Fault Diagnosis using Quantified Boolean Formulas

Hratch Mangassarian[1] Andreas Veneris[1,2] Marco Benedetti[3]

***Abstract*— Automatic debugging of sequential circuits has been considered a practically intractable task due to the excessive memory and run-time requirements associated with tackling industrial-size problems. This paper proposes a novel Quantified Boolean Formula (QBF) based approach for fault diagnosis in sequential circuits. A performance-driven succinct QBF encoding of the problem, coupled with the tremendous present-day advances in QBF solvers make this strategy a successful one. Extensive experiments on industrial circuits confirm the memory advantage and demonstrate the outstanding performance of the proposed framework.**

I. Introduction

Fault diagnosis is an integral part of *failure analysis* pertaining to the identification of the failing portions in a given faulty netlist. This information is later used in *defect analysis* where the chip is physically examined in order to determine the cause(s) of failure. Because physical examination is inevitably slow and resource intensive [1], the efficiency of failure analysis is conditioned on the resolution of fault diagnosis.

Combinational fault diagnosis is a well examined problem with a plethora of techniques to discover fault candidates [1]–[6]. On the other hand, there has been significantly less work in sequential fault diagnosis where not all the memory elements of the design are scanned and directly observable [7]. Recently, a Boolean satisfiability (SAT) based strategy [8] has been proven to be particularly effective in tackling combinational and sequential fault diagnosis, compared to traditional Binary Decision Diagram (BDD) and simulation-based diagnosis approaches [8]. However, all these methods require the *time-frame expansion* of the circuit under diagnosis for the lengths of the failing test traces. Given thousand-cycle long test traces, coupled with the ever-increasing design sizes, this *Iterative Logic Array* (ILA) representation inevitably strains memory resources and affects the diagnosis efficiency.

The contribution of this paper is a performance-driven succinct formulation of the multiple-fault diagnosis problem for sequential circuits as an instance of Quantified Boolean Formula (QBF) satisfiability. An appropriately constructed hardware construction, along with a suitable quantification of the problem variables, make it possible to replace the memory-intensive ILA circuit replication by a single copy of the netlist under diagnosis. This memory-efficient QBF encoding of the problem, coupled with a state-of-the-art QBF solver that was tuned to accommodate the intricacies and needs of diagnosis, culminate in a successful sequential diagnosis framework.

Empirical results show a 93% average reduction in the memory footprint, which confirms the expected memory advantage of the proposed approach. Furthermore, whereas previous QBF-based ILA encodings in the Bounded Model Checking community [9], [10] have invariably shown that memory compression comes at the cost of tremendous performance cutbacks, our technique also exhibits a run-time reduction of up to 5 times with respect to SAT-based diagnosis. Both memory and run-time improvements culminate in a remarkable 93% increase in the number of solved problem instances compared to the best previous method.

The rest of the paper is organized as follows. Section II gives background information on SAT and QBF and describes SAT-based fault diagnosis. Section III illustrates our novel formulation for QBF-based fault diagnosis. Section IV contains experiments and Section V concludes the paper.

II. Preliminaries

A. Boolean Satisfiability

A propositional logic formula Φ over a set of Boolean variables $\mathcal{V}$ is said to be satisfiable or SAT if it has a *satisfying assignment*: a *truth assignment* of $\mathcal{V}$ that makes it evaluate to 1 (true). Otherwise, Φ always evaluates to 0 (false) and is said to be unsatisfiable or UNSAT. The problem of Boolean satisfiability consists of determining whether Φ is SAT, thus formally asking whether $\exists \mathcal{V} \mid \Phi$. In modern SAT solvers, the logic formula Φ is given in *Conjunctive Normal Form* (CNF) as a conjunction (AND) of *clauses* where each clause is a disjunction (OR) of *literals*. A literal is an instance of a variable or its negation. In order for a formula to be SAT, at least one literal in each clause must evaluate to 1. For example, the CNF formula given in (1) is SAT because $\{a = 1, b = 0, c = 1\}$ is a satisfying assignment.

$$\Phi = (a \vee b) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (c) \tag{1}$$

A logic circuit can be converted to a CNF formula in linear time [11], such that there is a one-to-one correspondence between the variables of the generated CNF formula and the gate outputs of the corresponding circuit, and such that satisfying variable assignments in the CNF formula correspond to valid gate output values in the circuit. Hence, a circuit and its corresponding SAT formulation are often referred to interchangeably in this paper.

Modern SAT solvers [12] are based on the search algorithm of Davis, Putnam, Loveland and Logemann (DPLL) [13]. Although SAT is an NP-complete problem, large SAT problems with millions of variables and clauses are typically solvable thanks to advancements in dynamic branching heuristics, conflict-based learning and watched data structures.

B. Quantified Boolean Formulas

All variables of a SAT problem are existentially ($\exists$) quantified. QBF is a generalization of SAT in which some variables are instead universally ($\forall$) quantified. A QBF formula in prenex CNF form is written as:

[1]University of Toronto, ECE Department, Toronto, ON M5S 3G4 ({hratch, veneris}@eecg.toronto.edu)

[2]University of Toronto, CS Department, Toronto, ON M5S 3G4

[3]LIFO, University of Orléans, BP 6759 – 45067 Orléans Cedex 2, France (Marco.Benedetti@univ-orleans.fr)

$$Q_1\mathcal{V}_1, Q_2\mathcal{V}_2, \ldots, Q_r\mathcal{V}_r \mid \Phi$$

where

- the *prefix* $Q_1\mathcal{V}_1, Q_2\mathcal{V}_2, \ldots, Q_r\mathcal{V}_r$ consists of *quantifiers* $Q_i \in \{\forall, \exists\}$, such that $Q_i \neq Q_{i+1}$, and mutually disjoint variable sets $\mathcal{V}_i$ (also called *scopes*).
- the *matrix* Φ is a CNF formula on the variables in the prefix.

Q_r (Q_1) is the innermost (outermost) quantifier. A variable $v \in \mathcal{V}_i$ is called an *existential* (*universal*) variable if $Q_i = \exists$ ($Q_i = \forall$). A scope $\mathcal{V}_i$ is said to *dominate* a scope $\mathcal{V}_j$ if $i < j$. If there exists a way to assign a truth value to each existential variable as a function of its dominating universal variables such that every combination of assignments to the universal variables can be extended to satisfy the matrix, the QBF problem is said to be SAT, otherwise it is UNSAT. For example,

$$\exists a \forall b \exists c \quad | \quad (b \vee c) \wedge (\bar{a} \vee \bar{b} \vee \bar{c}) \wedge (a)$$

is SAT because there exists an assignment to a ($a = 1$) such that *for all* b there exists an assignment to c ($c = 1$ if $b = 0$ and $c = 0$ if $b = 1$) that satisfies the matrix. Note that in general, a *tree of satisfying assignments* is needed for satisfiability, due to the fact that both truth values of each universal variable need to be extended to satisfying assignments.

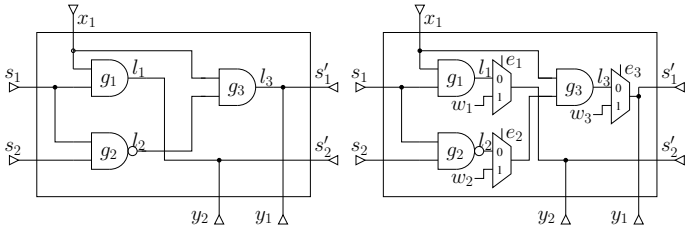
QBF is a PSPACE-complete problem, which makes it a more powerful encoding formalism than SAT since PSPACE $\supseteq$ NP. State-of-the-art QBF solvers [14]–[16] are much more diverse than SAT solvers in terms of their underlying solving strategies. Resolution and expansion [14], skolemization [15], as well as BDD-based strategies are common and competitive, next to the more standard search-based approaches [16]. Today, typically solvable QBF problems contain tens to hundreds of thousands of variables and clauses.

C. Notation and SAT-based Fault Diagnosis

In this section, we present notation and describe SAT-based fault diagnosis which will serve as the reference for QBF-based fault diagnosis both in Section III and in the experiments. The following notation is used throughout the paper. x, y, l and s (x_i, y_i, l_i and s_i) are the Boolean vectors (bits) respectively denoting the (i^{th} bit in the) primary inputs, primary outputs, internal circuit lines and state elements of a sequential circuit. The behavior of a sequential circuit can be formally described by the predicate $T(s, s', x, y, l)$, which evaluates to 1 if and only if, given the current-state s and primary input values x, the internal circuit lines evaluate to l, the primary outputs evaluate to y and the next-state evaluates to s'.

The fault diagnosis algorithm starts after testing has failed. The specification is given as a logic netlist $\mathcal{C}$, and the faulty behavior of the circuit under test is given as a set of q failing test sequences $\mathbf{V} = \{V^1, V^2, \ldots, V^q\}$, also called *counter-examples* that do not match the expected behavior of the specification. The objective of diagnosis is to locate all possible faults in the netlist that could explain the observed test-vector responses. Let m_j denote the number of cycles in test sequence V^j, also referred to as the *length* of V^j. Each counter-example consists of an initial-state, a sequence of input vectors and the corresponding faulty outputs:

$$V^j = \langle v_s^{j,0}, \langle v_x^{j,1}, v_x^{j,2}, \ldots, v_x^{j,m_j} \rangle, \langle v_y^{j,1}, v_y^{j,2}, \ldots, v_y^{j,m_j} \rangle \rangle$$



(a) Original circuit T (b) Enhanced circuit T_{en}

Fig. 1. Enhancing the original circuit with error models

where $v_s^{j,0}$ denotes the initial-state of the j^{th} counter-example, and $v_x^{j,k}(v_y^{j,k})$ denotes the primary input (output) values of the k^{th} cycle, or *time-frame*, of the j^{th} counter-example.

SAT-based fault diagnosis [8] encodes the problem as a SAT instance whose satisfying assignments correspond to the potential error locations in the circuit. This is done by: *i) enhancing* the specification netlist $\mathcal{C}$ by introducing appropriate *error modeling* hardware in the original circuitry, *ii)* appropriately replicating this enhanced netlist for all counter-examples and all time-frames, *iii)* applying the initial-state, input and output constraints according to the counter-examples in $\mathbf{V}$, and *iv)* constraining the number of simultaneous error locations to N, which is initialized to 1 and iteratively increased until a SAT solution is found. These steps are detailed in what follows.

i) For each internal circuit line l_i, a multiplexer with select line e_i is introduced. This is shown in Fig. 1(b) for lines l_1, l_2 and l_3. An inactive multiplexer select line ($e_i = 0$) does not modify the circuit, whereas an active select line ($e_i = 1$) disconnects l_i from its fanouts and replaces it with a new unconstrained input w_i, which can freely "fix" any potential fault on line l_i. The predicate of this enhanced circuit is denoted by $T_{en}(s, s', \{x, w, e\}, y, l)$, where w and e respectively denote the unconstrained inputs and select lines of the multiplexers.

ii) For each counter-example V^j, the enhanced circuit is replicated as an Iterative Logic Array (ILA) [17] of m_j cycles. This is also called *time-frame expansion*, which consists of unfolding the combinational component of a sequential circuit such that the next-state of each time-frame is connected to the current-state of the next time-frame. Building an ILA for each of the q counter-examples in $\mathbf{V}$ will result in a 2-dimensional *grid* of replicated circuits, as shown in Fig. 2. Counter-examples are replicated along the vertical dimension, whereas the horizontal dimension is associated with time-frame expansion. Note that throughout the grid, introduced multiplexers corresponding to the same original circuit line share the *same* select line because if a certain line is erroneous, it could potentially need a fix at any time-frame and for any counter-example.

For each $z \in \{s, x, w, y, l\}$ in T_{en}, let z^{c_j, t_k} denote the corresponding variable in the k^{th} time-frame of the j^{th} counter-example ILA in the grid; let Z^{c_j} (Z^{t_k}) denote the set of corresponding variables in all time-frames of the j^{th} counter-example ILA (in the k^{th} time-frame of all counter-examples); and let $\mathbf{Z}$ denote the set corresponding variables in the whole grid. Moreover, let s^{c_j, t_0} denote the initial-state of the j^{th} counter-example. Using this notation, the grid shown in Fig. 2 can be

formally encoded as:

$$\Phi_{grid} \equiv \bigwedge_{j=1}^{q} \Phi_{ILA}^{c_j} \equiv \tag{2}$$
$$\bigwedge_{j=1}^{q} \bigwedge_{k=1}^{m_j} T_{en}(s^{c_j,t_{k-1}}, s^{c_j,t_k}, \{x^{c_j,t_k}, w^{c_j,t_k}, e\}, y^{c_j,t_k}, l^{c_j,t_k})$$

where $\Phi_{ILA}^{c_j}$ represents the CNF encoding of a single ILA corresponding to the j^{th} counter-example and e denotes the vector of common multiplexer select lines.

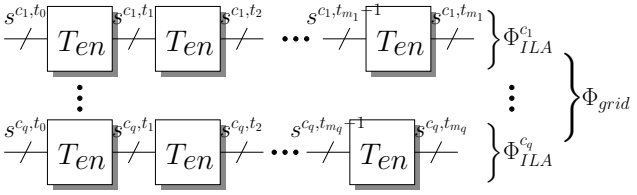


Fig. 2. The 2-dimensional grid Φ_{grid}

Note that this formulation allows for *model-free* fault diagnosis. Model-free diagnosis can "catch" any type of faulty behavior including non-determinism, which is potentially useful in identifying transient electrical faults. For a stuck-at-fault model, multiplexers corresponding to the same original circuit line should also share the same w's throughout the grid (*i.e.*, $\bigwedge_{j=1}^{q} \bigwedge_{k=1}^{m_j} (w^{c_j,t_k} \leftrightarrow w)$ is added to Φ_{grid}) in order to emulate a constant stuck-at value.

iii) Each counter-examples in $\mathbf{V}$ should constrain the initial-state, inputs and outputs of its corresponding ILA in order to ensure that the enhanced circuit's sequential response matches the counter-examples. We let Φ_{V^j} denote the input/output/initial-state constraints on the ILA corresponding to the j^{th} counter-example, and $\Phi_{\mathbf{V}} = \bigwedge_{j=1}^{q} \Phi_{V^j}$ denote all input/output/initial-state constraints applied to the grid.

iv) The number of activated select lines must be constrained, otherwise activating all the select lines would always satisfy the problem by replacing all circuit lines with unconstrained values. Hence, a final constraint Φ_N, forcing exactly N select lines in e to 1 is added to the problem.

The SAT problem corresponding to the j^{th} counter-example can now be given as:

$$\exists e, S^{c_j}, X^{c_j}, W^{c_j}, Y^{c_j}, L^{c_j} \quad | \quad \Phi_{ILA}^{c_j} \wedge \Phi_{V^j} \wedge \Phi_N \tag{3}$$

Finally, the full SAT problem is given as:

$$\exists e, \mathbf{S}, \mathbf{X}, \mathbf{W}, \mathbf{Y}, \mathbf{L} \quad | \quad \Phi_{grid} \wedge \Phi_{\mathbf{V}} \wedge \Phi_N \tag{4}$$

This problem essentially asks whether there exists a set of N erroneous lines that could be fixed (by activating the corresponding N bits in e) in order to match the response of the specification with the set of counter-examples. If Eq. 4 is UNSAT with $N = 1$, then $N = 2$ is tried and so on, until Eq. 4 becomes SAT. Furthermore, an *all-solution* SAT solver is used, which, provided that the problem is SAT, returns *all* possible fault locations (assignments to e) unlike a regular SAT solver which would return only one solution.

III. Fault Diagnosis using QBF

The bulk of the size of a SAT-based fault diagnosis problem comprises of the 2-dimensional circuit replication in Φ_{grid}, as shown in Fig. 2. Often, a large circuit must be replicated for several counter-examples, each containing thousands of cycles, which renders the memory requirements to build Φ_{grid} unreasonably excessive. In this section, we propose a succinct QBF-based formulation for fault diagnosis which "collapses" the replicated 2-dimensional grid into a single copy of the circuit, by taking advantage of universal quantification.

The sub-problem of first collapsing all q counter-examples into a 1-dimensional ILA by adding control circuitry and using universal quantification is simpler because the ILAs corresponding to the different counter-examples are decoupled, in the sense that one does not feed into the other. This has been solved for combinational circuits in [18] and is (implicitly) trivially extended to sequential circuits here. On the other hand, the remaining sub-problem of collapsing the time-frames of an ILA into a single copy of the circuit is more intricate because of the inherent state interdependence across ILA time-frames. In what follows, we first describe how to solve this harder sub-problem, namely how to simulate the behavior of an ILA using a single copy of the circuit. Later, we show how to integrate this with the approach given in [18] in order to collapse the whole grid into a single copy of the predicate.

Let us consider a single counter-example V of length m, which naturally corresponds to a single ILA, formally defined as Φ_{ILA} in Eq. 2. In what follows, all j and c_j superscripts are dropped because there is only one counter-example. We will demonstrate how to encode Eq. 3 as a QBF problem where the constraint equivalent to Φ_{ILA} includes only a *single copy* of the circuit, as opposed to a chain of m connected circuits as shown for each counter-example in Fig. 2. A hardware construction coupled with a meaningful QBF prefix help achieve this task efficiently by enabling the single predicate T_{en} to be used by the QBF solver to simulate every time-frame in the original ILA. In what follows, we present both an intuition and a formalization of our approach.

Intuitively, the same circuit predicate T_{en} must be satisfied under different primary inputs, primary outputs, current-states and next-states *for all* time-frames in the ILA. We first create a set of universal *time-frame select* variables dominating the replicated variables in $T_{en}(s, s', \{x, w, e\}, y, l)$ which will allow us to refer to all the time-frames in the ILA. The aim is to modify the inputs/outputs/states around the single circuit predicate T_{en} according to the time-frame select variables, such that as we go through all the assignments to the time-frame select vector, T_{en}'s environment goes through all the inputs/outputs/states in the original ILA.

Fig. 3 shows the hardware construction, denoted as Φ_{ILA}^{QBF} since it is logically equivalent to an ILA for a QBF formulation, consisting of four appropriately constrained multiplexers labeled MUX, respectively connected to the primary inputs, primary outputs, current-state and next-state of the circuit predicate T_{en}. The time-frame select variables are the common select lines of the MUXs. For a given assignment, they (conceptually) select a certain time-frame k in the original ILA, such that the MUXs

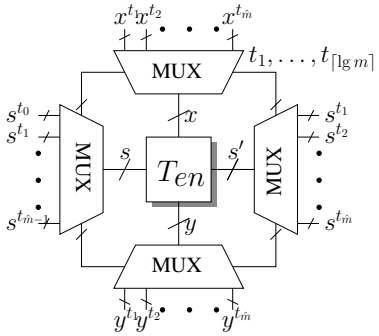


Fig. 3. The hardware construction Φ_{ILA}^{QBF}

connect the primary inputs $\{x, w, e\}$ (outputs y) of T_{en} to the primary inputs $\{x^{t_k}, w^{t_k}, e\}$ (outputs y^{t_k}) of the original ILA, the current state s of T_{en} to the $(k-1)^{st}$ state $s^{t_{k-1}}$ of the original ILA, and its next state s' to the k^{th} state s^{t_k}. This essentially serves to make T_{en} behave exactly like the predicate of the k^{th} time-frame in the ILA. Depending on the assignment given to the time-frame select vector, the single copy of T_{en} will simulate a different time-frame in the original ILA.

The next step is to state the SAT problem given in Eq. 3 as a QBF instance using the described construction. This amounts to deciding the order of the variables in the QBF prefix. In order to insure state contiguity (*i.e.*, that as T_{en} simulates different time-frames, the next-state of time-frame k and the current-state of time-frame $k+1$ are the *same* for all k), it is necessary that the set of ILA states $s^{t_0}, s^{t_1}, \ldots, s^{t_m}$ dominate the time-frame select variables. Informally, the QBF problem is then stated as follows:

> *Does there exist assignments to e and the states $s^{t_0}, s^{t_1}, \ldots, s^{t_m}$, such that for all time-frame select variables, $\Phi_{ILA}^{QBF} \wedge \Phi_V \wedge \Phi_N$ is satisfied?*

In formal terms, let $t = \langle t_1, t_2, \ldots, t_{\lceil \lg m \rceil} \rangle$ denote the vector of Boolean time-frame select variables. The binary description inherent to a QBF formulation forces us to consider $\hat{m} = 2^{\lceil \lg m \rceil}$ time-frames, where $\hat{m}$ denotes the smallest power of 2 greater or equal to m. This is harmless because Φ_V only constrains the first m time-frames to match the counter-example V, leaving the remaining unconstrained and hence trivially satisfiable. Now, Φ_{ILA}^{QBF} given in Fig. 3 can be formally described as:

$$\begin{aligned} \Phi_{ILA}^{QBF} \quad &\equiv \quad T_{en}(s, s', \{x, w, e\}, y, l) \qquad (5) \\ &\wedge MUX(\{s^{t_0}, \ldots, s^{t_{\hat{m}-1}}\}, t, s) \wedge MUX(\{s^{t_1}, \ldots, s^{t_{\hat{m}}}\}, t, s') \\ &\wedge MUX(\{x^{t_1}, \ldots, x^{t_{\hat{m}}}\}, t, x) \wedge MUX(\{y^{t_1}, \ldots, y^{t_{\hat{m}}}\}, t, y) \end{aligned}$$

where $MUX(\{x^{t_1}, \ldots, x^{t_{\hat{m}}}\}, t, x)$ sets x to x^{t_k} ($x \leftrightarrow x^{t_k}$) if and only if $t = \langle t_1, t_2, \ldots, t_{\lceil \lg m \rceil} \rangle$ is the binary encoding of $k-1$. The interpretation of the remaining MUXs is similar. For example, if $t = \langle t_1, t_2, t_3 \rangle = \langle 0, 1, 1 \rangle$, then x, y, s and s' will be respectively set to x^{t_4}, y^{t_4}, s^{t_3} and s^{t_4} in Φ_{ILA}^{QBF} in order to simulate the 4^{th} time-frame in the original ILA.

The question stated in italic above can now be formally expressed as the following QBF problem:

$$\exists e, S \ \forall t \ \exists X, Y, s, s', x, w, y, l \ \mid \ \Phi_{ILA}^{QBF} \wedge \Phi_V \wedge \Phi_N \qquad (6)$$

where S, X and Y respectively denote the set all states, original primary inputs and outputs in the original ILA.

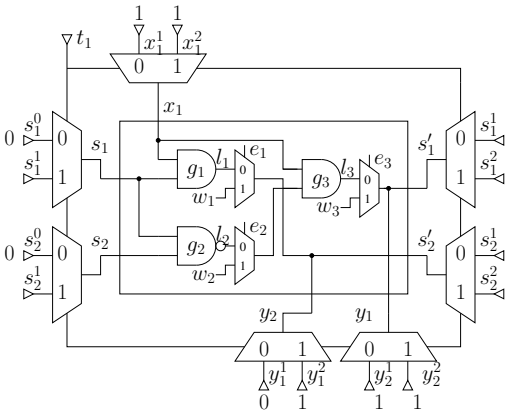


Fig. 4. Φ_{ILA}^{QBF} and Φ_V for the example

In the following example, the QBF fault diagnosis formulation given by Eq. 6 is generated in bit-level detail for a specific netlist and counter-example. This makes it easier for the reader to see and verify the formulation for a concrete instance.

Example 1 *Consider the specification netlist given in Fig. 1(a). Testing of a corresponding design implementation gives a counter-example consisting of 2 time-frames $V = \langle v_s^0, \langle v_x^1, v_x^2 \rangle, \langle v_y^1, v_y^2 \rangle \rangle$, where*

$$\begin{aligned} v_s^0 &= \langle v_{s_1}^0, v_{s_2}^0 \rangle = \langle 0, 0 \rangle \\ \langle v_x^1, v_x^2 \rangle &= \langle v_{x_1}^1, v_{x_1}^2 \rangle = \langle 1, 1 \rangle \\ \langle v_y^1, v_y^2 \rangle &= \langle \langle v_{y_1}^1, v_{y_2}^1 \rangle, \langle v_{y_1}^2, v_{y_2}^2 \rangle \rangle = \langle \langle 1, 0 \rangle, \langle 0, 1 \rangle \rangle \end{aligned}$$

We have the following constraints in the QBF formulation of the fault diagnosis problem:

$$\begin{aligned} \Phi_{ILA}^{QBF} \quad &\equiv \\ &T_{en}(\{s_1, s_2\}, \{s'_1, s'_2\}, \{x_1, w_1, w_2, w_3, e_1, e_2, e_3\}, \{y_1, y_2\}, \{l_1, l_2, l_3\}) \\ &\wedge MUX(\{\{s_1^{t_0}, s_2^{t_0}\}, \{s_1^{t_1}, s_2^{t_1}\}\}, t_1, \{s_1, s_2\}) \\ &\wedge MUX(\{\{s_1^{t_1}, s_2^{t_1}\}, \{s_1^{t_2}, s_2^{t_2}\}\}, t_1, \{s'_1, s'_2\}) \\ &\wedge MUX(\{x_1^{t_1}, x_1^{t_2}\}, t_1, x_1) \\ &\wedge MUX(\{\{y_1^{t_1}, y_2^{t_1}\}, \{y_1^{t_2}, y_2^{t_2}\}\}, t_1, \{y_1, y_2\}) \\ \Phi_V \quad &\equiv \quad \bar{s}_1^{t_0} \wedge \bar{s}_2^{t_0} \wedge x_1^{t_1} \wedge x_1^{t_2} \wedge y_1^{t_1} \wedge \bar{y}_2^{t_1} \wedge \bar{y}_1^{t_2} \wedge y_2^{t_2} \\ \Phi_N \quad &\equiv \quad (e_1 + e_2 + e_3) \leftrightarrow N \end{aligned}$$

where Φ_{ILA}^{QBF} is generated according to Eq. 5, Φ_V constrains the inputs/outputs/initial-state, and Φ_N constrains the number of active select lines in e to N.

Therefore, the corresponding QBF fault diagnosis problem is:

$$\begin{aligned} &\exists e_1, e_2, e_3, s_1^{t_0}, s_2^{t_0}, s_1^{t_1}, s_2^{t_1}, s_1^{t_2}, s_2^{t_2} \ \forall t_1 \qquad (7) \\ &\exists x_1^{t_1}, x_1^{t_2}, y_1^{t_1}, y_1^{t_2}, y_2^{t_1}, y_2^{t_2}, s_1, s_2, s'_1, s'_2, x_1, w_1, w_2, w_3, y_1, y_2, l_1, l_2, l_3 \\ &\quad \mid \ \Phi_{ILA}^{QBF} \wedge \Phi_V \wedge \Phi_N \end{aligned}$$

Fig. 4 shows the hardware construction Φ_{ILA}^{QBF} as well as the counter-example constraints Φ_V next to their corresponding inputs, outputs and initial-state bits. Given the QBF problem in Eq. 7 and starting with $N = 1$, a QBF solver will return the assignment $\{e_1 = 0, e_2 = 1, e_3 = 0\}$ of multiplexer select lines, which means that line l_2 is faulty.

A. Extension to Multiple Counter-Examples

In order to extend the QBF formulation given in Eq. 6 to deal with multiple (q) counter-examples, another hardware structure has to be built around that of Fig. 3, which allows us to go

through the different counter-examples in $\mathbf{V}$. Since the ILAs corresponding to different counter-examples are decoupled from each other, as shown in Fig. 2, this simply consists of introducing new MUXs that select the input/output/initial-state constraints corresponding to a certain counter-example, as done in [18]. Fig. 5 shows this new construction, referred to as Φ_{grid}^{QBF} because it essentially replaces Φ_{grid} of the SAT formulation given in Eq. 2. Here $c = \langle c_1, c_2, \ldots, c_{\lceil \lg q \rceil} \rangle$ denotes the vector of Boolean *counter-example select* variables. Once again, The binary description inherent to a QBF formulation forces us to consider $\hat{q} = 2^{\lceil \lg q \rceil}$ counter-examples, where $\hat{q}$ denotes the smallest power of 2 greater or equal to q. As explained for the time-frame select vector, this is harmless.

Given q counter-examples, the QBF problem corresponding to fault diagnosis can be stated as the following question:

> *Does there exist an assignment to e such that for all counter-example select variables, there exist assignments to the states $s^{t_0}, s^{t_1}, \ldots, s^{t_{m_j}}$, where m_j is the length of the j^{th} counter-example, such that for all time-frame select variables, $\Phi_{grid}^{QBF} \wedge \Phi_{\mathbf{V}} \wedge \Phi_N$ is satisfied?*

Formally, Φ_{grid}^{QBF} given in Fig. 5 is described as:

$$\begin{aligned} \Phi_{grid}^{QBF} \equiv \Phi_{ILA}^{QBF} &\wedge MUX(\{s^{c_1,t_0}, \ldots, s^{c_{\hat{q}},t_0}\}, c, s^{t_0}) \\ &\wedge MUX(\{X^{c_1}, \ldots, X^{c_{\hat{q}}}\}, c, X) \\ &\wedge MUX(\{Y^{c_1}, \ldots, Y^{c_{\hat{q}}}\}, c, Y) \end{aligned} \quad (8)$$

where the three new MUXs correspond to the three outer MUXs in Fig. 5 which select the counter-example. Note that the m used in Φ_{ILA}^{QBF} (see Eq. 5) should now be equal to the *maximum* counter-example length (*i.e.*, $m = \max_{j \in [1,q]} m_j$) in order to accommodate all the counter-examples.

The question stated in italic above can now be formally expressed as the following QBF problem:

$$\begin{aligned} &\exists e \;\; \forall c \;\; \exists S \;\; \forall t \;\; \exists \mathbf{X}, \mathbf{Y}, S^{t_0}, X, Y, s, s', x, w, y, l \\ &\quad | \;\; \Phi_{grid}^{QBF} \wedge \Phi_{\mathbf{V}} \wedge \Phi_N \end{aligned} \quad (9)$$

where $S^{t_0} = \{s^{c_1,t_0}, \ldots, s^{c_{\hat{q}},t_0}\}$ denotes the set of initial-states of all counter-examples.

The QBF formulation given in Eq. 9 contains 5 scopes ($\exists\forall\exists\forall\exists$). It is possible to represent the same problem with 3 scopes ($\exists\forall\exists$) and still have a single copy of the circuit in the problem matrix. Although this is also implemented, we will not go into the formulation details due to the lack of space. The idea is to put the ILAs corresponding to the counter-examples *next to* each other, thus obtaining a long 1-dimensional chain of not necessarily connected circuits, as opposed to the 2-dimensional grid shown in Fig. 2. Collapsing this chain using Eq. 6, with the exception that some current and next-states should *not* be connected, generates a 3-scope QBF problem.

IV. Experiments

We have implemented an encoding module in C++ which, given the specification and a set of failing traces, produces the described QBF formulations in either 3 or 5 scopes. The generated QBF problem is given to the state-of-the-art QBF solver `sKizzo` [15], which has been purposely modified to return, on `SAT` instances, not just one but all the valid assignments to e that identify fault locations. Five industrial circuits from OpenCores.org [19] are used to construct several fault-diagnosis problems.

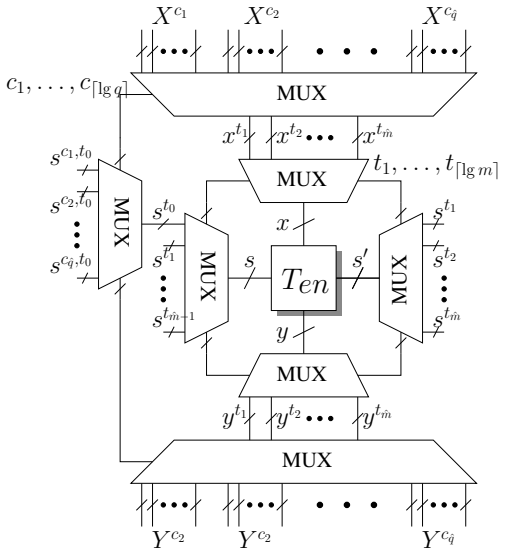


Fig. 5. The hardware construction Φ_{grid}^{QBF}

Counter-examples are created by manually changing the functionality of certain gates in the circuits and running pseudo-random simulations until a faulty response is exhibited. For each circuit, 6 different instances, each consisting of 8 counter-examples, are generated and the results are averaged out over these 6 instances. The number of simultaneous fault locations N is set to 1. The results of the proposed QBF-based formulations are compared to the SAT-based approach [8] that uses replication and `zChaff` [12] as the underlying SAT solver. Experiments are conducted on a Pentium IV 2.8 GHz Linux platform with a memory limit of 2 GB and a time-out of 2000 seconds.

Table I compares QBF-based and SAT-based model-free fault diagnosis results. The first three columns respectively show the circuit name, its number of gates and number of flip-flops (`DFF`s). The fourth and fifth columns respectively show the average and maximum counter-example lengths. The sixth column gives the average number of solutions (*i.e.*, possible fault locations). For each of the three approaches, namely SAT, 5-scope QBF and 3-scope QBF, columns *i) # solved*, *ii) abort reason*, *iii) time* and *iv) mem* respectively show *i)* the number of solved problem instances (out of 6), where an instance is considered solved if and only if *all* possible solutions are returned successfully and it is proved that no other solutions exist, *ii)* the most common aborting reason, where [TO] stands for time-out and [MO] stands for mem-out, *iii)* the total run-time in seconds to find all solutions including the time to prove completeness, and *iv)* the memory footprint of the file containing the problem instance in MBs. When averaging the run-times, an unsolved instance is counted as 2000 seconds, which is the time-out.

Table II compares QBF-based and SAT-based stuck-at-fault diagnosis results for the first three circuits. The counter-examples,

TABLE I

QBF-BASED VERSUS SAT-BASED MODEL-FREE FAULT DIAGNOSIS

Circuit and Diagnosis Info						SAT				5-scope QBF				3-scope QBF			
$\mathcal{C}$	# gates	# DFFs	avg m_j $j\in[1,q]$	max m_j $j\in[1,q]$	avg # solutions	# solved	abort reason	time (sec)	mem (MB)	# solved	abort reason	time (sec)	mem (MB)	# solved	abort reason	time (sec)	mem (MB)
A1	5,248	388	27.5	111	7.8	3/6	[MO]	1232.8	199	4/6	[TO]	791.7	7	6/6	–	210.0	12
A2	12,041	521	2.5	8	11.5	6/6	–	30.7	27	6/6	–	17.1	1	6/6	–	12.3	2
A3	265	22	365.1	931	7.0	0/6	[MO]	–	125	6/6	–	98.8	21	6/6	–	448.1	11
A4	2,449	347	89.3	449	3.0	3/6	[MO]	1001.8	351	3/6	[TO]	1001.6	33	4/6	[TO]	771.1	32
A5	2,012	90	21.5	56	3.2	6/6	–	41.2	56	6/6	–	17.0	2	6/6	–	9.1	2

TABLE II

QBF-BASED VERSUS SAT-BASED STUCK-AT-FAULT DIAGNOSIS

Diagnosis		SAT			5-scope QBF			3-scope QBF	
$\mathcal{C}$	avg # sols	# solv	abort reason	time (sec)	# solv	abort reason	time (sec)	# solv	time (sec)
A1	7.8	3/6	[MO]	1080.2	4/6	[TO]	907.7	6/6	311.5
A2	11.5	6/6	–	25.0	6/6	–	27.2	6/6	22.5
A3	4.0	0/6	[MO]	–	6/6	–	193.7	6/6	214.7

and therefore the memory footprints of the problems, are unchanged. The difference is that only stuck-at-faults are allowed to match the response of the specification to that of the counter-examples. This additional constraint on the problem upper-bounds the number of stuck-at-fault solutions by the number of model-free solutions. In fact, whereas A1 and A2 have the same average number of solutions in Tables I and II, A3 has an average of 7.0 free solutions versus an average of 4.0 stuck-at-fault solutions.

As expected, the results demonstrate a clear memory advantage for the QBF-based formulations. In fact, both 5-scope and 3-scope QBF formulations are 93% more succinct than the SAT formulation on average. In total, SAT-based fault diagnosis solves 27 out of 54 instances (50%), whereas 5-scope and 3-scope QBF formulations respectively solve 47 (87%) and 52 (96%) out of 54 instances, respective improvements of 74% and 93% in the number of solved instances. Even in cases that are solved completely by all three approaches such as all the problem instances of circuits A2 and A5, QBF-based diagnosis is significantly faster than SAT-based diagnosis.

Fig. 6, which plots the number of solved instances as a function of run-time, clarifies the QBF versus SAT-based comparison. Clearly, both QBF approaches have a sizable advantage over SAT, with the 3-scope fault-diagnosis formulation virtually consistently dominating the 5-scope formulation.

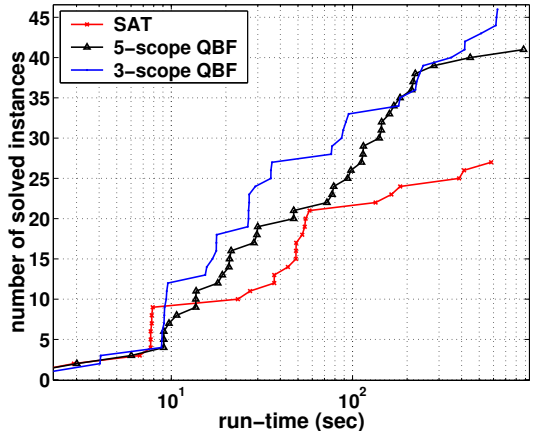


Fig. 6. QBF versus SAT-based fault diagnosis results

V. CONCLUSION

This work presents a memory-efficient QBF-based formulation for fault diagnosis in sequential circuits. Experiments confirm the memory advantage of the approach and demonstrate significant performance improvements for industrial problems. These positive results encourage further research in the application of QBF solvers to diagnosis.

REFERENCES

[1] J. B. Liu and A. Veneris, "Incremental fault diagnosis," in *IEEE Trans. on CAD*, vol. 24, no. 2, 2005, pp. 240–251.

[2] V. Boppana and M. Fujita, "Modeling the unknown! Toward model-independent fault and error diagnosis," in *Int'l Test Conf.*, 1998, pp. 1094–1101.

[3] L. M. Huisman, "Diagnosing arbitrary defects in logic designs using single location at a time (SLAT)," in *IEEE Trans. on CAD*, vol. 23, no. 1, 2004, pp. 91–101.

[4] D. B. Lavo, I. Hartanto, and T. Larrabee, "Multiplets, models, and the search for meaning: Improving per-test fault diagnosis," in *Int'l Test Conf.*, 2002, pp. 250–259.

[5] S. M. Reddy, I. Pomeranz, H. Tang, S. Kajihara, and K. Kinoshita, "On testing of interconnect open defects in combinational logic circuits with stems of large fanout," in *Int'l Test Conf.*, 2002, pp. 83–89.

[6] S. Venkataraman and S. B. Drummonds, "A technique for logic fault diagnosis of interconnet open defects," in *VLSI Test Symp.*, 2000, pp. 313–318.

[7] S.-Y. Huang and K.-T. Cheng, "Errortracer: Design error diagnosis based on fault simulation techniques," *IEEE Trans. on CAD*, vol. 18, no. 9, pp. 1341–1352, 1999.

[8] A. Smith, A. Veneris, M. F. Ali, and A. Viglas, "Fault diagnosis and logic debugging using Boolean satisfiability," *IEEE Trans. on CAD*, vol. 24, no. 10, pp. 1606–1621, 2005.

[9] N. Dershowitz, Z. Hanna, and J. Katz, "Bounded model checking with QBF," in *Int'l Conf. on Theory and Applications of Satisfiability Testing*, 2005, pp. 408–414.

[10] T. Jussila and A. Biere, "Compressing BMC encodings with QBF," in *Intl. Workshop on Bounded Model Checking*, 2006, pp. 1–14.

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

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

[13] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem proving," *Comm. of the ACM*, vol. 5, pp. 394–397, 1962.

[14] A. Biere, "Resolve and expand," in *Int'l Conf. on Theory and Applications of Satisfiability Testing*, 2004, pp. 238–246.

[15] M. Benedetti, "sKizzo: a suite to evaluate and certify QBFs," in *Int'l Conf. on Automated Deduction*, 2005, pp. 369–376.

[16] H. Samulowitz and F. Bacchus, "Binary clause reasoning," in *Int'l Conf. on Theory and Applications of Satisfiability Testing*, 2006, pp. 353–367.

[17] M. Abramovici, M. Breuer, and A. Friedman, *Digital Systems Testing and Testable Design*. Computer Science Press, 1990.

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

[19] OpenCores.org, "http://www.opencores.org," 2006.