically it has been shown that the basic simulation-based approaches BSIM and COV are fast, but cannot guarantee to return a valid correction. Moreover COV may not retrieve all valid corrections. This costs time when the designer has to filter out corrections that are not valid. BSAT needs more computation time, but returns good diagnosis results that are guaranteed to be a valid correction for a given test-set. The same is true for the advanced approaches that use different search paradigms.

Conclusions that show directions for future research on the diagnosis approaches follow. While BSIM does not guarantee that an actual error site has been marked by the largest number of tests, this happened in almost all experiments. In the same way the results returned by COV were not too far from the real errors in most cases. This suggests a hybrid approach. The fast engines of BSIM and COV can be used to direct the SAT-search by tuning the decision heuristics of the solver. A second possibility is to choose an initial correction (that may not be valid) and use SAT-based diagnosis to turn it into a valid correction.

References

[1] M. S. Abadir, J. Ferguson, and T. Kirkland. Logic verification via test generation. *IEEE Trans. on CAD*, 7:172–177, 1988.

[2] M. Abramovici, P. Menon, and D. Miller. Critical path tracing - an alternative to fault simulation. In *Design Automation Conf.*, pages 214–220, 1983.

[3] M. Ali, S. Safarpour, A. Veneris, M. Abadir, and R. Drechsler. Post-verification debugging of hierarchical designs. In *Int'l Conf. on CAD*, pages 871–876, 2005.

[4] M. Ali, A. Veneris, S. Safarpour, R. Drechsler, A. Smith, and M.S.Abadir. Debugging sequential circuits using Boolean satisfiability. In *Int'l Conf. on CAD*, pages 204–209, 2004.

[5] V. Boppana, R. Mukherjee, J. Jain, M. Fujita, and P. Bollineni. Multiple error diagnosis based on Xlists. In *Design Automation Conf.*, pages 660–665, 1999.

[6] G. Cabodi, P. Camurati, F. Corno, P. Prinetto, and M. Reorda. A new model for improving symbolic product machine traversal. In *Design Automation Conf.*, pages 614–619, 1992.

[7] M. Garey and D. Johnson. *Computers and Intractability - A Guide to NP-Completeness*. Freeman, San Francisco, 1979.

[8] D. W. Hoffmann and T. Kropf. Efficient design error correction of digital circuits. In *Int'l Conf. on Comp. Design*, pages 465–472, 2000.

[9] S.-Y. Huang and K.-T. Cheng. Errortracer: Design error diagnosis based on fault simulation techniques. *IEEE Trans. on CAD*, 18(9):1341–1352, 1999.

[10] A. Kuehlmann, D. I. Cheng, A. Srinivasan, and D. P. LaPotin. Error diagnosis for transistor-level verification. In *Design Automation Conf.*, pages 218–224, 1994.

[11] T. Larrabee. Test pattern generation using Boolean satisfiability. *IEEE Trans. on CAD*, 11:4–15, 1992.

[12] C.-C. Lin, K.-C. Chen, S.-C. Chang, M. Marek-Sadowska, and K.-T. Cheng. Logic synthesis for engineering change. In *Design Automation Conf.*, pages 647–651, 1995.

[13] J. Liu and A. Veneris. Incremental fault diagnosis. *IEEE Trans. on CAD*, 24(4):1514–1545, 2005.

[14] J. Marques-Silva and K. Sakallah. GRASP – a new search algorithm for satisfiability. In *Int'l Conf. on CAD*, pages 220–227, 1996.

[15] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Design Automation Conf.*, pages 530–535, 2001.

[16] A. Smith. Diagnosis of combinational logic circuits using Boolean satisfiability. Master's thesis, University of Toronto, Canada, 2004.

[17] A. Smith, A. Veneris, and A. Viglas. Design diagnosis using Boolean satisfiability. In *ASP Design Automation Conf.*, pages 218–223, 2004.

[18] A. Veneris and I. N. Hajj. Design error diagnosis and correction via test vector simulation. *IEEE Trans. on CAD*, 18(12):1803–1816, 1999.

[19] J. Whittemore, J. Kim, and K. Sakallah. SATIRE: A new incremental satisfiability engine. In *Design Automation Conf.*, pages 542–545, 2001.