Exact Logic and Fault Simulation in Presence of Unknowns

Erb, Dominik; Kochte, Michael A.; Sauer, Matthias; Hillebrecht, Stefan; Schubert, Tobias; Wunderlich, Hans-Joachim; Becker, Bernd

ACM Transactions on Design Automation of Electronic Systems (TODAES) Vol. 19(3)
June 2014

doi: http://dx.doi.org/10.1145/2611760

Abstract: 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 that are free of any simulation pessimism in presence of unknowns. The SAT-based algorithms exactly classifiy all signal states. During fault simulation, each fault is accurately classified as either undetected, definitely detected, or possibly detected. The pessimism with respect to 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.

Preprint

Exact Logic and Fault Simulation in Presence of Unknowns

DOMINIK ERB, University of Freiburg
MICHAEL A. KOCHTE, University of Stuttgart
MATTHIAS SAUER, University of Freiburg
STEFAN HILLEBRECHT, University of Freiburg
TOBIAS SCHUBERT, University of Freiburg
HANS-JOACHIM WUNDERLICH, University of Stuttgart
BERND BECKER, University of Freiburg

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 classifiy 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].
Author's addresses: D. Erb, M. Sauer, S. Hillebrecht, T. Schubert and B. Becker, University of Freiburg, Georges-Köhler-Allee 051, 79110 Freiburg, Germany; M. A. Kochte and H.-J. Wunderlich, ITI, University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Germany.

DOI:http://dx.doi.org/10.1145/0000000.0000000

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 $(a, b, c) = (1, \mathrm{X}, 1)$ 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.

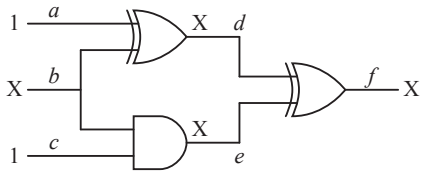


Fig. 1: Pessimistic simulation result with a 3-valued logic simulator.

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 $\{0, 1, \mathrm{X}\}$ 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 $\mathrm{PEX}(p)$. $\mathrm{PEX}(p)$ can be partitioned into the sets of Real-Xs $\mathrm{REX}(p)$ and False-Xs $\mathrm{FEX}(p)$. $\mathrm{FEX}(p)$ contains the signals of $\mathrm{PEX}(p)$ which are independent from the X-sources, i. e., the signals have a binary value of logic-0 or logic-1. $\mathrm{REX}(p)$ contains all signals which do depend on at least one X-source. In Figure 1, output $f \in \mathrm{FEX}(p)$, while $b, d, e \in \mathrm{REX}(p)$.

These sets differ in the fault free and in the faulty cases. Superscripts 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-ϕ fault f ($\phi \in \{0, 1\}$) at line l under pattern p is given as

$$\mathrm{DD}^f(p) := \exists o \in O : v^G(p, o), v^f(p, o) \in \{0, 1\} \wedge v^G(p, o) \neq v^f(p, o), \tag{1}$$

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:

$$\begin{aligned}\mathrm{PD}^f(p) := \neg\mathrm{DD}^f(p) \wedge v^G(p, l) = \neg\phi \wedge \\ \exists o \in O : v^G(p, o) \in \{0, 1\} \wedge o \in \mathrm{REX}^f(p).\end{aligned} \tag{2}$$

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 ϕ to activate the fault. For a slow-to-rise transition-delay fault, ϕ is logic-0, for a slow-to-fall fault, ϕ 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-ϕ 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

$$\begin{aligned}\mathrm{DD}^{tf}(p^{-1}, p) := v^G(p^{-1}, l) = \phi \ \wedge \ v^G(p, l) = \neg\phi \wedge \\ \exists o \in O : v^G(p, o), v^{tf}(p, o) \in \{1, 0\} \wedge v^G(p, o) \neq v^{tf}(p, o),\end{aligned} \tag{3}$$

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

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

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

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 fault 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

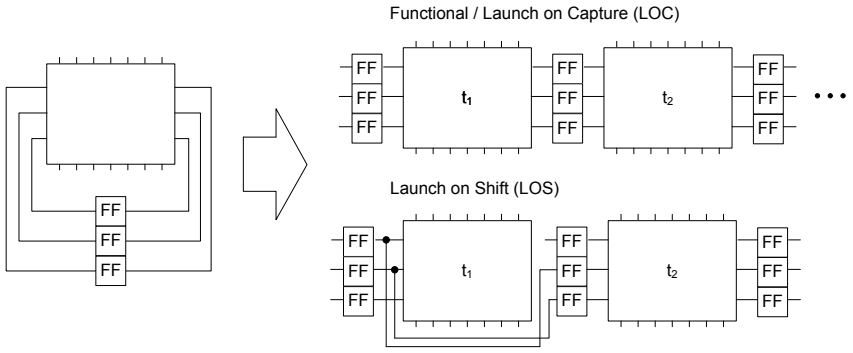


Fig. 2: Combinational expansion for (a) exact sequential simulation and LOC-based transition-delay fault simulation and (b) LOS-based transition-delay fault simulation.

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.

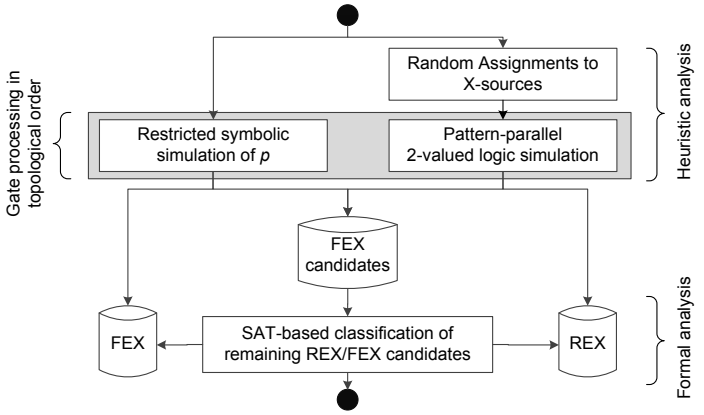


Fig. 3: Exact fault free simulation for a pattern p.

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_b. 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 $v_q = [v_q^0, \ldots, v_q^{63}], v_q^k \in \{0,1\}, 0 \leq k \leq 63$. 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 $v_s = [v_s^0, \ldots, v_s^{63}]$ of the pattern-parallel simulation is taken into account. If at least one pair of values $v_s^k, v_s^l (0 \leq k, l \leq 63)$ 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 $\{\neg s\}$ 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.

```
//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 ∪ {¬s};
    return;
end
if v^G(p, s) = 1 then
    //Exploit knowledge from RSS
    CNF ∪ {s};
    return;
end
CNF ∪ 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

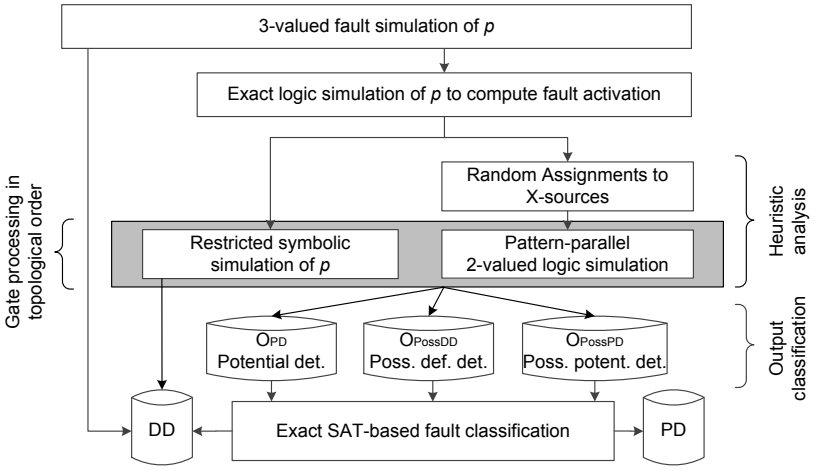


Fig. 4: Exact fault simulation for a pattern p and classification as definitely detected (DD) or potentially detected (PD).

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_{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_{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 $v^j_{o_i}$ $(0 \leq j \leq 63)$ 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 $v^j_{o_i}$ $(0 \leq j \leq 63)$ 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_{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_{possDD} and added to O_{PD}. In the other case, the fault is marked as DD, because

$$v^G(p, o_i), v^f(p, o_i) \in \{0, 1\} \wedge v^G(p, o_i) = \neg v^f(p, o_i) \tag{5}$$

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_{possDD} is empty and O_{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_{possPD} is not empty, the SAT solver is used to determine if one of the outputs in O_{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_{possPD} is member of $\text{REX}^f(p)$, the fault is marked as PD. In the case that all outputs in O_{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 ϕ 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 ϕ 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 ϕ, 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 propgation 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.

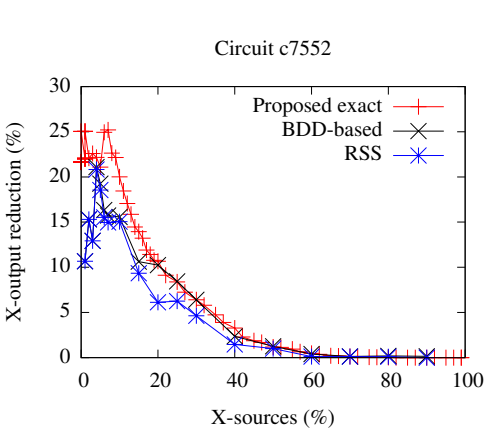
5.1. Reduction of Unknown Output Values

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

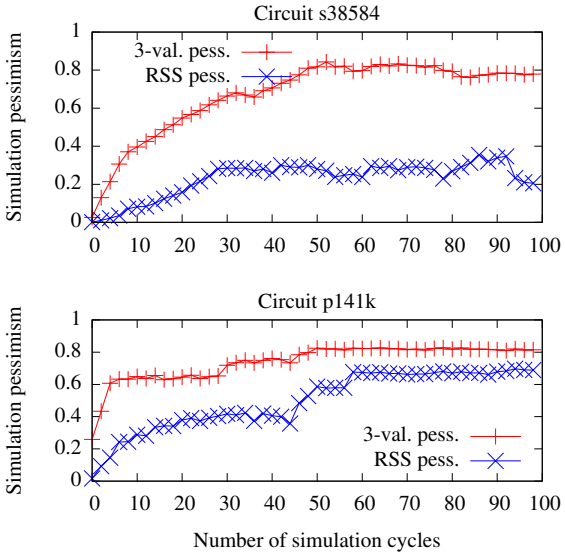
EDT environments to avoid unnecessary DFT overhead for X-masking or X-blocking structures, and overmasking of FEX-valued outputs.

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 1 000 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.



a) Reduction of unknown output values of ISCAS circuit c7552 (Red: Exact method, black: BDD-based [Kochte et al. 2011], blue: RSS).



b) Pessimism in sequential simulation for 100 cycles.

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 1 000 test patterns and 5% X-sources.

Circuit	Gates	Outputs		
		PEX	REX	Reduction (%)
c6288	2 416	23 387	15 215	34.9
c7552	3 512	11 826	9 387	20.6
s38417	22 179	143 195	122 760	14.3
s38584	19 253	95 323	91 877	3.6
p77k	68 146	248 242	195 593	21.2
p78k	74,243	685 605	454 267	33.7
p81k	88 741	491 965	384 691	21.8
p89k	83 538	309 047	255 373	17.4
p100k	90 712	325 348	277 661	14.7
p141k	163 568	1 154 394	919 633	20.3
p267k	262 451	1 284 472	1 124 916	12.4
p286k	337 048	1 172 617	758 732	35.3
p295k	274 872	2 506 184	2 102 686	16.1
p330k	330 014	1 583 657	1 322 674	16.5
p378k	371 215	3 637 222	2 352 902	35.3
p388k	456 964	2 326 166	1 371 057	41.1
Average		999 916	734 964	26.5

5.2. Exact Sequential Logic Simulation

For sequential ISCAS'89 and NXP circuits additional experiments considering multiple time frames have been conducted to assess the pessimism of 3-valued sequential simulation for a given number of time frames. Five simulation runs considering a set of five input pattern sequences and 100 time frames were performed. In each run, 1% or 2% of the flip-flops in the first time frame have been randomly selected as X-sources.

Figure 5 b) shows the pessimism of sequential simulation of 100 cycles for circuits s38584 and p141k and an X-ratio of 2%. The pessimism per cycle w.r.t. 3-valued simulation is computed as the ratio of PEX-values computed by 3-valued simulation which are identified as FEX by accurate analysis: $p = \frac{|\mathrm{PEX}_{3V}|-|\mathrm{REX}|}{|\mathrm{PEX}_{3V}|}$. If $p = 1$, then all PEXs computed by 3-valued simulation are actually classified as FEX by the accurate simulation and have a binary value. As shown in the figure, simulation pessimism increases during the first few cycles and saturates at a very high level of 0.8. In average over the cycles and circuits, the simulation pessimism is 0.72 for an X-ratio of 1%, and 0.71 for an X-ratio of 2%.

The figure also shows the pessimism of restricted symbolic simulation ('RSS pess.'), computed accordingly. Depending on circuit structure and input patterns, the pessimism of restricted symbolic simulation may still be very high.

5.3. Exact Fault Simulation

This section presents the increase of fault coverage of a test pattern set due to the exact analysis with the proposed algorithm. Similar to the previous section, five simulation runs are performed per circuit for stuck-at as well as transition-delay faults and

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.

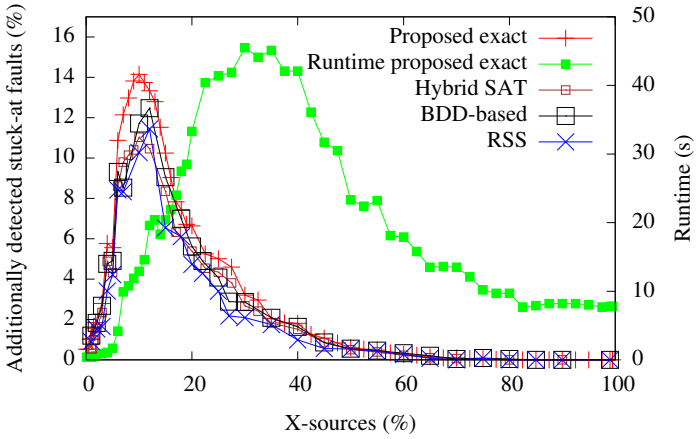


Fig. 6: Increase in stuck-at fault coverage by the proposed exact algorithm, the hybrid SAT-based algorithm [Kochte and Wunderlich 2011], the BDD-based algorithm [Kochte et al. 2011], and restricted symbolic simulation (RSS) with 1 000 random test patterns for circuit c7552.

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 'Δ 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.

Circuit	Num. of faults	3-val. Fsim. DD		Δ Exact sim. DD		Exact sim. PD		Run time
		Num.	F.C.(%)	Num.	F.C.(%)	Num.	F.C.(%)	(s)
c6288	8 704	4,453	51.16	3 809	43.77	110	1.27	3
c7552	9 756	6 222	63.77	597	6.12	1 283	13.15	1
s38417	56 325	44 535	79.07	282	0.50	2 781	4.94	46
s38584	53 845	43 459	80.71	161	0.30	1 930	3.58	15
p77k	186 645	87 337	46.72	1 151	0.62	9 204	4.93	26 695
p78k	225 476	189 586	84.08	18 602	8.25	7 295	3.24	1 761
p81k	272 322	141 953	52.13	4 205	1.54	22 579	8.29	29 055
p89k	228 888	124 090	54.21	3 571	1.56	19 137	8.36	1 881
p100k	247 376	173 073	69.96	4 297	1.74	24 483	9.90	4 292
p141k	434 363	334 535	77.02	13 501	3.11	16 544	3.81	9 889
p267k	640 221	449 787	70.26	8 404	1.31	50 527	7.89	13 719
p295k	728 246	410 393	56.35	61 897	8.50	35 339	4.85	36 607
p330k	840 690	540 257	64.26	32 143	3.82	70 710	8.41	181 308
p378k	1 127 364	933 325	82.79	101 519	9.00	39 266	3.48	170 132
p388k	1 287 689	1 016 295	78.92	61 841	4.80	66 144	5.14	201 917
Total	6 347 910	4 449 172	70.88	315 981	4.98	367 332	5.79	686 326

according to equation (2). The last column lists the runtime for the exact analysis in seconds.

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 1 000 patterns and 5% X-sources.

Circuit	Num. of faults	3-val. Fsim. DD		Δ Exact sim. DD		Exact sim. PD		Run time
		Num.	F.C.(%)	Num.	F.C.(%)	Num.	F.C.(%)	(s)
s38417	68 744	44 168	64.25	1414	2.06	2 002	2.91	11
s38584	66 504	46 382	69.74	187	0.28	1 254	1.89	10
p77k	237 550	64 734	27.25	1 458	0.61	1 589	0.67	366
p78k	308 208	241 034	78.21	39 025	12.66	8 026	2.60	116
p81k	374 456	120 305	32.13	18 075	4.83	19 145	5.11	242
p89k	299 028	82 583	27.62	4 520	1.51	6 660	2.23	587
p100k	324 586	125 323	38.61	5 175	1.59	14 677	4.52	563
p141k	550 868	267 492	48.56	31 054	5.64	18 473	3.35	1 451
p267k	788 014	298 429	37.87	14 592	1.85	39 600	5.03	4 877
p295k	943 626	244 877	25.95	31 557	3.34	22 222	2.35	7 090
p330k	1 054 568	440 993	41.82	50 823	4.82	55 503	5.26	8 523
p378k	1 541 024	1 188 930	77.15	197 206	12.80	44 544	2.89	4 241
p388k	1 615 722	914 186	56.58	134 228	8.31	74 013	4.58	17 980
Total	8 172 898	4 079 437	49.91	529 315	6.48	307 708	3.76	46 063

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

5.3.1. Runtime Reduction by Applying a Timeout. As stated before, the runtime can by 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 1 000 patterns for different scan-chain configurations as X-sources.

Circuit	Num. of faults	X-Sources Num. SC	X-Sources FF in %	3-val. Fsim. DD (%)	Δ Exact sim. DD (%)	Exact sim. PD (%)
p77k	186 645	1	8.19	43.05	1.02	4.01
		2	15.16	37.13	1.43	6.65
		3	23.65	30.17	1.41	7.32
p78k	225 476	1	2.03	93.06	0.44	0.62
		2	3.75	89.17	0.67	1.01
		3	5.68	83.24	1.03	1.56
p100k	247 375	1	7.66	63.71	2.41	9.64
		2	16.28	45.42	4.97	16.80
		3	19.21	35.34	5.90	16.87

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

K.J. Antreich and M.H. Schulz. 1987. Accelerated Fault Simulation and Fault Grading in Combinational Circuits. *IEEE Trans. CAD of Integrated Circuits and Systems* 6, 5 (september 1987), 704 – 712. DOI:http://dx.doi.org/10.1109/TCAD.1987.1270316

B. Becker, M. Keim, and R. Krieger. 1999. Hybrid Fault Simulation for Synchronous Sequential Circuits. *Journal of Electronic Testing: Theory and Applications (JETTA)* 15, 3 (1999), 219–238. DOI:http://dx.doi.org/10.1023/A:1008376522451

R.E. Bryant. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. on Computers* 35, 8 (1986), 677–691. DOI:http://dx.doi.org/10.1109/TC.1986.1676819

J.L. Carter, B.K. Rosen, G.L. Smith, and V. Pitchumani. 1989. Restricted symbolic evaluation is fast and useful. In *Proc. IEEE International Conference on Computer-Aided Design (ICCAD'89)*. IEEE, Piscataway, NJ, USA, 38 –41. DOI:http://dx.doi.org/10.1109/ICCAD.1989.76900

H.P. Chang and J.A. Abraham. 1987. The Complexity of Accurate Logic Simulation. In *Proc. IEEE International Conference on Computer Aided Design (ICCAD'87)*. IEEE, Piscataway, NJ, USA, 404–407.

H. Chou, K. Chang, and S. Kuo. 2010. Accurately Handle Don't-Care Conditions in High-Level Designs and Application for Reducing Initialized Registers. *IEEE Trans. CAD of Integrated Circuits and Systems* 29, 4 (2010), 646–651. DOI:http://dx.doi.org/10.1109/TCAD.2010.2042905

S. Hillebrecht, M.A. Kochte, H.-J. Wunderlich, and B. Becker. 2012. Exact Stuck-at Fault Classification in Presence of Unknowns. In *Proc. 17th IEEE European Test Symposium (ETS'12)*. IEEE, Piscataway, NJ, USA, 98–103. DOI:http://dx.doi.org/10.1109/ETS.2012.6233017

A. Jain, V. Boppana, R. Mukherjee, J. Jain, M. Fujita, and M. Hsiao. 2000. Testing, Verification, and Diagnosis in the Presence of Unknowns. In *Proc. IEEE VLSI Test Symposium (VTS'00)*. IEEE, Piscataway, NJ, USA, 263–268. DOI:http://dx.doi.org/10.1109/VTEST.2000.843854

S. Kajihara, K. Saluja, and S.M. Reddy. 2004. Enhanced 3-valued logic/fault simulation for full scan circuits using implicit logic values. In *Proc. IEEE European Test Symposium (ETS'04)*. IEEE, Piscataway, NJ, USA, 108–113. DOI:http://dx.doi.org/10.1109/ETSYM.2004.1347620

S. Kang and S.A. Szygenda. 2003. Accurate Logic Simulation by Overcoming the Unknown Value Propagation Problem. *Simulation* 79, 2 (2003), 59–68. DOI:http://dx.doi.org/10.1177/0037549703254811

M. Keim, B. Becker, and B. Stenner. 1996. On the (non-)resetability of synchronous sequential circuits. In *Proc. IEEE VLSI Test Symposium (VTS'96)*. IEEE, Piscataway, NJ, USA, 240–245. DOI:http://dx.doi.org/10.1109/VTEST.1996.510863

M.A. Kochte, S. Kundu, K. Miyase, X. Wen, and H.-J. Wunderlich. 2011. Efficient BDD-based Fault Simulation in Presence of Unknown Values. In *Proc. IEEE 20th Asian Test Symposium (ATS'11)*. IEEE Computer Society, Los Alamitos, CA, USA, 383–388. DOI:http://dx.doi.org/10.1109/ATS.2011.52

M.A. Kochte and H.-J. Wunderlich. 2011. SAT-Based Fault Coverage Evaluation in the Presence of Unknown Values. In *Proc. Design, Automation and Test in Europe (DATE'11)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–6. DOI:http://dx.doi.org/10.1109/DATE.2011.5763209

S. Kundu, I. Nair, L. Huisman, and V. Iyengar. 1991. Symbolic implication in test generation. In *Proc. Conference on European Design Automation (DATE'91)*. IEEE Computer Society Press, Los Alamitos, CA, USA, 492–496. DOI:http://dx.doi.org/10.1109/EDAC.1991.206456

H.K. Lee and D.S. Ha. 1991. An Efficient, Forward Fault Simulation Algorithm Based on the Parallel Pattern Single Fault Propagation. In *Proc. International Test Conference (ITC'91)*. IEEE Computer Society, Washington, DC, USA, 946–955. DOI:http://dx.doi.org/10.1109/TEST.1991.519760

S. Mitra and K.S. Kim. 2004. X-Compact: an Efficient Response Compaction Technique. *IEEE Trans. CAD of Integrated Circuits and Systems* 23, 3 (2004), 421–432. DOI:http://dx.doi.org/10.1109/TCAD.2004.823341

M. Naruse, I. Pomeranz, S.M. Reddy, and S. Kundu. 2003. On-Chip Compression of Output Responses with Unknown Values Using LFSR Reseeding. In *Proc. of the IEEE International Test Conference (ITC'03)*. IEEE Computer Society, Los Alamitos, CA, USA, 1060–1068. DOI:http://dx.doi.org/10.1109/TEST.2003.1271094

A. Ramdas and O. Sinanoglu. 2012. Toggle-masking scheme for X-filtering. In *Proc. IEEE European Test Symposium (ETS'12)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–6. DOI:http://dx.doi.org/10.1109/ETS.2012.6233024

E.M. Rudnick, J.H. Patel, and I. Pomeranz. 1996. On potential fault detection in sequential circuits. In *Proc. IEEE International Test Conference (ITC'96)*. IEEE Computer Society, Los Alamitos, CA, USA, 142–149. DOI:http://dx.doi.org/10.1109/TEST.1996.556956

J. Savir and S. Patil. 1993. Scan-based transition test. *IEEE Trans. CAD of Integrated Circuits and Systems* 12, 8 (1993), 1232–1241. DOI:http://dx.doi.org/10.1109/43.238615

J. Savir and S. Patil. 1994. On broad-side delay test. In *Proc. IEEE VLSI Test Symposium (VTS'94)*. 284–290. DOI:http://dx.doi.org/10.1109/92.311647

C. Scholl and B. Becker. 2001. Checking equivalence for partial implementations. In *Proc. Design Automation Conference (DAC'01)*. ACM, New York, NY, USA, 238–243. DOI:http://dx.doi.org/10.1109/DAC.2001.156142

T. Schubert, M. Lewis, and B. Becker. 2010. Antom—Solver Description. *SAT Race* (2010).

Y. Tang, H.-J. Wunderlich, P. Engelke, I. Polian, B. Becker, J. Schlöffel, F. Hapke, and M. Wittke. 2006. X-masking during logic BIST and its impact on defect coverage. *IEEE Trans. on VLSI Systems* 14, 2 (2006), 193–202. DOI:http://dx.doi.org/10.1109/TVLSI.2005.863742

G.S. Tseitin. 1968. On the complexity of derivation in propositional calculus. *Studies in constructive mathematics and mathematical logic* 2, 115-125 (1968), 10–13. DOI:http://dx.doi.org/10.1007/978-3-642-81955-1_28

M. Turpin. 2003. The Dangers of Living with an X (bugs hidden in your Verilog). In *Boston Synopsys Users Group Meeting*. 1–34.

E.G. Ulrich and T. Baker. 1988. The concurrent simulation of nearly identical digital networks. In *Papers on Twenty-five years of electronic design automation (25 years of DAC)*. ACM, New York, NY, USA, 318–323. DOI:http://dx.doi.org/10.1145/62882.62918

J.A. Waicukauski, E.B. Eichelberger, D.O. Forlenza, E. Lindbloom, and T. McCarthy. 1985. Fault simulation for structured VLSI. *VLSI Systems Design* 6, 12 (1985), 20–32.

C. Wilson, D. Dill, and R. Bryant. 2000. Symbolic Simulation with Approximate Values. In *Formal Methods in Computer-Aided Design*, Warren Hunt and Steven Johnson (Eds.). LNCS, Vol. 1954. Springer, Berlin / Heidelberg, Germany, 507–522. DOI:http://dx.doi.org/10.1007/3-540-40922-X_29

P. Wohl, J.A. Waicukauski, and F. Neuveux. 2008. Increasing Scan Compression by Using X-chains. In *Proc. IEEE International Test Conference (ITC'08)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–10. DOI:http://dx.doi.org/10.1109/TEST.2008.4700646