Exact Diagnosis using Boolean Satisfiability

Heinz Riener*
*Institute of Space Systems,
German Aerospace Center, Germany
heinz.riener@dlr.de

Goerschwin Fey*†
†Faculty of Mathematics and Computer Science,
University of Bremen, Germany
goerschwin.fey@dlr.de

***Abstract*—We propose an exact algorithm to model-free diagnosis with an application to fault localization in digital circuits. We assume that a faulty circuit and a correctness specification, e.g., in terms of an un-optimized reference circuit, are available. Our algorithm computes the exact set of all minimal diagnoses up to cardinality k considering all possible assignments to the primary inputs of the circuit. This exact diagnosis problem can be naturally formulated and solved using an oracle for *Quantified Boolean Satisfiability* (QSAT). Our algorithm uses *Boolean satisfiability* (SAT) instead to compute the exact result more efficiently. We implemented the approach and present experimental results for determining fault candidates of digital circuits with seeded faults on the gate level. The experiments show that the presented SAT-based approach outperforms state-of-the-art techniques from solving instances of the QSAT problem by several orders of magnitude while having the same accuracy. Moreover, in contrast to QSAT, the SAT-based algorithm has any-time behavior, i.e., at any-time of the computation, an approximation of the exact result is available that can be used as a starting point for debugging. The result improves while time progresses until eventually the exact result is obtained.**

I. Introduction

Motivation. Growing design complexity and shrinking time-to-market make verification and debugging for hardware designs crucial. Respins at 45nm technologies cost millions per mask set. Functional errors, e.g., due to bugs in electronic design automation tools or implementations of erroneous engineering change orders, are hard to catch, and when caught during functional verification, designers are left with the problem of identifying their root causes and fixing the design, which is an even more challenging and time-consuming task.

Formal tools that assist in automatically finding and fixing design bugs are desirable. Given a specification, e.g., in form of an un-optimized reference circuit, functional equivalence checking allows to check whether a design produces the same outputs as its specification for all possible input assignments. The benefits over simulation-based methods lies in the fact that the whole input space is taken into account, avoiding that subtle corner cases slip through the analysis. Especially algorithms that leverage reductions to the *Boolean Satisfiabiilty* (SAT) problem have been successfully used for solving a wide range of formal verification and debugging problems. The digital design and its specification are conjointly expressed as a Boolean formula interwoven with additional debugging logic such that potential errors (or corrections) can be extracted from satisfying assignments or proofs of the unsatisfiability of the formula.

Problem. *Fault localization based on SAT* [1] was introduced as an algorithmic approach to identify potentially faulty gates in a circuit described as a net-list on the gate level using a SAT oracle. No specification is required, but a list of input assignments for which the circuit behaves incorrectly and the corresponding correct, expected outputs have to be provided. The fault candidates computed are all gates at which the circuit can be rectified with respect to the provided input-output assignments. Since the list of assignments is a priori fixed, typically only an overapproximation of the fault candidates is obtained without any indication of the precision or quality of the result. If a specification of the correct, intended behavior of the circuit is available, e.g., in terms of an un-optimized reference implementation, the exact set of all fault candidates can be computed by quantifying over all input assignments for which the faulty circuit and the reference circuit produce different outputs. Due to the quantifier alternation in the exact formulation, i.e., does a diagnosis (subset of components) *exist* such that the circuit can be rectified at those components *for all* possible assignments to the primary inputs, SAT-based techniques are not directly applicable. Oracles that decide the *Quantified Boolean Satisfiability* (QSAT) problem exist. Our experiments with QSAT-based formulation, however, indicate that QSAT oracles do not scale well on our problem instances and time out in almost any case.

Solution. In this paper, we propose the first exact approach to diagnosis using a SAT oracle. The approach first generates a coarse overapproximation of the set of diagnoses and then iteratively refines the approximation by generating counterexamples that are particularly generated to refute spurious diagnoses and remove them from the set, until the exact set is obtained. This approach is essentially a *CounterExample-Guided Abstraction Refinement* (CEGAR) directed search technique that lazily instantiates quantifiers to precisely solve the QSAT problem. The formulation of the problem can deal with multiple faults and works with or without a fault model. We implemented the approach and compute the exact set of fault candidates for digital circuits with seeded faults assuming that no fault model is available. The experimental results indicate that the proposed approach is several orders of magnitude faster than implementations based on state-of-the-art oracles for the QSAT problem. The speed-up stems from the fact that our approach to exact diagnosis takes advantage of several properties of the problem which are not recognized by general-purpose approaches to QSAT and thus outperforms them on

our problem instances. Moreover, the proposed approach in contrast to QSAT, has any-time behavior such that at any-time an (over-)approximation of the exact result is available that can be used for debugging and improves when time progresses.

Contribution. This paper makes the following contributions.

- An exact approach to model-free diagnosis using a SAT oracle with any-time behavior. The approach computes a coarse overapproximation of the set of diagnoses and systematically removes spurious diagnoses until the exact set is obtained.
- An approach to solve a special case of $\exists\forall\exists$-queries, which enumerates all minimal elements of the outermost exist quantifier, and is applicable for other problems with the same structure, e.g., non-Boolean domains or modulo background theories.
- An implementation of the exact diagnosis approach and experimental evaluation in the context of exact fault localization for gate level circuits with seeded faults. The experiments indicate that the presented approach outperforms QSAT by several orders of magnitude.

Structure. The remainder of the paper is structured as follows: in the next section, Section II, related work is discussed. Section III is dedicated to exact diagnosis. The k-all diagnosis problem is introduced and an approximate algorithm as well as two exact algorithms are described. In Section IV, we present experimental results. Section V concludes.

II. Related Work

In this section, we shortly survey approaches to diagnosis and fault localization. This paper presents the first exact approach to enumerate all minimal diagnoses for a faulty circuit when a correctness specification is available.

Diagnostic reasoning. Early attempts to fault localization stem from diagnostic reasoning developed in the field of artificial intelligence. These approaches are inspired by ideas and notions from philosophy. Two central principles are common: i) identifying a fault corresponds to finding the root *cause* of the observed specification violation and ii) according to Occam's razor, simpler causes are always preferred. The notion of cause hereby depends on a *causal theory* and was for many years researched in philosophy.

Diagnosis from first principles [2], [3] rests on the observation that when a change applied to a system results in a system that no longer exhibits the specification violation, the changed components can be used as a characterization of the fault. From this perspective, the changed components are one potential cause for the observed specification violation. Assuming that changes that involve fewer parts of the system, are easier to understand, diagnostic reasoning strives for computing small changes. Finding changes of minimal size, called *minimal diagnoses*, is NP-complete [3]. All minimal diagnoses can be efficiently enumerated using a hitting set algorithm [3]. The general theoretical framework of diagnosis from first principles is flexible and requires only that the system has to be formalized in a suitable logic. A component may be a gate of a digital circuit or an expression in a program. The theory is fault model-free and allows for considering single as well as multiple faulty components [2]. Diagnosis from first principles developed into *consistency-based diagnosis*. Alternatively, *abductive diagnosis* [4] arose which focused on incorporating knowledge about the causal relationship between faults and systems. The framework of abductive diagnosis captures common reasoning in medical diagnosis to explain a symptom by finding a set of causes that imply the symptom. In contrast to consistency-based diagnosis, abductive diagnosis is not fault model-free, but requires a description of the possible failure modes of a component. An explanation of an observed error is then an assignment of failure modes to the system's components. In the 90ies, both approaches to diagnosis eventually converged and are today known as *model-based diagnosis*.

Fault localization. In the hardware community, different approaches to localize faults in combinational and sequential circuits have been developed. For combinational circuits, early approaches to fault localization (and rectification) are either based on Boolean equation solving [5], [6] (predating most work on SAT), BDDs [7], [8], or multi-valued simulation [9], [10]. The problem of diagnosing sequential circuits was reduced to combinational diagnosis by constructing an *iterative logic array* (ILA) (also called *time-frame expansion*) from the circuit [11]. The considered fault models are mostly limited to stuck-at faults and inversion of gate outputs and thus not applicable for the general case of diagnosis. With the widespread usage of SAT-based techniques, *fault diagnosis using SAT* [12], [1] was developed. The approach is applicable to combinational and sequential circuits using an a priori fixed set of counterexamples. Moreover, fault diagnosis using SAT could be used with and without additional fault models. QBF-based formalizations [13] were proposed as alternative and more succinct encodings for diagnosing sequential circuits which avoid the blow-up of the problem size due to the time-frame expansion. Automatic fault localization [14] was also addressed in the context of LTL model checking for sequential circuits at the gate level and the register-transfer level. Maximum satisfiability-based techniques [15] as well as techniques based on the extraction of UNSAT cores [16] further improved the performance of SAT-based debugging techniques.

Solving $\exists\forall$-queries and beyond. At the hearth of the k-all diagnosis problem, a restricted QSAT formula has to be solved with exactly two quantifier alternations. Due to the fact that many problems can be expressed as QSAT problems with only a few quantifier alternations, algorithms for solving these queries received recently more attention. *CounterExample-Guided Inductive Synthesis* (CEGIS) [17] was introduced as a technique for solving $\exists\forall$-queries with an application to parameter synthesis in sketches of programs. For Boolean domains, the QBF solver RaReQS [18] generalized this approach to an arbitrary number of quantifiers. EFSMT [19] was proposed to solve $\exists\forall$-queries modulo theories with several applications in control theory, e.g., finding Lynjapunov functions [19] or

parameter synthesis for cyber-physical systems [20].

III. Exact Diagnosis

This section is dedicated to the major contributions of the paper. In Section III-A, the k-all diagnosis problem is introduced that can be exactly solved using a QSAT oracle. The exact computation based on QSAT, however, is typically too costly in practice. As an alternative, in Section III-B, an approximate algorithm is presented that leverages — instead of QSAT — a SAT oracle with the drawback that no indication of the quality of the computed diagnoses is possible. In Section III-C, an exact algorithm to k-all diagnosis is given that outperforms the QSAT-based algorithm by several orders of magnitude but has the same accuracy, i.e., the k-all diagnosis problem is solved exactly.

A. The (k-All) Diagnosis Problem

Let $\mathbb{B} := \{0, 1\}$. Suppose that C is a circuit and S its correctness specification. Let $\mathcal{X}$ be the input domain and $\mathcal{Y}$ the output domain. We use the predicate $\mathsf{correct} : \mathcal{X} \to \mathbb{B}$ to denote if circuit C is correct with respect to specification S for input $x \in \mathcal{X}$ and call $x \in \mathcal{X}$ a *counterexample* if $\neg\mathsf{correct}(x)$ holds.

Further suppose that $\mathcal{G} := \{g_1, \ldots, g_p\}$ is the totally ordered set of gates of C. We describe sets of gates as bit-strings of length p and define a function

$$\begin{aligned}\mathsf{bs} \; &: 2^{\mathcal{G}} \to \mathbb{B}^p, \\ &\Delta \mapsto 1^{\chi_\Delta(g_1)} \cdots 1^{\chi_\Delta(g_p)},\end{aligned}$$

that maps from a set of gates to the corresponding bit-string, where χ_Δ is the characteristic function of $\Delta \subseteq \mathcal{G}$. The function bs is bijective such that the inverse bs^{-1} that maps a bit-string back to its representation as set of gates is uniquely defined.

The *diagnosis problem* is, given a faulty circuit C with specification S, i.e., for some inputs $x \in \mathcal{X} : \neg\mathsf{correct}(x)$ holds, to determine where C potentially can be rectified. We use the modified correctness predicate $\mathsf{diag} : \mathcal{X} \times \mathbb{B}^{|\mathcal{G}|} \times \mathbb{B}^{|\mathcal{G}|} \to \mathbb{B}$ to denote if circuit C becomes correct with respect to specification S for input $x \in \mathcal{X}$ when the values $v \in \mathbb{B}^{|\mathcal{G}|}$ are injected at the outputs of the gates $\Delta \subseteq \mathcal{G}$ described by the bit-string $\mathsf{bs}(\Delta) = d \in \mathbb{B}^{|\mathcal{G}|}$. Consequently, Δ is a *diagnosis* (or *fault candidate*) if

$$\forall x \in \mathcal{X} : \exists v \in \mathbb{B}^{|\mathcal{G}|} : \mathsf{diag}(x, \mathsf{bs}(\Delta), v), \tag{1}$$

holds.

The *k-all diagnosis problem* is, given a faulty circuit C and a specification S, to determine the set $\mathcal{D}$ of all minimal subsets $\Delta \subseteq \mathcal{G}$ with $|\Delta| \leqslant k$ such that Δ is a diagnosis for C.

An *exact algorithm* to diagnosis determines $\mathcal{D}$ exactly such that every element $\Delta \in \mathcal{D}$ is a minimal diagnosis (*soundness*) and no element not in $\mathcal{D}$ is also a minimal diagnosis (*completeness*). The exact k-all diagnosis problem can be formalized as an incremental QSAT query

$$\begin{aligned}\exists d \in \mathbb{B}^{|\mathcal{G}|} : \forall x \in \mathcal{X} : \exists v \in \mathbb{B}^{|\mathcal{G}|} : \\ \mathsf{diag}(x, d, v) \wedge |d| \leqslant k,\end{aligned} \tag{2}$$

where each assignment to d is systematically extracted and blocked. This formulation of diagnosis is fault model-free and consequently all kinds of design faults can be considered. The user, however, has to provide a maximal bound on the fault cardinality k. If knowledge about the nature of the faults to be diagnosed is available, this information can be incorporated into the QSAT instances in form of additional constraints to reduce the number of computed diagnoses.

A straight-forward implementation of an exact algorithm that computes the set of all minimal diagnoses exactly is presented in Fig. 1. Initially, the set D is empty and the cardinality bound $i = 1$. In each iteration, a QSAT query of the form of Eq. 2 is checked for satisfiability. If satisfiable, a new diagnosis bit-string d is extracted and added to D. If unsatisfiable, the cardinality bound is increased. Since all domains are finite, eventually the algorithm terminates when $i > k$ and returns the set D of all diagnosis bit-strings with cardinality at most k. Each bit-string in $d \in D$ corresponds to a diagnosis $\Delta = \mathsf{bs}^{-1}(d)$ such that the set $\mathcal{D}$ of all minimal diagnoses is obtained as

$$\mathcal{D} := \{\mathsf{bs}^{-1}(d) \mid d \in D\}.$$

B. Approximate k-All Diagnosis using SAT

Fault localization using SAT [1] computes an over-approximation of the exact k-all diagnosis problem by invoking a SAT oracle multiple times. The $\forall$-quantifier in Eq. 2 is approximated using a fixed list of counterexamples. These counterexamples are, e.g., pre-computed using model checking or obtained while testing the circuit. Suppose that $\hat{x}_1, \ldots, \hat{x}_n$ are counterexamples and a diagnosis is only computed with respect to these counterexamples, then Eq. 2 reduces to

$$\bigwedge_{i=1}^{n} \mathsf{diag}(\hat{x}_i, d, v_i) \wedge |d| \leqslant k, \tag{3}$$

with free variables $d \in \mathbb{B}^{|\mathcal{G}|}$ and $v_i \in \mathbb{B}^{|\mathcal{G}|}$ for $1 \leqslant i \leqslant n$.

Eq. 3 can be used to check if a given set $\Delta \subseteq \mathcal{G}$ is a diagnosis considering counterexamples $\hat{x}_1, \ldots, \hat{x}_n$ using a SAT oracle. In order to obtain the modified correctness predicate diag, the circuit C is instrumented with debugging logic and then translated into *conjunctive normal form* (CNF) — the preferred input form of state-of-the-art SAT oracles. The instrumentation used is sketched in Fig. 2 (on the top): for each component, e.g., gate to be diagnosed, a multiplexer is added at the output. The multiplexer is used to inject values when enabled. If the select signal $sel = 1$, the output y of the component is assigned to a fresh input variable v and if $sel = 0$ the output $y = f(u_1, \ldots, u_l)$ is not affected, where f is a function that computes the component's output from some internal signals $u_1, \ldots, u_l$. The fresh input variable v has a non-deterministic (or existentially quantified) value. In practice, instead of encoding the multiplexer and the non-deterministic v variables into the SAT instance, only a variable sel for each select signal is added, which is then used as shown

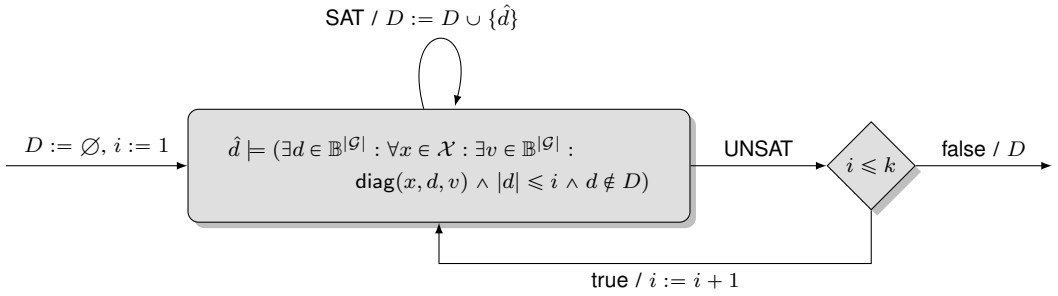
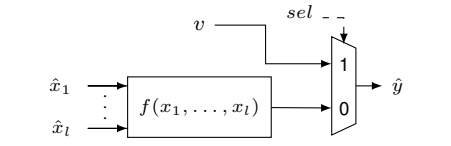


Fig. 1. Enumerating all diagnoses of cardinality less or equal to k using QSAT.



$$cl_1, \ldots, cl_l := \mathsf{CNF}(\hat{y} \leftrightarrow f(\hat{x}_1, \ldots, \hat{x}_n))$$

$$(sel \vee cl_1) \ldots (sel \vee cl_l)$$

Fig. 2. Debugging logic and CNF encoding for SAT-based diagnosis.

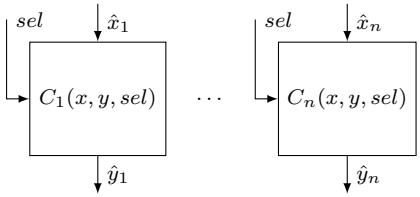


Fig. 3. SAT-based diagnosis for a circuit C and counterexamples $\hat{x}_1, \ldots, \hat{x}_n$.

in Fig. 2 (on the bottom) to disable clauses when assigned true.[1]

Given the counterexamples $\hat{x}_1, \ldots, \hat{x}_n$. For each counterexample $\hat{x}_i$, a copy of the circuit C_i instrumented with debugging logic is constructed, where the primary inputs are assigned to $\hat{x}_i$ and the primary outputs are assigned to the corresponding correct values $\hat{y}_i$ (obtained from the specification). The select signals sel are shared over all copies such that the same multiplexers are selected from all copies. The construction is sketched in Fig. 3. By incrementally checking for satisfying assignments to sel and blocking them, all solutions to the k-all diagnosis problem are obtained.

Example 1: Consider the gate level circuit `c17` in Fig. 4 (on the top) as a simple example of a faulty circuit. The circuit is taken from the ISCAS'85 benchmarks and consists of 6 gates with 5 PIs and 2 POs. The circuit is faulty, i.e., a stuck-at one fault materializes at the input side of g_2 denoted as S-a-1 in the figure. Suppose that the counterexample $x_1 \cdots x_5 = 01101$

[1]If v is not added to the CNF instance, still the component's output variables have to be quantified such that the number of quantifier alternations is not reduced.

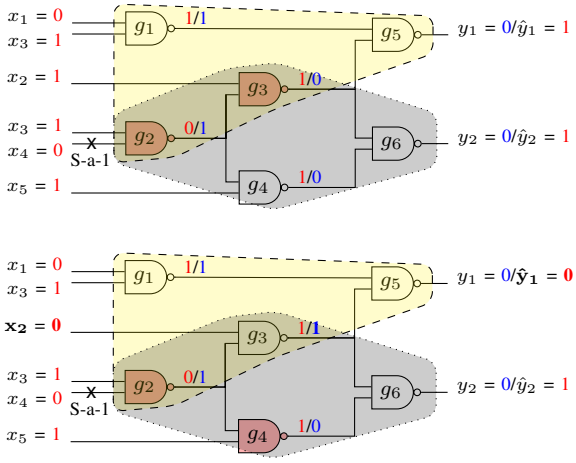


Fig. 4. A faulty circuit realization (`c17`) with an injected stuck-at one fault with two different counterexamples (on top 01101-00 and on bottom 00101-00).

with corresponding correct outputs $y_1 y_2 = 00$ is known and is used for diagnosis. The counterexample is annotated to the circuit in form of blue and red values. The blue values are actually not known, but only implicitly available through the specification. However, we use them to illustrate how diagnosis operates. If a blue and red value at the output of a gate differ, we say the gate is *conflicting*. The conflicting gates, here g_2, g_3, g_4, g_5, g_6, qualify as potential candidates for rectifying the circuit. In the figure, additionally, the input cones of g_5 and g_6, respectively, are shown with dashed and dotted borders. Since the stuck-at fault affects both outputs, only gates in the intersection of the two input cones of g_5 and g_6 are potential causes for the fault, which further excludes g_4, g_5, and g_6. Under these considerations g_2 and g_3 are potential causes for the faulty circuit and reported as diagnosis to the user. In fault localization using SAT, the structure of the circuit is encoded by logical constraints that are solved to compute the two singletons $\{g_2\}$ or $\{g_3\}$ as an approximate solution to the 1-all diagnosis problem for the given counterexample.

Example 2: Now, consider the same gate level circuit with the same fault again but assume the counterexample shown in Fig. 4 (on the bottom) is provided. The counterexample $x_1 \cdots x_5 = 01101$ with the corresponding correct outputs

$y_1y_2 = 00$ when used for diagnosis reveals the conflicting gates g_2, g_4, g_6. Due to the same structural arguments as discussed before g_2 can be excluded such that fault localization reports the two singletons $\{g_2\}$, $\{g_4\}$ as diagnosis to the user. If the counterexample from the previous example is added, fault localization computes the intersection of the two solutions, which exactly pinpoints the fault at $\{g_2\}$.

The two previous examples, Ex. 1 and Ex. 2, illustrate that the result of fault localization using SAT strongly depends on providing the right counterexamples to the algorithm as input. Moreover, the approximate solution of the 1-all diagnosis problem eventually converges to the exact solution if the right counterexamples are added. In the next section, we introduce an approach to systematically construct counterexamples for this purpose.

C. Exact k-All Diagnosis using SAT

In this section, we introduce an exact approach to k-all diagnosis based on SAT. The approach computes the exact set $\mathcal{D}$ of all minimal diagnoses (as in the Fig. 1) but avoids the costly calls to the QSAT oracle and uses a SAT oracle instead. Additional tweaks in the encoding improve the performance, such that the described approach significantly outperforms QSAT-based formulations of the problem. The overall approach is shown in Fig. 5. As input a faulty circuit C, its correctness specification S and an initial set X_0 of counterexamples is required. The initial counterexamples X_0 may be generated by model checking C with respect to S. As output, the set D of all diagnosis bit-strings is computed, which directly corresponds to the set $\mathcal{D}$ of all minimal diagnoses.

The construction of the diagnosis predicate diag depends on how the correctness specification S is provided. For instance, if S is a reference circuit, techniques from functional equivalence checking are applicable and diag corresponds to a miter constructed from the circuit C and the reference circuit S, where C is instrumented with additional debugging logic. If S is a set of properties, techniques from property checking are applicable and diag corresponds to a checker circuit that is constructed from C and the properties of S, where C is again instrumented with additional debugging logic. In the following, we will assume that diag can be effectively constructed from C and S as a CNF formula.

The exact approach to k-all diagnosis based on a SAT oracle repeats two steps to compute the exact set of diagnoses:

1) **Overapproximate diagnoses**: A coarse overapproximation D_j (initialized with $D_1 := \varnothing$) of the potential causes of the fault in C is computed in terms of components that can be rectified to correct the circuit's input-output relation. The approximation algorithm from Eq. 3 is used with a fixed list X_j of counterexamples (initialized with X_0).
2) **Generate counterexample**: The overapproximation is strengthened by constructing new counterexamples that refute spurious causes. Whenever a new counterexample was constructed, the counterexample $\hat{x}$ is added to $X_{j+1} := X_j \cup \{\hat{x}\}$ and the computation of step 1 is repeated to compute a new D_{j+1} in the next step. Eventually the exact set of causes is obtained.

The mutual interplay of step 1 and step 2 leverages the fact that the computation of the overapproximation in step 1 is monotonic such that whenever a counterexample is added to X_j, the number of diagnoses in the next step either stays the same or decreases (but never increases). However, in the sets D_j only minimal diagnoses are collected, i.e., for no two elements $d, d' \in D_j : d \subseteq d'$.[2] Consequently, $D_{j+1} \subseteq D_j$ if $k = 1$. Otherwise, if $k > 1$, when a counterexample is added, either $|D_{j+1}| < |D_j|$ or $|D_{j+1}| \geqslant |D_j|$. In the first case, the counterexample refutes one (or more) of the diagnoses. In the latter case, the counterexample refutes a diagnosis d and collects one (or more) new diagnoses $d'_1, \ldots, d'_l$, which were previously not considered because $d \subseteq d'_i$ for $1 \leqslant i \leqslant l$. Since the domain of $\mathcal{X}$ is finite, the termination of the loop is guaranteed. In the worst-case all counterexamples have to be enumerated, however, in practice often a few iterations are sufficient.

Refuting spurious diagnoses. In order to terminate fast the *right* counterexamples have to be generated which are those that refute spurious diagnoses. A diagnosis Δ is *spurious* if a counterexample exists that produces an output that does not agree with the specification regardless which values are injected at the outputs of the diagnosed components. For each counterexample $\hat{x}$ that refutes Δ the property

$$\forall v \in \mathbb{B}^{|\mathcal{G}|} : \neg\mathsf{diag}(\hat{x}, \mathsf{bv}(\Delta), v) \tag{4}$$

has to hold.

Consequently, to classify a diagnosis Δ as spurious, a counterexample x has to be found such that

$$\exists x \in \mathcal{X} : \forall v \in \mathbb{B}^{|\mathcal{G}|} : \neg\mathsf{diag}(x, \mathsf{bv}(\Delta), v) \tag{5}$$

holds. This QSAT problem has a single quantifier alternation and cannot be solved with a SAT oracle. Also, experiments revealed that state-of-the-art QSAT oracles are not effective in solving this query.

However, notice that Δ determines at which component's outputs the circuit can potentially be rectified such that all $v \in \mathbb{B}^{|\mathbb{G}|}$ are don't care for all components in $\mathcal{G}\backslash\Delta$ and Eq. 5 holds if and only if

$$\exists x \in \mathcal{X} : \forall v \in \mathbb{B}^{|\Delta|} : \neg\mathsf{diag}(x, \mathsf{bv}(\Delta), v) \tag{6}$$

holds.

Since $|\Delta| \leqslant k$, the $\forall$-quantor in the formula can be avoided by replicating the diagnosis predicate diag and instantiating v for all possible evaluations in $\mathbb{B}^{|\Delta|}$. This results in a blow-up of the formula that is exponential in k. Since in practice k is typically a very small value, this blow-up is acceptable and the resulting formulæ can be effectively solved with a SAT solver. Alternatively, lazy approaches [17], [18] to instantiate the $\forall$-quantor are applicable, but not expected to perform better as long as the flattened query fits into the computer's main memory.

[2]The elements of D_j form an antichain.

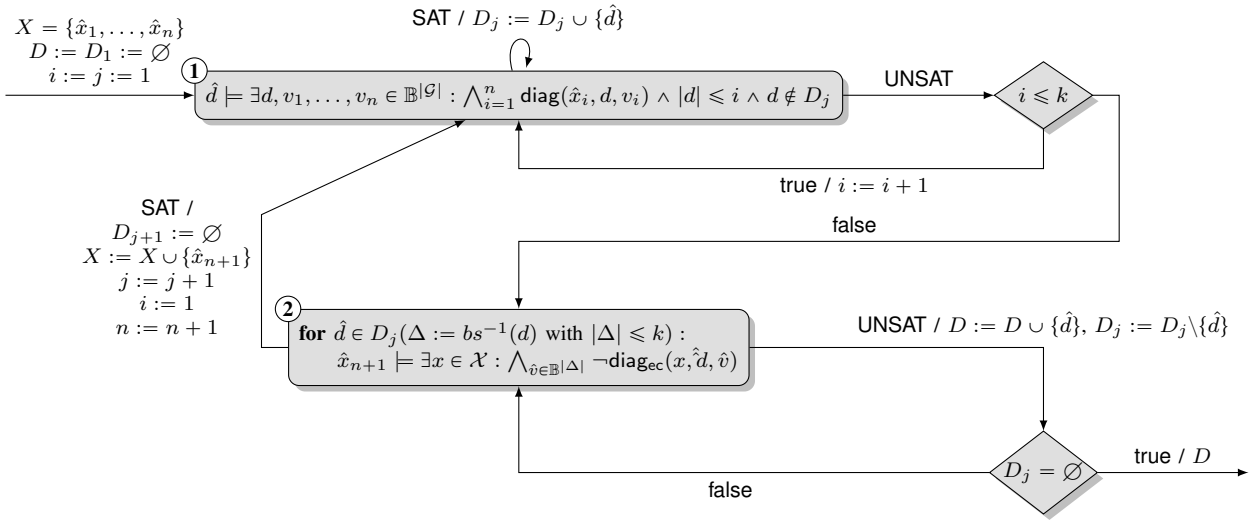


Fig. 5. Enumerating all diagnoses of cardinality less or equal to k using SAT.

Notice also that immediately when a new counterexample is discovered, step 2 terminates and the set of diagnoses are re-computed. Alternatively, step 2 could have been repeated to generate more counterexamples to reduce the set of diagnoses when re-computed even further. However, re-computing the diagnoses immediately has the advantage that in the best case not only the currently analyzed diagnosis is refuted but also several others too, which avoids costly SAT refutation calls as well as the construction of redundant counterexamples.

Any-time character. The diagnosis approach described above has any-time character. Quickly a coarse overapproximation of the set of diagnoses is computed and then iteratively strengthened until the exact set is obtained. In contrast to a QSAT-based approach, at any-time the approach provides a result that can be used as a starting point for debugging. When time progresses, more spurious candidate diagnoses are removed until eventually the exact set is obtained. Also, the approach can be interrupted, e.g., when the SAT oracle got stuck with a few remaining queries that cannot be solved in the given resource limits.

IV. Experiments

The exact algorithm to k-all diagnosis described in the previous section as well as the QSAT-based formulation from Fig. 1 were implemented and evaluated in the context of exact fault localization for gate level circuits. Both algorithms are by design exact and produce the same sets of diagnoses. The SAT-based formulation in contrast to the general QSAT-based formulation leverages several properties of the problem that improves its overall performance.

Benchmarks. As benchmarks, the ISCAS'85 benchmark suite as well as crafted adder and multiplier designs were used. The ISCAS'85 benchmarks were selected to allow for a comparison to other fault localization techniques from the literature. The crafted designs were used to shed more light on the performance characteristic of the QSAT-based approach. To keep our implementations simple and flexible, all circuits were first translated to *And-Inverter Graphs* (AIGs) utilizing ABC [21].

SAT-based diagnosis. As SAT solver, we used MiniSAT 2.2.0[3] via API. The maximal timeout for refuting a diagnosis was configured to 32 seconds. For each SAT query, a timer thread is forked that waits on a conditional variable and signals an interrupt to MiniSAT if the time bound is exceeded, such that the SAT query is terminated. The timeout is stepwisely increased (starting from 1 second), i.e., whenever all SAT queries in a round timeout, the timeout is doubled for the next round. With this strategy, easy to solve SAT queries are prioritized assuming that the generated counterexamples refute also other diagnoses when considered.

QSAT-based diagnosis. As QSAT oracles, we used DepQBF 5.0[4], RAReQS 1.1[5], and Quantor 3.2[6] via their QDIMACS file interfaces. To ensure that writing and reading QDIMACS files does not have a drastic impact on the performance of the QSAT oracles, we repeated the experiments with DepQBF via API which slightly (but not significantly) improved the performance.

Experiments. Both algorithms were used to generate the exact sets of all minimal diagnoses for the benchmark circuits with seeded faults. The diagnosis algorithms are fault model-free and can be used with any fault if a specification of the correct, intended behavior of the circuit is available. For the experiments, we seeded 100 different faults into each benchmark circuit and computed for each benchmark and seeded fault the exact set of all minimal diagnoses. The experiments were conducted on a quad-core Intel® Core™ i5-2520M CPU with 2.50GHz and 8GB RAM running Arch Linux kernel 4.4.5-1. Table I and Table II present the experimental results for

[3]MiniSAT, http://minisat.se/MiniSat.html [22]
[4]DepQBF, http://lonsing.github.io/depqbf/ [23]
[5]RAReQS, http://sat.inesc-id.pt/~mikolas/sw/rareqs/ [24]
[6]Quantor, http://fmv.jku.at/quantor/ [25]

TABLE I
EXACT 1-ALL DIAGNOSIS FOR ISCAS'85 BENCHMARKS AND CRAFTED DESIGNS.

Name	Exact Diagnosis			SAT-Based			QSAT-Based		
	$\mathbf{c}$ -	$\lvert\boldsymbol{\Delta}\rvert_{\mu}^{\mathbf{1}}$ -	$\lvert\boldsymbol{\Delta}\rvert_{\sigma}^{\mathbf{1}}$ -	$\lvert\mathbf{X}\rvert_{\mu}$ -	$\lvert\mathbf{X}\rvert_{\sigma}$ -	$\mathbf{t}$ $[s]$	$\mathbf{t_D}$ $[s]$	$\mathbf{t_R}$ $[s]$	$\mathbf{t_Q}$ $[s]$
bvadd02	0.62	1.87	1.92	0.90	0.92	0.00	0.01	0.01	0.01
bvadd04	0.84	2.85	1.84	1.64	1.14	0.01	0.04	0.05	*0.02
bvadd08	0.93	2.99	1.56	1.81	1.10	0.02	9.35	41.58	8.84
bvmul02	0.64	1.48	1.40	0.86	0.77	0.00	0.01	0.01	0.01
bvmul04	0.66	3.91	4.58	2.13	2.01	0.01	0.04	0.08	0.06
bvmul08	0.84	6.73	9.41	3.40	2.76	1.96	T/O	T/O	T/O
c17	1.00	1.50	0.58	1.43	0.50	0.00	0.01	0.02	0.01
c432	0.98	6.96	6.20	2.18	1.10	0.11	T/O	T/O	T/O
c499	1.00	5.88	8.76	3.36	2.22	0.84	T/O	T/O	T/O
c880	1.00	5.12	3.72	2.84	1.45	0.87	T/O	T/O	T/O
c1355	1.00	5.95	5.95	3.72	2.35	1.06	T/O	T/O	T/O
c1908	1.00	6.27	8.82	2.42	1.45	0.87	T/O	T/O	T/O
c2670	0.99	12.29	17.44	3.81	2.53	2.91	T/O	T/O	T/O
c3540	0.99	8.18	7.04	2.20	1.24	25.74	T/O	T/O	T/O
c5315	1.00	6.94	5.22	2.95	2.26	11.48	T/O	T/O	T/O
c6288	1.00	1.82	0.90	4.26	2.17	117.09	T/O	T/O	T/O
c7552	1.00	10.25	12.59	3.75	2.38	29.39	T/O	T/O	T/O

TABLE II
EXACT 2-ALL DIAGNOSIS FOR ISCAS'85 BENCHMARKS AND CRAFTED DESIGNS.

Name	Exact Diagnosis					SAT-Based			QSAT-Based		
	$\mathbf{c}$ -	$\lvert\boldsymbol{\Delta}\rvert_{\mu}^{\mathbf{1}}$ -	$\lvert\boldsymbol{\Delta}\rvert_{\sigma}^{\mathbf{1}}$ -	$\lvert\boldsymbol{\Delta}\rvert_{\mu}^{\mathbf{2}}$ -	$\lvert\boldsymbol{\Delta}\rvert_{\sigma}^{\mathbf{2}}$ -	$\lvert\mathbf{X}\rvert_{\mu}$ -	$\lvert\mathbf{X}\rvert_{\sigma}$ -	$\mathbf{t}$ $[s]$	$\mathbf{t_D}$ $[s]$	$\mathbf{t_R}$ $[s]$	$\mathbf{t_Q}$ $[s]$
bvadd02	1.44	2.04	1.66	2.26	3.28	1.67	0.96	0.00	0.01	0.01	*0.00
bvadd04	1.86	1.22	1.79	8.81	8.41	3.56	1.78	0.03	0.10	0.19	*0.04
bvadd08	1.98	0.57	1.36	9.79	6.94	3.99	1.47	0.11	35.19	139.03	*34.62
bvmul02	1.48	1.81	1.34	1.00	1.19	1.48	0.78	0.00	0.00	0.00	*0.00
bvmul04	1.68	3.70	4.76	23.17	27.99	6.86	4.72	0.12	0.46	1.13	*0.43
bvmul08	2.00	3.05	7.03	118.21	195.32	14.41	11.03	103.09	T/O	T/O	T/O
c17	1.91	0.19	0.49	3.85	2.08	2.48	0.95	0.01	0.01	0.01	*0.00
c432	1.98	0.69	2.08	40.77	47.39	5.01	2.09	0.87	T/O	T/O	T/O
c499	2.00	0.19	1.13	32.91	57.33	7.41	4.60	14.31	T/O	T/O	T/O
c880	2.00	0.04	0.32	23.67	26.68	5.49	2.29	15.53	T/O	T/O	T/O
c1355	2.00	0.00	0.00	36.67	59.64	8.01	3.86	21.06	T/O	T/O	T/O
c1908	1.99	0.32	1.56	52.85	122.12	5.53	2.66	19.80	T/O	T/O	T/O
c2670	1.99	0.47	1.85	155.49	390.83	11.67	7.28	93.64	T/O	T/O	T/O
c3540	2.00	0.15	1.13	59.49	81.35	5.30	2.33	552.08	T/O	T/O	T/O
c5315	2.00	0.00	0.00	47.40	52.06	6.07	3.27	385.89	T/O	T/O	T/O
c6288	2.00	0.00	0.00	3.58	2.59	7.86	2.94	247.27	T/O	T/O	T/O
c7552	2.00	0.04	0.32	141.03	300.09	8.83	5.11	1849.34	T/O	T/O	T/O

all benchmarks with single and double faults, respectively. The tables are built similarly: the first column names the benchmark (Name); the remaining columns of the table are split into three parts. The first part is dedicated to the results of exact diagnosis and lists the mean of the effective cardinality ($\mathbf{c}$) as well as the mean and the standard derivation ($|\boldsymbol{\Delta}^{\mathbf{f}}|_{\mu}$ and $|\boldsymbol{\Delta}^{\mathbf{f}}|_{\sigma}$ with $f = 1$ for single faults and $f = 2$ for double faults) of the generated diagnoses. The second part is dedicated to the exact algorithm to k-all diagnosis using SAT and lists the mean and standard derivation ($|\mathbf{X}|_{\mu}$ and $|\mathbf{X}|_{\sigma}$) of the number of counterexamples generated and the time for computing the exact diagnosis sets ($\mathbf{t}$). The third part is dedicated to the QSAT-based formulation and lists the time for computing the exact diagnosis sets for different QSAT oracles ($\mathbf{t_D}$ refers to DepQBF, $\mathbf{t_R}$ refers to RAReQS, and $\mathbf{t_Q}$ refers to Quantor). For each experiment, a timeout of 3600 seconds (1 hour) was set. A timeout is marked in the table with T/O.

Discussion. As expected, the presented exact algorithms based on SAT and QSAT computed the same exact sets of minimal diagnoses except for Quantor, which missed the enumeration of several possible assignments to the outermost existentially quantified variables. The corresponding entries are marked with a * in the table.

For some of the benchmarks, a mean effective cardinality less than 1.0 is reported which indicates that some of the seeded faults did not affect the semantics of the circuits, i.e., the circuits are functionally equivalent to the design without the seeded faults, such that no diagnosis can be generated.

The SAT-based exact algorithm outperformed the exact QSAT-based formulation in all cases and for all QSAT oracles considered. The QSAT oracles timed out for all ISCAS benchmarks except for the toy example c17. In an attempt to determine the runtime required to compute all minimal diagnoses for c432, we applied the QSAT-based algorithm without a timeout. DepQBF was able to solve *one* QSAT query, i.e., one diagnosis was produced, within 7 hours. The

other QSAT oracles did not succeed within this time bound.

Within the considered QSAT oracles, DepQBF performed best, but still timed out in most of the cases. In most cases, the exact algorithm based on SAT required only a few counterexamples (between 2 and 25 were typically enough) to exactly pinpoint all minimal diagnoses. We did not implement any optimization for generating the counterexamples, e.g., structural optimizations of the circuit or clause minimization strategies, but directly encoded the circuit using Tseyting encoding. An improved encoding thus has the potential to speed-up the SAT-based algorithm even more. The time required for computing the minimal diagnoses for the larger benchmarks, e.g., `c7552`, stems mostly from our timeout strategy. In the first iteration of the algorithm several thousand diagnoses are generated. The SAT-based algorithm then attempts to refute each individual diagnosis with a timeout of 1 second but due to the size of the circuit does not succeed in solving the corresponding SAT query within the timeout. The timeout is then increased and the refutation step is repeated. Consequently, many seconds are lost for finding the right timeout. We assume that more advanced strategies for refuting diagnoses would improve performance, e.g., using multiple threads for refuting counterexamples with different initial timeouts or different strategies to increase the timeouts.

V. Conclusion

In this paper, the first exact algorithm to model-free diagnosis was introduced. The algorithm computes the exact set of minimal diagnoses with the aid of a SAT oracle by iteratively constructing counterexamples to strengthen an overapproximation of the set of minimal diagnoses. In each iteration, a new counterexample is generated and some spurious diagnoses are removed from the overapproximation until the exact set is obtained. The algorithm implicitly uses a new approach to effectively solve the special case of $\exists\forall\exists$-QSAT queries and enumerates all minimal solutions of the outermost $\exists$-quantifier. The diagnosis algorithm was implemented and an experimental evaluation in the context of exact fault localization for gate level circuits with seeded faults shows that the presented algorithm outperforms state-of-the-art QSAT oracles by several orders of magnitude.

Acknowledgment

This work was supported in part by the European Union (Horizon 2020 IMMORTAL project, grant no. 644905).

References

[1] A. Smith, A. G. Veneris, M. F. Ali, and A. Viglas, "Fault diagnosis and logic debugging using Boolean satisfiability," *IEEE Transactions on CAD of Intergrated Circuits and Systems*, vol. 24, no. 10, pp. 1606–1621, 2005.

[2] J. de Kleer and B. C. Williams, "Diagnosing multiple faults," *Artificial Intelligence*, vol. 32, no. 1, pp. 97–130, 1987.

[3] R. Reiter, "A theory of diagnosis from first principles," *Artificial Intelligence*, vol. 32, no. 1, pp. 57–95, 1987.

[4] P. T. Cox and T. Pietrzykowski, "General diagnosis by abductive inference," in *Symposium on Logic Programming*, 1987, pp. 183–189.

[5] J. C. Madre, O. Coudert, and J.-P. Billon, "Automating the diagnosis and the rectification of design errors with PRIAM," in *International Conference on Computer Aided Design*, 1989, pp. 30–33.

[6] M. Tomita, T. Yamamoto, F. Sumikawa, and K. Hirano, "Rectification of multiple logic design errors in multiple output circuits," in *Design Automation Conference*, 1994, pp. 212–217.

[7] P.-Y. Chung, Y.-M. Wang, and I. N. Hajj, "Diagnosis and correction of logic design errors in digital circuits," in *Design Automation Conference*, 1993, pp. 503–508.

[8] Q. Zhang and C. Trullemans, "Logic verification of incomplete functions and design error location," in *Correct Hardware Design and Verification Methods*, 1993, pp. 68–79.

[9] S.-Y. Kuo, "Locating logic design errors via test generation and don't-care propagation," in *European Design Automation*, 1992, pp. 466–471.

[10] A. M. Wahba and E. J. Aas, "Verification and diagnosis of digital systems by termary reasoning," in *Correct Hardware Design and Verification Methods*, 1993, pp. 55–67.

[11] A. M. Wahba and D. Borrione, "Design error diagnosis in sequential circuits," in *Correct Hardware Design and Verification Methods*, 1995, pp. 171–188.

[12] M. F. Ali, A. G. Veneris, A. Smith, S. Safarpour, R. Drechsler, and M. S. Abadir, "Debugging sequential circuits using Boolean satisfiability," in *International Conference on Computer Aided Design*, 2004, pp. 204–209.

[13] H. Mangassarian, A. G. Veneris, S. Safarpour, M. Benedetti, and D. E. Smith, "A performance-driven QBF-based iterative logic array representation with applications to verification, debug and test," in *International Conference on Computer Aided Design*, 2007, pp. 240–245.

[14] G. Fey, S. Staber, R. Bloem, and R. Drechsler, "Automatic fault localization for property checking," *IEEE Transactions on CAD of Intergrated Circuits and Systems*, vol. 27, no. 6, pp. 1138–1149, 2008.

[15] S. Safarpour, H. Mangassarian, A. G. Veneris, M. H. Liffiton, and K. A. Sakallah, "Improved design debugging using maximum satisfiability," in *Formal Methods in Computer-Aided Design*, 2007, pp. 13–19.

[16] A. Sülflow, G. Fey, R. Bloem, and R. Drechsler, "Using unsatisfiable cores to debug multiple design errors," in *Great Lakes Symposium on VLSI*, 2008, pp. 77–82.

[17] A. Solar-Lezama, "Program sketching," *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 5-6, pp. 475–495, 2013.

[18] M. Janota, W. Klieber, J. Marques-Silva, and E. M. Clarke, "Solving QBF with counterexample guided refinement," *Artificial Intelligence*, pp. 1–25, 2016.

[19] C. Cheng, N. Shankar, H. Ruess, and S. Bensalem, "EFSMT: A logical framework for cyber-physical systems," *CoRR*, vol. abs/1306.3456, 2013. [Online]. Available: http://arxiv.org/abs/1306.3456

[20] H. Riener, R. Könighofer, G. Fey, and R. Bloem, "SMT-based CPS parameter synthesis (tool presentation)," in *ARCH Workshop*, 2016.

[21] R. K. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," 2010, pp. 24–40.

[22] N. Eén and N. Sörensson, "An extensible SAT-solver," in *International Conference on Theory and Applications of Satisfiability Testing*, 2003, pp. 502–518.

[23] F. Lonsing and U. Egly, "Incremental QBF solving," in *International Conference on Principles and Practice of Constraint Programming*, 2014, pp. 514–530.

[24] M. Janota, W. Klieber, J. Marques-Silva, and E. M. Clarke, "Solving QBF with counterexample guided refinement," in *International Conference on Theory and Applications of Satisfiability Testing*, 2012, pp. 114–128.

[25] A. Biere, "Resolve and expand," in *International Conference on Theory and Applications of Satisfiability Testing*, 2004, pp. 59–71.