Preprint

Exact Logic and Fault Simulation in Presence of Unknowns


doi: 10.1145/2611760

This is the author's "personal copy" of the final, accepted version of the paper published by ACM.

ACM COPYRIGHT NOTICE

© 2014 ACM. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Logic and fault simulation are essential techniques in electronic design automation. The accuracy of standard simulation algorithms is compromised by unknown or X-values. This results in a pessimistic overestimation of X-valued signals in the circuit, and a pessimistic underestimation of fault coverage.

This work proposes efficient algorithms for combinational and sequential logic as well as for stuck-at and transition-delay fault simulation which are free of any simulation pessimism in presence of unknowns. The SAT-based algorithms exactly classify all signal states. During fault simulation, each fault is accurately classified as either undetected, definitely detected or possibly detected.

The pessimism w.r.t. unknowns present in classic algorithms is thoroughly investigated in the experimental results on benchmark circuits. The applicability of the proposed algorithms is demonstrated on larger industrial circuits. The results show that by accurate analysis the number of detected faults can be significantly increased without increasing the test set size.

Categories and Subject Descriptors: B.6.2 [Logic Design]: Reliability and Testing; B.7.3 [Integrated Circuits]: Reliability and Testing

General Terms: Algorithms, Reliability

Additional Key Words and Phrases: Unknown values, simulation pessimism, exact logic simulation, exact fault simulation, SAT

1. INTRODUCTION

Logic and fault simulation are fundamental techniques in electronic design automation with applications, e.g. in validation, test generation and product quality estimation.

Unknown or X-values may emerge during the design and test generation process due to black boxes in the design. During operation and test application, X-values may be caused by uncontrolled sequential elements, at clock domain crossings or A/D boundaries for example. Depending on the circuit and test method, a very high fraction of signals may have X-values. During test and in particular for special test modes such as faster than at-speed test, a high density of X-values has been reported [Wohl et al. 2008], [Ramdas and Sinanoglu 2012].

This work was partially supported by the German Research Foundation (DFG) under grants BE 1176/14-2, WU 245/9-1, and WU 245/11-1. It is an extended version of [Hillebrecht et al. 2012].


Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© YYYY ACM 1084-4309/YYYY/01-ARTA $15.00
DOI: http://dx.doi.org/10.1145/000000.000000

Standard logic and fault simulation algorithms are based on \( n \)-valued logics with a limited number of symbols to denote the signal states in the simulation. Not all X-states, and the correlations between them, are represented accurately. Thus, reconvergences of X-values, where canceling of Xs may occur, are not evaluated correctly and the resulting signal values are not exact. The result may either underestimate the number of X-values as in the case of logic simulation using Verilog models [Turpin 2003], or pessimistically overestimate their number as illustrated below.

The example in Figure 1 shows a circuit with three gates and three inputs. The simulation result of pattern \((a, b, c) = (1, X, 1)\) with a 3-valued logic simulator is annotated to the circuit lines. The signals \(d, e,\) and \(f\) are evaluated to the unknown value \(X\) by the simulator. Simulations with \(b = 0\) and \(b = 1\) show that output \(f\) has the logic value 1 in both cases and thus, its state is known. Three-valued simulation overestimates the number of signals with unknown state.

![Fig. 1: Pessimistic simulation result with a 3-valued logic simulator.](image)

For sequential logic simulation, the problem of simulation pessimism is more pronounced since the number of signals with X-value may even increase over time. Pessimistic sequential simulation may prevent the verification of reset or initialization sequences which target the initialization of a design from an unknown or partially unknown state [Keim et al. 1996].

For fault simulation, well defined logic values are required for fault activation and propagation. In consequence, fault simulation algorithms based on \( n \)-valued logics like the parallel pattern single fault (PPSFP) or the concurrent algorithm [Ulrich and Baker 1988], [Waicukauski et al. 1985], [Antreich and Schulz 1987], [Lee and Ha 1991]\(^1\), pessimistically underestimate the number of detected faults since logic simulation overestimates the number of X-valued signals. The number of potentially detected faults is overestimated which may skew the fault coverage since a fraction of potentially detected faults is often counted as detected in commercial tools [Rudnick et al. 1996]. Both inaccuracies impact product quality and may increase test overhead and cost.

For the example in Figure 1, the input stimuli cannot detect any stuck-at fault in the circuit based on 3-valued analysis. Yet, the actual value of output \(f\) is known and thus, the pattern is indeed a test for the stuck-at 0 fault at \(f\).

If X-values propagate into compaction logic as found in embedded deterministic test (EDT) or built-in self test (BIST) environments, the response signature may be corrupted. X-blocking, X-masking [Naruse et al. 2003], [Tang et al. 2006] or X-tolerant [Mitra and Kim 2004] design-for-test structures try to remedy the problem at increased hardware overhead or reduced response compaction ratio. A pessimistic analysis of X-values further increases this overhead and may cause overmasking of failure data with impact on diagnosability.

\(^1\)In the following they are referred to as 3-valued fault simulators.
This work presents algorithms for exact combinational and sequential logic simulation as well as for exact fault coverage computation for stuck-at and transition faults. The algorithms are free of any simulation pessimism and compute the exact result in presence of X-sources in the circuit. For the example of Figure 1, the proposed fault simulation algorithm also identifies the detection of the stuck-at 0 fault at signal $a$ by the given pattern.

The reduction of the pessimism of logic and fault simulation has been targeted in previous work using heuristics, formal reasoning or a combination thereof.

Heuristic approaches are typically very fast, but they cannot provide the exact result. Proposed methods include circuit analysis like static learning [Kajihara et al. 2004], or partitioning and local exhaustive simulation [Kang and Szygenda 2003]. In restricted symbolic simulation [Carter et al. 1989], the number of symbols to express different X-values is increased, allowing to correctly evaluate a subset of reconvergences of X-valued signals.

The exact logic simulation in presence of X-values corresponds to an NP-complete problem [Chang and Abraham 1987]. The exact solution can be computed by symbolic simulation of a circuit using reduced ordered BDDs (ROBDDs, [Bryant 1986]), which may cause excessive memory consumption for arithmetic or larger circuits. The SAT-based approach of [Kochte and Wunderlich 2011] evaluates each reconvergence of X-valued signals for X-canceling. It provides the exact result for logic simulation, but at high runtimes for larger circuits and many X-sources. In [Chou et al. 2010], satisfiability of quantified Boolean formulae is used to identify the registers in a design that do not need to be initialized and to compute corresponding reset sequences. Reasoning about X-values also gained importance for verification of designs with black boxes. While modeling X-valued signals with 3-valued logic [Jain et al. 2000] only helps to distinguish the signals from these with defined binary values, an exact X-analysis based on symbolic simulation [Wilson et al. 2000], [Scholl and Becker 2001] increases the accuracy of the verification task.

In fault simulation, each fault free and faulty machine has to be analyzed per pattern, causing very high computational effort or excessive memory consumption. Therefore, the pessimism in fault simulation could only be targeted by heuristic or hybrid approaches combining heuristics and formal methods so far. This includes heuristics based on static learning [Kajihara et al. 2004] or restricted symbolic simulation [Kundu et al. 1991]. Hybrid SAT- or BDD-based fault simulation methods limit the application of formal reasoning in space or time: The SAT-based method of [Kochte and Wunderlich 2011] computes the exact result only for the fault-free circuit. The propagation of faults is evaluated pessimistically. The hybrid BDD-based method of [Becker et al. 1999] restricts or even discards BDD-based symbolic simulation when memory consumption exceeds a threshold. In [Kochte et al. 2011], an approximate symbolic fault simulation is proposed which constructs local BDDs limited in size to evaluate local reconvergences of X-states correctly. The result of these approaches is still pessimistic.

The recent progress in SAT solvers enables the exact reasoning about fault detection in presence of X-values even for larger circuits [Hillebrecht et al. 2012]. This work proposes a formal method to exactly compute the stuck-at and transition-delay fault coverage of a test set in presence of X-values. It combines heuristics and SAT reasoning to remove any simulation pessimism found in previous approaches. The state-of-the-art incremental SAT solver antom [Schubert et al. 2010] is used to incrementally build and solve the SAT instances during analysis and reduce runtime.

Section 2 introduces the used terminology and a formal problem statement. The exact logic simulation algorithm is explained in Section 3, followed by the fault sim-
ulation in Section 4. Section 5 presents experimental results on ISCAS benchmark circuits and NXP circuits. Section 6 summarizes the paper.

2. TERMINOLOGY AND OVERVIEW

This section introduces the used terminology and outlines the algorithms for the exact logic as well as exact stuck-at and transition-delay fault classification.

2.1. Terminology and Definitions

In 3-valued logic, the three symbols \( \{0, 1, X\} \) are used to represent logic value 0 (logic-0), logic value 1 (logic-1) and an unknown state, i.e., either logic-0 or logic-1. Signals at which unknown values originate are called X-sources. During logic simulation of a test pattern \( p \), a 3-valued simulator assigns logic-0, logic-1 or X to the signals. Signals with value X for pattern \( p \) belong to the set of Pessimistic-Xs \( \text{PEX}(p) \). \( \text{PEX}(p) \) can be partitioned into the sets of Real-Xs \( \text{REX}(p) \) and False-Xs \( \text{FEX}(p) \). \( \text{FEX}(p) \) contains the signals of \( \text{PEX}(p) \) which are independent from the X-sources, i.e., the signals have a binary value of logic-0 or logic-1. \( \text{REX}(p) \) contains all signals which do depend on at least one X-source. In Figure 1, output \( f \in \text{FEX}(p) \), while \( b, d, e \in \text{REX}(p) \).

These sets differ in the fault free and in the faulty cases. Superscripts \( \text{G} \) and \( f \) are used to distinguish between the good (fault free) and the faulty case, respectively.

In this work, definite detection (DD) and potential detection (PD) of a fault are distinguished. A fault \( f \) is definitely detected (DD) if an observable output \( o \) exists where the fault effect is visible independent of the logic value assignment to the X-sources. Let the functions \( v^G(p, s) \) and \( v^f(p, s) \) return the logic value of signal \( s \) under pattern \( p \) in the fault free and faulty case in presence of unknown values.

The definite detection of a stuck-at-\( \phi \) fault \( f (\phi \in \{0, 1\}) \) at line \( l \) under pattern \( p \) is given as

\[
\text{DD}^f(p) := \exists o \in O : v^G(p, o), v^f(p, o) \in \{0, 1\} \land v^G(p, o) \neq v^f(p, o),
\]

(1)

where \( O \) is the set of output signals of the circuit. If \( f \) is not definitely detected, \( f \) is potentially detected (PD) if the fault is activated and an observable output \( o \) exists where the fault effect can be deterministically measured for at least one logic value assignment to the X-sources:

\[
\text{PD}^f(p) := \neg \text{DD}^f(p) \land v^G(p, l) = \neg \phi \land \exists o \in O : v^G(p, o) \in \{0, 1\} \land o \in \text{REX}^f(p).
\]

(2)

The definite detection of a transition-delay fault \( tf \) at line \( l \) requires the consideration of two cycles. In the first cycle, line \( l \) is driven to a defined value \( \phi \) to activate the fault. For a slow-to-rise transition-delay fault, \( \phi \) is logic-0, for a slow-to-fall fault, \( \phi \) equals logic-1. In the second cycle, the value of \( l \) is inverted and the resulting transition is propagated from \( l \) to an observable circuit output. This corresponds to detecting the stuck-at-\( \phi \) fault at line \( l \) in the propagation cycle. Thus, the definite detection of \( tf \) at line \( l \) under pattern pair \( (p^{-1}, p) \) is given as

\[
\text{DD}^{tf}(p^{-1}, p) := v^G(p^{-1}, l) = \phi \land v^G(p, l) = \neg \phi \land \exists o \in O : v^G(p, o), v^f(p, o) \in \{0, 1\} \land v^G(p, o) \neq v^f(p, o),
\]

(3)

with \( O \) the set of circuit outputs. Similar to the potential detection requirement for stuck-at faults, potential detection of a transition-delay fault requires that the fault is activated and its effect can be deterministically measured in the propagation cycle at
Exact Logic and Fault Simulation in Presence of Unknowns

at least one output $o$ for at least one logic value assignment to the X-sources:

$$\text{PD}^f(p^{-1}, p) := \neg \text{DD}^f(p^{-1}, p) \land v^G(p^{-1}, l) = \phi \land v^G(p, l) = \neg \phi \land \exists o \in O : v^G(p, o) \in \{1, 0\} \land o \in \text{REX}^f(p).$$ (4)

Note that for both, stuck-at and transition-delay faults, 3-valued fault simulation underapproximates the number of definitely detected faults since three-valued simulation overestimates the number of signals with X-values. Consequently, the number of potentially detected stuck-at or transition-delay faults provides an overapproximation.

2.2. Overview of the Exact Logic and Fault Simulation Algorithms

The exact logic simulation algorithm efficiently computes the exact signal states in a combinational circuit by use of heuristics and formal reasoning based on incremental SAT solving. Exact sequential logic simulation is achieved by unrolling the sequential circuit for the number of considered time frames.

Exact fault simulation is performed for stuck-at and transition-delay faults. The proposed fault simulation process is divided into two parts. First, the test pattern set is pessimistically simulated with a parallel pattern single fault propagation simulator based on 3-valued logic to mark as many faults as DD as possible. Afterwards the test pattern set is simulated by the exact stuck-at fault simulator, which performs an exact logic simulation of the fault-free circuit per pattern, and then analyzes the activated faults. The algorithm distinguishes definitely detected (DD), potentially detected (PD) and undetected faults.

2.3. Combinational Expansion of the Sequential Circuit

Both exact sequential logic simulation and exact transition-delay fault simulation require the consideration of multiple clock cycles or time frames. Transition-delay fault simulation requires the modeling of a minimum of two time frames. Combinational expansion of the circuit model is used to create a combinational circuit model representing all required time frames. It is used as a substitute for the original sequential circuit within the simulation. The combinational part of the circuit is duplicated by the number of required time frames i.e. there are instances of the circuit for each time frame considered. The different time frames are connected to each other according to the sequential elements in the circuit and depending on the targeted simulation.

In sequential logic simulation, the value captured by a sequential element is the initial value of the sequential element in the following time frame. This also holds for launch on capture (LOC) resp. broadside transition-delay fault simulation [Savir and Patil 1994]. For these two cases, the input to a sequential element is directly connected to the corresponding input in the next time frame.

In contrast, in the launch on shift (LOS) resp. skewed load transition-delay testing, the value stored in a sequential element in a scan chain is shifted in the following time frame according to the order in the scan chain [Savir and Patil 1993].

Figure 2 shows a sequential circuit and the two possible combinational expansions. The first type of expansion is used for functional sequential simulation as well as fault simulation of LOC tests. The second type is used for fault simulation of LOS tests.

If a sequential element is controllable in a time frame, it serves as a primary input to the respective signal in the time frame. If the value it captures is observable, then it serves as primary output of the corresponding time frame.

3. EXACT COMBINATIONAL AND SEQUENTIAL LOGIC SIMULATION

The exact logic simulation is performed using either the original combinational or the combinational expansion of the circuit under simulation. It consists of two consecutive
steps. In the first step, a restricted symbolic simulator and a 2-valued logic simulator are used as heuristics to classify a high number of REXs, FEXs and FEX candidates at low computational cost. In the second step, the set of FEX candidates is formally analyzed. For the formal proof whether a FEX candidate is a REX or not, the state-of-the-art incremental SAT solver antom [Schubert et al. 2010] is utilized. Figure 3 depicts the flow of the exact logic simulation.

3.1. Heuristic Analysis
In the heuristic analysis the pattern $p$ is simulated using restricted symbolic simulation (RSS, [Carter et al. 1989]) and 2-valued pattern-parallel simulation of randomized assignments to the X-sources to classify as many signals as REX, FEX and FEX candidates as possible. The gates of the circuit are processed in topological order and for each gate, RSS and 2-valued simulation are performed. The identified FEX candidates are later classified using SAT reasoning.

In RSS, for each X-value at the X-sources a unique symbol $X_i$ is introduced in addition to the two symbols for logic-0 and logic-1. Hence, X-values from different X-sources are distinguishable. Furthermore, each X-symbol can be negated. This allows the correct evaluation of simple local reconvergences of X-valued signals and increases
accuracy compared to 3-valued simulators. For the example in Figure 1, RSS correctly computes the output value at \( f \) as logic-1, since the symbol \( X_b \) introduced at X-source \( b \) is correctly tracked at \( d \) as \( \neg X_b \) and at signal \( e \) as \( X_e \). Hence, the reconvergence is exactly evaluated to logic-1. Thus, RSS identifies a subset of \( FEX^G(p) \). In the proposed algorithm, the resulting value of RSS of signal \( s \) and pattern \( p \) is stored in \( v^G(p,s) \).

A subset of \( REX^G(p) \) is efficiently found by a 2-valued pattern-parallel logic simulation. 64 random patterns are generated by assigning randomized values to the X-sources. The signal values are computed in one simulation run. One 64-bit integer \( v = [v_0, \ldots, v_{63}] \) is used to represent the values of each signal. For input \( i \), \( v_i \) is derived from the simulated pattern \( p \) and set to \( v_i = [0, \ldots, 0] \) or \( v_i = [1, \ldots, 1] \) if \( i \) is logic-0 or logic-1, respectively. At X-source \( q \), a randomized 64-bit integer is generated and assigned to \( v_q = [v_q^0, \ldots, v_{63}^q] \), \( v_k^q \in \{0, 1\}, 0 \leq k \leq 63 \). \( v_q \) is used for the evaluation of the direct fanout of \( q \).

After finishing both simulations, each signal is classified as logic-0, logic-1 or REX, FEX or FEX candidate. If RSS derived a logic value, the signal does not need to be considered in the subsequent steps. If an unknown value is calculated for \( s \), the values of \( v_s = [v_s^0, \ldots, v_{63}^s] \) of the pattern-parallel simulation is taken into account. If at least one pair of values \( v_s^0, v_s^1 (0 \leq k, l \leq 63) \) has complementary values, the signal \( s \) belongs to \( REX^G(p) \). If all \( v_s^k \) bit are equal, \( s \) is marked as FEX candidate. The classification of these signals is done with an incremental SAT-solver as explained in the next section.

### 3.2. Classification of Remaining FEX Candidates

The FEX candidates computed in the previous step for pattern \( p \) are exactly classified by use of an incremental SAT solver. Input to the SAT solver is a Boolean formula in conjunctive normal form (CNF) which maps the classification of a signal to a Boolean satisfiability problem.

For each FEX candidate \( s \) it is already known that all 64 random assignments to the X-sources force \( s \) to value \( v_s^k (0 \leq k \leq 63) \) of either logic-0 or logic-1. Signal \( s \) is a FEX, if and only if it can be proven that \( s \) cannot have the complementary value \( \neg v_s^k \) for any assignment to the X-sources. Thus, the Boolean formula is constructed such that it is satisfiable, if and only if \( s \) can be driven to \( \neg v_s^k \). If the formula is satisfiable, \( s \) depends on the X-sources and is classified as REX. Otherwise \( s \) is independent of the X-sources and classified as FEX.

In the following we provide additional details on the generation of the SAT instances. The FEX candidates are evaluated starting from the X-sources in topological order. To increase efficiency, the SAT instance is extended incrementally for each FEX candidate exploiting the result from the simulation step as well as learnt knowledge from analysis of previous FEX candidates.

To check whether \( s \) can be driven to \( \neg v_s^k \), the characteristic equations of the gates in the adjustment cone, resp. transitive fanin, of \( s \) are translated into CNF and added to the SAT instance. This is done using the Tseitin transformation [Tseitin 1968]. The size of the resulting SAT instance is reduced by only considering the gates which have been classified as REX or FEX candidate for pattern \( p \). The CNF for the adjustment cone of a signal \( s \) is created recursively as outlined in Algorithm 1.

This SAT instance is extended by a temporary unit clause with only one literal (called assumption) for FEX candidate \( s \) which constrains the value of \( s \) in the search process of the SAT solver. If the value of \( s \) in the pattern parallel simulation was \( v_s = [0, \ldots, 0] \), the assumption \( \{s\} \) is added to constrain the SAT search to assignments to the X-sources which imply \( s \) to logic-1. If the instance is satisfiable, \( s \) belongs to the set REX. Otherwise \( s \) is a FEX with value logic-0 and \( v^G(p,s) \) is updated. In the latter case, the unit clause \( \{-s\} \) is added permanently to the SAT instance to reduce
runtime for subsequent calculations of the SAT solver. Correspondingly, if the value of $s$ in the pattern parallel simulation was $v_s = [1, \ldots, 1]$, the assumption $\{\neg s\}$ is added.

**ALGORITHM 1:** CNF creation of the adjustment/fanin cone.

```plaintext
//addSignalToCNF(s, CNF)
G ← getDrivingGateOf(s);
if G already Tseitin transformed then
    return;
end
if $v^G(p, s) = 0$ then
    //Exploit knowledge from RSS
    CNF $\cup \{\neg s\}$;
    return;
end
if $v^G(p, s) = 1$ then
    //Exploit knowledge from RSS
    CNF $\cup \{s\}$;
    return;
end
CNF $\cup$ getTseitinTransformation(G);
forall the Inputs $s_i$ of gate G do
    addSignalToCNF($s_i$, CNF);
end
```

For the classification of the next FEX candidate $s'$ in topological order, the CNF instance is extended incrementally to include the adjustment cone of $s'$, i.e., only the clauses for gates which are not yet Tseitin transformed are added.

During exact simulation, the algorithm maintains a lookup table derived from the result of the RSS step. The table contains the information if a symbol for an X-state assigned to signals during RSS is a logic-0, a logic-1 or a REX. Before analyzing a FEX candidate $s$ using the SAT technique, a fast lookup is performed to check whether the corresponding symbol $X_s$ has already been computed. If the classification for $X_s$ is already known, $s$ is set to the corresponding state. Otherwise, $s$ is classified as described above. This effectively restricts the use of the SAT solver to signals at which REX values converge.

4. EXACT STUCK-AT AND TRANSITION-DELAY FAULT SIMULATION

The exact stuck-at fault simulation classifies a set of target faults as definitely detected (DD), potentially detected (PD) or undetected for a test set in presence of unknowns. It uses the heuristics and formal SAT reasoning explained in the previous section.

An overview of the fault simulation of a pattern $p$ is given in Figure 4. 3-valued fault simulation is used to mark as many target faults as possible as DD. For the remaining faults, an exact analysis is conducted.

The exact analysis starts with the exact logic simulation of the fault free circuit for pattern $p$ to compute the set of activated faults. These faults are then analyzed serially. For the fault simulation of an activated fault $f$, $f$ is injected into the circuit model. The algorithm then proceeds in two phases similar to the fault free approach: A heuristic simulation and an exact calculation step. During the simulation step the behavior of the faulty circuit is simulated in event-driven manner by RSS and 2-valued pattern-parallel logic simulation which evaluates random assignments to the X-sources. If the results of the simulations allow the fault classification as DD or undetected, further
Fig. 4: Exact fault simulation for a pattern $p$ and classification as definitely detected (DD) or potentially detected (PD).

analysis is not required. Otherwise, the SAT solver is invoked for analysis of the outputs of the faulty circuit. Internal signals in the faulty circuit do not need to be considered since the values at observable outputs are sufficient to reason about fault detection.

4.1. Fault Analysis by RSS and Pattern-Parallel Simulation

For an activated fault $f$, the circuit outputs $o_1, \ldots, o_k$ in the propagation cone, resp. transitive fanout, of $f$ are analyzed using the results of the faulty circuit simulations. According to Section 2.1, we only consider outputs $o_i$ which have a defined value in the fault free circuit $v^G(p,o_i) \in \{0,1\}$.

If there is one output $o_i$ with a defined value in the faulty case $v^f(p,o_i) \in \{0,1\}$ according to RSS, and $v^f(p,o_i) \neq v^G(p,o_i)$, then $f$ is marked as DD and the algorithm proceeds with the next fault. If all outputs in the propagation cone have defined values equal to the fault free case, i.e., $v^f(p,o_i) \in \{0,1\}$ and $v^f(p,o_i) = v^G(p,o_i)$ for $1 \leq i \leq k$, then $f$ is undetected by the pattern and the algorithm continues with the next fault.

Otherwise, the outputs are divided into three sets: Potential detect outputs $O_{PD}$, possibly definitive detect outputs $O_{possDD}$, and possibly potential detect outputs $O_{possPD}$. The set $O_{PD}$ will contain all outputs at which fault $f$ can be potentially detected. An output $o_i$ is added to the set $O_{PD}$ if the faulty value $v^f(o_i)$ is not equal to $[0,\ldots,0]$ or $[1,\ldots,1]$. Note that these outputs are elements of the set $\text{REX}^f(p)$.

For an output for which RSS derived an X symbol and $v^f(o_i)$ equals either $[0,\ldots,0]$ or $[1,\ldots,1]$, it is not known whether it belongs to $\text{REX}^f(p)$ or $\text{FEX}^f(p)$. A later exact analysis will determine its state. If all $v^f_j$ ($0 \leq j \leq 63$) are equal to $\neg v^G(p,o_i)$, $o_i$ is added to $O_{possDD}$ since it may be an output at which the fault can be definitely detected. If the exact analysis later reveals that $o_i$ is a FEX, then $f$ is a DD, otherwise $f$ is a PD.

On the other hand, if all $v^f_j$ ($0 \leq j \leq 63$) are equal to $v^G(p,o_i)$, $o_i$ is added to $O_{possPD}$ since it may be an output at which the fault can be potentially detected. If the exact analysis reveals that $o_i$ is a REX, then $f$ is a PD, otherwise $f$ cannot be detected at $o_i$ at all.
4.2. Fault Classification by SAT Reasoning

If the set $O_{\text{possDD}}$ is not empty, the output values in the faulty circuit are iteratively derived using the incremental SAT solver. This is similar to the fault free case. A SAT instance is constructed which is satisfiable iff the considered output is a REX (see Section 3.2). If output $o_i$ belongs to $\text{REX}^f(p)$, $o_i$ is removed from $O_{\text{possDD}}$ and added to $O_{\text{PD}}$. In the other case, the fault is marked as DD, because

$$v^G(p, o_i), v^f(p, o_i) \in \{0, 1\} \land v^G(p, o_i) = \neg v^f(p, o_i)$$

(5)

is true. Thus, the fault is detected for all logic value assignments to the X-sources. Then the next stuck-at fault is analyzed.

If $O_{\text{possDD}}$ is empty and $O_{\text{PD}}$ is not empty, the stuck-at fault is marked as PD and the algorithm proceeds with the next stuck-at fault.

If the current fault is neither marked DD nor PD and $O_{\text{possPD}}$ is not empty, the SAT solver is used to determine if one of the outputs in $O_{\text{possPD}}$ belongs to $\text{REX}^f(p)$. Note that this step is performed only if the fault is not yet marked as PD. If one output of $O_{\text{possPD}}$ is member of $\text{REX}^f(p)$, the fault is marked as PD. In the case that all outputs in $O_{\text{possPD}}$ belong to $\text{FEX}^f(p)$, the fault remains unmarked and undetected.

4.3. Extension to Transition-Delay Fault Simulation

The exact transition-delay fault simulation requires the combinational expansion of the sequential circuit model as outlined in section 2.3, and an extension of the fault simulation algorithm introduced above.

According to the definition in Section 2.1, a transition-delay fault is detected if

— the activation value $\phi$ stipulated by the fault model is justified at the faulty signal line in the activation cycle and
— the stuck-at fault which describes the behavior of the considered transition-delay fault is detectable in the succeeding propagation cycle.

The exact fault simulation algorithm is extended such that before simulating the stuck-at fault in the propagation time frame, it is checked whether the faulty signal has the value $\phi$ in the fault free case in the activation time frame. If the faulty signal has the value $\neg \phi$ in the activation cycle, the fault is marked as undetected under the current pattern pair.

If the faulty signal has the activation value $\phi$, the stuck-at fault in the succeeding time frame is analyzed as described above. If the stuck-at fault is detected, the transition-delay fault is also marked as detected. If the stuck-at fault is potentially detected in the propagation time frame, the transition-delay fault is marked as PD. It is marked undetected if the stuck-at fault is not detected.

5. EXPERIMENTAL RESULTS

The presented algorithm has been applied to ISCAS benchmarks and large industrial circuits from NXP. The experiments were run on an Intel Xeon CPU with 3.3 GHz. The following two sections discuss the pessimism of classical combinational and sequential logic simulation algorithms. Section 5.3 discusses the increase in fault coverage in exact fault simulation, the trade-off between runtime and accuracy by use of timeouts, and the impact of clustered X-sources.

5.1. Reduction of Unknown Output Values

The exact logic simulation algorithm of Section 3 efficiently computes the exact output values of the circuit for a test set. This is in particular important for BIST and
EDT environments to avoid unnecessary DFT overhead for X-masking or X-blocking structures, and overmasking of FEX-valued outputs.

For the considered circuits modeling one time frame, five simulation runs are performed and averaged. In each run, a fixed percentage of the controllable circuit inputs is randomly selected as X-sources (X-ratio). Then, a test set of 1000 random patterns is analyzed. The difference in the number of PEX outputs of a 3-valued simulation, i.e. the number of X-outputs due to a standard 3-valued simulation run and the REX outputs of the exact analysis is compared.

Figure 5 a) shows the reduction of the number of outputs with X-value for ISCAS’85 circuit c7552 for different X-ratios when the exact algorithm is used (in red). The number of X-valued outputs is reduced by more than 25% for the X-source scenarios with 1% and 7% X-sources. The reduction decreases to 0% if nearly all inputs are X-sources. The reduction of X-valued outputs is not monotonously falling as different X-ratios and input patterns may lead to different numbers of reconvergences within the circuit and therefore also to different numbers of outputs showing an unknown value. Hence, it may happen that an increased X-ratio leads to additional reconvergences of X-valued signals. This may cause X-canceling at the reconvergences and a reduction of outputs with X-value compared to 3-valued simulation. The figure also shows the reduction of X-valued outputs for restricted symbolic simulation (RSS, in blue) and the approximate combinational BDD-based method of [Kochte et al. 2011] (in black): Restricted symbolic simulation and the approximate method of [Kochte et al. 2011] are able to reduce the pessimism of 3-valued combinational logic simulation and compute better lower bounds for X-propagation in the circuit, but in general they fail to provide the exact result.

![Figure 5a: Reduction of unknown output values of ISCAS circuit c7552](image1)

![Figure 5b: Pessimism in sequential simulation for 100 cycles](image2)

Fig. 5: Pessimism in classical combinational and sequential logic simulation.

Similar experiments have been conducted for other circuits as well. In Table I, we present the reduction of X-valued outputs of the proposed exact method for the case of 5% X-sources. Column ‘Circuit’ contains the circuit name. Column ‘PEX’ and ‘REX’...
show the absolute number of unknown values at the outputs for the test set computed by 3-valued simulation respectively the exact algorithm. In a BIST architecture, only these REX outputs have to be masked for the computation of a signature. The last column in the table contains the reduction of X-values at the circuit outputs. In average, the number of X-values is reduced by 20.2%.

Table I: Reduction of X-values at the outputs for 1000 test patterns and 5% X-sources.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Gates</th>
<th>Outputs</th>
<th>PEX</th>
<th>REX</th>
<th>Reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c6288</td>
<td>2416</td>
<td>23387</td>
<td>15215</td>
<td>34.9</td>
<td></td>
</tr>
<tr>
<td>c7552</td>
<td>3512</td>
<td>11826</td>
<td>9387</td>
<td>20.6</td>
<td></td>
</tr>
<tr>
<td>s38417</td>
<td>22179</td>
<td>143195</td>
<td>122760</td>
<td>14.3</td>
<td></td>
</tr>
<tr>
<td>s38584</td>
<td>19253</td>
<td>95323</td>
<td>91877</td>
<td>3.6</td>
<td></td>
</tr>
<tr>
<td>p77k</td>
<td>68146</td>
<td>248242</td>
<td>195593</td>
<td>21.2</td>
<td></td>
</tr>
<tr>
<td>p78k</td>
<td>74243</td>
<td>685605</td>
<td>454267</td>
<td>33.7</td>
<td></td>
</tr>
<tr>
<td>p81k</td>
<td>88741</td>
<td>491965</td>
<td>384691</td>
<td>21.8</td>
<td></td>
</tr>
<tr>
<td>p88k</td>
<td>83538</td>
<td>309047</td>
<td>255373</td>
<td>17.4</td>
<td></td>
</tr>
<tr>
<td>p100k</td>
<td>90712</td>
<td>325348</td>
<td>277661</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td>p141k</td>
<td>163568</td>
<td>1154394</td>
<td>919633</td>
<td>20.3</td>
<td></td>
</tr>
<tr>
<td>p267k</td>
<td>262451</td>
<td>1284472</td>
<td>1124916</td>
<td>12.4</td>
<td></td>
</tr>
<tr>
<td>p286k</td>
<td>337048</td>
<td>1172617</td>
<td>758732</td>
<td>35.3</td>
<td></td>
</tr>
<tr>
<td>p295k</td>
<td>274872</td>
<td>2096184</td>
<td>2102686</td>
<td>16.1</td>
<td></td>
</tr>
<tr>
<td>p330k</td>
<td>330014</td>
<td>1583657</td>
<td>1322674</td>
<td>16.5</td>
<td></td>
</tr>
<tr>
<td>p378k</td>
<td>371215</td>
<td>3637222</td>
<td>2352902</td>
<td>35.3</td>
<td></td>
</tr>
<tr>
<td>p388k</td>
<td>456964</td>
<td>2326166</td>
<td>1371057</td>
<td>41.1</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>999916</td>
<td>734964</td>
<td>26.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
averaged. In each run, a fixed percentage of the controllable circuit inputs is randomly selected as X-sources. Then, the fault coverage of 1 000 random patterns is computed using 3-valued fault simulation and the proposed exact algorithm.

For circuit c7552, Figure 6 depicts the increase in fault coverage of the exact algorithm w.r.t. 3-valued fault simulation for different X-ratios, and the runtime in seconds. The data points indicate the increase of fault coverage if 1 000 test patterns are analyzed exactly. The exact algorithm increases fault coverage by up to 14.2%. The highest increase of fault coverage is achieved when approximately 10% of the inputs are X-sources.

The figure also shows the increase of fault coverage for three approximate fault simulation algorithms: The hybrid SAT-based method of [Kochte and Wunderlich 2011], the BDD-based algorithm of [Kochte et al. 2011], and fault simulation based on restricted symbolic simulation (RSS). Compared with these approximate methods, the exact algorithm reveals that a significant number of additional faults are actually detectable with the simulated test set.

The runtime of the proposed algorithm reaches the maximum of 45s at an X-ratio of about 35%. Compared to the method of [Kochte and Wunderlich 2011] with a runtime of 2 358s, the proposed algorithm is $52.4 \times$ faster. For small X-ratios, the runtime is low since RSS uncovers many FEXs at simple X-reconvergences. If the SAT solver is required, the size of the SAT instance is small. For high X-ratios, the pattern parallel simulation of random assignments to X-sources determines most of the REX signals.

Table II reports the results for stuck-at faults considering a larger set of ISCAS and industrial circuits. Due to limited space, the results are limited to the case of 5% X-sources. For each circuit, the table shows the absolute number of stuck-at faults. Column ‘3-val. Fsim. DD’ shows the absolute number of detected faults and the fault coverage in % of 3-valued fault simulation.

The number of additionally detected faults and fault coverage increase by the exact algorithm according to equation (1) is given in column ‘$\Delta$ Exact sim. DD.’ Column ‘Exact sim. PD’ lists the number and ratio of faults marked as potential detect (PD)
Table II: Detected stuck-at faults by a test pattern set with 1 000 patterns and 5% X-sources.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Num. of faults</th>
<th>3-val. Fsim. DD</th>
<th>Δ Exact sim. DD</th>
<th>Exact sim. PD</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Num. F.C.(%)</td>
<td>Num. F.C.(%)</td>
<td>Num. F.C.(%)</td>
<td>(s)</td>
<td></td>
</tr>
<tr>
<td>c6288</td>
<td>8704</td>
<td>4453</td>
<td>51.16</td>
<td>3009</td>
<td>43.77</td>
</tr>
<tr>
<td>c7552</td>
<td>9756</td>
<td>6222</td>
<td>63.77</td>
<td>597</td>
<td>6.12</td>
</tr>
<tr>
<td>s38417</td>
<td>56325</td>
<td>44535</td>
<td>79.07</td>
<td>282</td>
<td>0.50</td>
</tr>
<tr>
<td>s38584</td>
<td>53845</td>
<td>43459</td>
<td>80.71</td>
<td>161</td>
<td>0.30</td>
</tr>
<tr>
<td>p77k</td>
<td>186645</td>
<td>87337</td>
<td>46.72</td>
<td>1151</td>
<td>0.62</td>
</tr>
<tr>
<td>p78k</td>
<td>225476</td>
<td>189586</td>
<td>84.08</td>
<td>18602</td>
<td>8.25</td>
</tr>
<tr>
<td>p81k</td>
<td>272322</td>
<td>141953</td>
<td>52.13</td>
<td>4205</td>
<td>1.54</td>
</tr>
<tr>
<td>p89k</td>
<td>228888</td>
<td>124090</td>
<td>54.21</td>
<td>3571</td>
<td>1.56</td>
</tr>
<tr>
<td>p100k</td>
<td>247376</td>
<td>173073</td>
<td>69.96</td>
<td>4297</td>
<td>1.74</td>
</tr>
<tr>
<td>p141k</td>
<td>434363</td>
<td>334535</td>
<td>77.02</td>
<td>13501</td>
<td>3.11</td>
</tr>
<tr>
<td>p267k</td>
<td>640221</td>
<td>449787</td>
<td>70.26</td>
<td>8404</td>
<td>1.31</td>
</tr>
<tr>
<td>p295k</td>
<td>728246</td>
<td>410393</td>
<td>56.35</td>
<td>61897</td>
<td>8.50</td>
</tr>
<tr>
<td>p330k</td>
<td>840690</td>
<td>540257</td>
<td>64.26</td>
<td>32143</td>
<td>3.82</td>
</tr>
<tr>
<td>p378k</td>
<td>1127364</td>
<td>933235</td>
<td>82.79</td>
<td>101519</td>
<td>9.00</td>
</tr>
<tr>
<td>p388k</td>
<td>1287689</td>
<td>1016295</td>
<td>78.92</td>
<td>61841</td>
<td>4.80</td>
</tr>
<tr>
<td>Total</td>
<td>6347910</td>
<td>4449172</td>
<td>70.88</td>
<td>315981</td>
<td>4.98</td>
</tr>
</tbody>
</table>

The lower bound on fault coverage computed by 3-valued fault simulation ranges from 46.72 to 84.08%, with an average of 70.88%. The exact fault simulation proves that in average an additional 4.98% of the faults are detected by the test sets. The increase in additionally detected faults is very high for the multiplier c6288 due to high signal observability and propagation of many X-values in the pessimistic simulation. The results also show that on average 5.79% of the stuck-at faults are classified as potential detect. The runtime of the algorithm for the considered ratio of X-sources ranges from 0.4 milliseconds up to 210 seconds for a single pattern.

Table III additionally reports the results for transition-delay faults considering ISCAS’89 and NXP circuits. Like for Table II, the columns show the absolute number of detected faults and fault coverage for 3-valued simulation, the number and coverage of the additionally detected faults by the proposed accurate simulation as well as the number and coverage of potentially detected faults.

For transition-delay faults, the lower bound on the fault coverage computed by 3-valued fault simulation ranges between 27.25 and 78.21%. In average, 3.76% of the faults are marked as potentially detected. With the proposed exact fault simulation, 6.48% of the faults are additionally definitely detected in average. The runtime for the exact analysis is notably smaller although two time frames have to be considered. This is mainly because for many faults the complex output classification (c.f. formula (3) in Section 2.1 and Figure 4) is skipped if the fault is not activated in the first time frame.

For all conducted tests, the allocated memory of the proposed approach was below 3.5 GiBytes. The overall runtime of the exact stuck-at fault simulation is in the order of classical 3-valued ATPG, but the proposed algorithm is able to classify faults as detected for which classical 3-valued ATPG fails to generate a test pattern. For circuit p100k, the runtime is 23% less than the runtime of commercial ATPG, while for the largest circuit p388k, the runtime is about 7× larger than the ATPG runtime. Compared to a 3-valued fault simulation, the overall runtime is 130× larger on average. Considering transition-delay faults, the differences in runtime shrink notably. On av-
Table III: Detected transition-delay faults by a test pattern set with 1000 patterns and 5% X-sources.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Num. of faults</th>
<th>3-val. Fsim. DD</th>
<th>∆ Exact sim. DD</th>
<th>Exact sim. DD</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Num. F.C. (%)</td>
<td>Num. F.C. (%)</td>
<td>Num. F.C. (%)</td>
<td>(s)</td>
</tr>
<tr>
<td>s38417</td>
<td>68 744</td>
<td>44 168</td>
<td>64.25</td>
<td>2.06</td>
<td>2.002</td>
</tr>
<tr>
<td>s38584</td>
<td>66 504</td>
<td>46 382</td>
<td>69.74</td>
<td>0.28</td>
<td>1 254</td>
</tr>
<tr>
<td>p77k</td>
<td>237 550</td>
<td>64 734</td>
<td>27.25</td>
<td>1 458</td>
<td>1589</td>
</tr>
<tr>
<td>p78k</td>
<td>308 208</td>
<td>241 034</td>
<td>78.21</td>
<td>39 025</td>
<td>8 026</td>
</tr>
<tr>
<td>p81k</td>
<td>374 456</td>
<td>120 305</td>
<td>32.13</td>
<td>18 075</td>
<td>19 145</td>
</tr>
<tr>
<td>p89k</td>
<td>299 028</td>
<td>82 583</td>
<td>27.62</td>
<td>4 520</td>
<td>6 660</td>
</tr>
<tr>
<td>p100k</td>
<td>324 586</td>
<td>125 323</td>
<td>38.61</td>
<td>5 175</td>
<td>14 677</td>
</tr>
<tr>
<td>p141k</td>
<td>550 868</td>
<td>267 492</td>
<td>48.56</td>
<td>31 054</td>
<td>18 473</td>
</tr>
<tr>
<td>p267k</td>
<td>788 014</td>
<td>298 429</td>
<td>37.87</td>
<td>14 592</td>
<td>39 000</td>
</tr>
<tr>
<td>p295k</td>
<td>943 626</td>
<td>244 877</td>
<td>25.95</td>
<td>31 557</td>
<td>22 222</td>
</tr>
<tr>
<td>p330k</td>
<td>1 054 568</td>
<td>440 993</td>
<td>41.82</td>
<td>5 082</td>
<td>55 053</td>
</tr>
<tr>
<td>p378k</td>
<td>1 541 024</td>
<td>1 188 930</td>
<td>77.15</td>
<td>197 206</td>
<td>44 544</td>
</tr>
<tr>
<td>p388k</td>
<td>1 615 722</td>
<td>914 186</td>
<td>56.58</td>
<td>134 228</td>
<td>74 013</td>
</tr>
<tr>
<td>Total</td>
<td>8 172 898</td>
<td>4 079 437</td>
<td>49.91</td>
<td>529 315</td>
<td>307 708</td>
</tr>
</tbody>
</table>

...erage, the runtime of the accurate approach is $7 \times$ higher compared to 3-valued fault simulation.

5.3.1. Runtime Reduction by Applying a Timeout. As stated before, the runtime can be traded off with the accuracy by introducing a timeout for each invocation of the SAT-solver during the simulation. Considering stuck-at fault simulation for the larger circuit p388k (p378k) and a timeout of 5 seconds per SAT-solver invocation, the runtime reduces to 173 484 s (120 750s). This is 14% (29.0%) less compared to the accurate solution. The number of additionally detected faults does not change, but the number of potentially detected faults decreases by 6.42% (13.2%). A more aggressive timeout of 1 second for circuit p388k (p378k) reduces the runtime further to 169 866 s (105 276 s). This reduces the number of additionally detected faults by 2 (213, 0.21%), and the potentially detected faults by 7.94% (20.0%) compared to the accurate result.

5.3.2. Clustered X-Sources. In order to evaluate the impact of clustered X-sources, additional stuck-at fault simulation experiments are conducted for NXP circuits in which the X-sources are clustered. In the first experiment, the scan cells of one, two or three randomly selected scan-chains are chosen as X-sources. The results are averaged over 5 runs per circuit and X-source configuration. Table IV presents for the different numbers of scan-chains (SC) selected as X-sources the percentage of flip-flops (FF) which generate X-values in column 4, and the results of fault simulation in columns 5 to 7.

For circuit p100k and one scan-chain selected as X-source, i.e. 7.66% of the flip-flops generate X-values, fault coverage increases by 2.41% compared to classical 3-valued simulation. Limiting the number of X-sources to 5% (selecting a consecutive part of the scan-chain), the fault coverage increases by 2.11%, which is 21.3% more than in the case of 5% randomly selected X-sources (cf. Table II).

In the second experiment, the X-sources are clustered by the input signal name: From the inputs sorted by name, a consecutive subset of 5% of the inputs is selected as X-sources, and fault simulation is performed with 1000 random patterns. The results are averaged over 5 runs per circuit and X-source configuration. For circuit p100k, the fault coverage increase is 2.41% compared to 3-valued simulation, which is 38.5% more than in the case of randomly selected X-sources. A similar result is observed for...
Table IV: Detected stuck-at faults by a test pattern set with 1000 patterns for different scan-chain configurations as X-sources.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Num. of faults</th>
<th>X-Sources</th>
<th>3-val. Fsim.</th>
<th>Δ Exact sim. Fsim.</th>
<th>Exact sim. PD (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Num. SC FF in %</td>
<td>DD (%)</td>
<td>DD (%)</td>
<td>PD (%)</td>
<td></td>
</tr>
<tr>
<td>p77k</td>
<td>186 645</td>
<td>8.19</td>
<td>43.05</td>
<td>1.02</td>
<td>4.01</td>
</tr>
<tr>
<td></td>
<td>15.16</td>
<td>37.13</td>
<td>1.43</td>
<td>6.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>23.65</td>
<td>30.17</td>
<td>1.41</td>
<td>7.32</td>
<td></td>
</tr>
<tr>
<td>p78k</td>
<td>225 476</td>
<td>2.03</td>
<td>93.06</td>
<td>0.44</td>
<td>0.62</td>
</tr>
<tr>
<td></td>
<td>3.75</td>
<td>89.17</td>
<td>0.67</td>
<td>1.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.68</td>
<td>83.24</td>
<td>1.03</td>
<td>1.56</td>
<td></td>
</tr>
<tr>
<td>p100k</td>
<td>247 375</td>
<td>7.66</td>
<td>63.71</td>
<td>2.41</td>
<td>9.64</td>
</tr>
<tr>
<td></td>
<td>16.28</td>
<td>45.42</td>
<td>4.97</td>
<td>16.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>19.21</td>
<td>35.34</td>
<td>5.90</td>
<td>16.87</td>
<td></td>
</tr>
</tbody>
</table>

circuit p141k, where fault coverage increases by 5.44%, which is 74.9% more than for randomly selected X-sources.

The experiments indicate that for clustered X-sources, exact logic and fault simulation as proposed here yield even better results than for randomly selected X-sources since clustering further increases the pessimism in classical simulation algorithms.

6. CONCLUSIONS

The work presented the first stuck-at and transition-delay fault simulator, which is able to calculate the exact fault coverage of a test pattern set in the presence of unknown values. The simulator employs logic and restricted symbolic simulation to classify as many signal states as possible without invoking formal SAT reasoning. Incremental SAT solving is utilized only to exactly analyze the remaining signal states. The usage and runtime of the SAT-solver and the size of the CNF formulae are strongly reduced by considering the simulation results and employing incremental SAT techniques. The runtime can also be traded off against accuracy by use of timeouts. The algorithm is able to handle large industrial circuits. The results show that in presence of unknown values, fault coverage is significantly increased by an accurate analysis, without increasing the number of test patterns.

REFERENCES


