# Accurate X-Propagation for Test Applications by SAT-Based Reasoning

Kochte, Michael A.; Elm, Melanie; Wunderlich, Hans-Joachim

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)
Vol. 31(12) December 2012

doi: http://dx.doi.org/10.1109/TCAD.2012.2210422

**Abstract:** Unknown or X-values during test application may originate from uncontrolled sequential cells or macros, from clock or A/D boundaries or from tri-state logic. The exact identification of X-value propagation paths in logic circuits is crucial in logic simulation and fault simulation. In the first case, it enables the proper assessment of expected responses and the effective and efficient handling of X-values during test response compaction. In the second case, it is important for a proper assessment of fault coverage of a given test set and consequently influences the efficiency of test pattern generation. The commonly employed n-valued logic simulation evaluates the propagation of X-values only pessimistically, i.e. the X-propagation paths found by n- valued logic simulation are a superset of the actual propagation paths. This paper presents an efficient method to overcome this pessimism and to determine accurately the set of signals which carry an X-value for an input pattern. As examples, it investigates the influence of this pessimism on the two applications X-masking and stuck-at fault coverage assessment. The experimental results on benchmark and industrial circuits assess the pessimism of classic algorithms and show that these algorithms significantly overestimate the signals with X-values. The experiments show that overmasking of test data during test compression can be reduced by an accurate analysis. In stuck-at fault simulation, the coverage of the test set is increased by the proposed algorithm without incurring any overhead.

Preprint

**General Copyright Notice**

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

This is the author's "personal copy" of the final, accepted version of the paper published by IEEE.[1]

[1] **IEEE COPYRIGHT NOTICE**
©2012 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

# Accurate X-Propagation for Test Applications by SAT-Based Reasoning

Michael A. Kochte, Melanie Elm, Hans-Joachim Wunderlich

***Abstract*—Unknown or X-values during test application may originate from uncontrolled sequential cells or macros, from clock or A/D boundaries or from tri-state logic. The exact identification of X-value propagation paths in logic circuits is crucial in logic simulation and fault simulation. In the first case, it enables the proper assessment of expected responses and the effective and efficient handling of X-values during test response compaction. In the second case, it is important for a proper assessment of fault coverage of a given test set and consequently influences the efficiency of test pattern generation. The commonly employed $n$-valued logic simulation evaluates the propagation of X-values only pessimistically, i.e. the X-propagation paths found by $n$-valued logic simulation are a superset of the actual propagation paths.**

**This paper presents an efficient method to overcome this pessimism and to determine accurately the set of signals which carry an X-value for an input pattern. As examples, it investigates the influence of this pessimism on the two applications X-masking and stuck-at fault coverage assessment.**

**The experimental results on benchmark and industrial circuits assess the pessimism of classic algorithms and show that these algorithms significantly overestimate the signals with X-values. The experiments show that overmasking of test data during test compression can be reduced by an accurate analysis. In stuck-at fault simulation, the coverage of the test set is increased by the proposed algorithm without incurring any overhead.**

***Index Terms*—Unknown values, stuck-at fault coverage, accurate fault simulation, simulation pessimism**

## I. Introduction

During test application, unknown values (Xs) can emerge from either external or internal sources of a circuit. Their sources may be uninitialized or uncontrollable sequential elements, clock domain crossings or tri-state circuitry. Signals with an X-value are useless for test purposes. They do not provide information on faults and may even cancel out informative values if fed into compaction logic. In consequence, they have a negative influence on test set size and test response size.

For an accurate computation of the set of X-valued signals and primary/pseudo primary outputs (PO/PPO), $n$-valued logic simulation is insufficient due to its pessimistic evaluation of reconvergences. An example is given in Fig. 1 where a 3-valued simulation computes the state of the output signal as X, while the signal actually takes the value 1 as two correlated Xs are fed into the XOR gate and cause X-canceling.

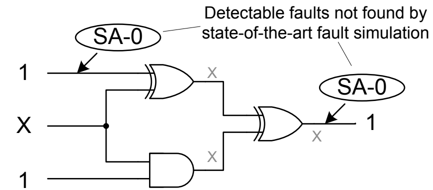


Fig. 1. Pessimism in 3-valued logic simulation and fault simulation

Clearly, the pessimism of $n$-valued logic simulation increases the amount of alleged useless signals and thereby unnecessarily increases test data volume as detailed in the following paragraphs.

### A. X-Cancelation and Pessimistic Logic Simulation

If an X occurs at a PO or PPO and is fed into compaction hardware, it may cancel out defined (non-X) values and thereby corrupt the signature produced by the compactor. To prevent this, different counter measures (so called X-handling measures) can be taken. State of the art X-handling measures either control Xs to known (non-X) values by extra logic and extra control signals, or they tolerate a certain amount of Xs while sacrificing the compaction ratio of the employed compactor. In both cases, Xs result in an increase in test data, either in control data, i.e. test patterns, or in test response size. Typically, the overhead introduced by the X-handling measures increases with the number of Xs to be handled. As a consequence, an exact analysis of X-propagation to POs and PPOs results in smaller test response sets and smaller test sets.

### B. Stuck-at Fault Simulation

Fault simulation algorithms such as the PPSFP (Parallel Pattern Single Fault Propagation) and the concurrent algorithm [1–4] use an $n$-valued logic with limited number of symbols to compute the signal values in the fault-free and faulty circuit. In 5-valued test generation or fault simulation for instance, the five signal states $\{0, 1, D, \bar{D}, X\}$ are distinguished, where $D$ and $\bar{D}$ correspond to faulty values, i.e. $D$ corresponds to a fault-free value of 1 and a faulty value of 0 and $\bar{D}$ vice versa. These algorithms are again not capable of correctly

Manuscript received March 5, 2012; revised May 21, 2012; accepted July 1, 2012. Date of current version July 10, 2012. This work was supported in part by the DFG under contract WU 245/5-2 and contract WU 245/11-1.
M. A. Kochte, M. Elm, and H.-J. Wunderlich are with the Institute of Computer Architecture and Computer Engineering, University of Stuttgart, 70569 Stuttgart, Germany (e-mail: kochte@iti.uni-stuttgart.de; melanie.elm@ieee.org; wu@informatik.uni-stuttgart.de).
Copyright © 2012 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org.

evaluating reconvergences of Xs. In the example of Fig. 1, the two stuck-at-0 faults are actually propagated and observable at the circuit output. Yet, 5-valued simulation evaluates the output signal to X. $n$-valued fault simulation results in a pessimistic estimation of fault coverage. However, correctly evaluating fault coverage is important to guarantee high product quality. A less pessimistic analysis may also reduce costs in terms of test data volume and test time, the number of patterns or number of specified bits.

To summarize, exact evaluation of X-propagation in logic and fault simulation will benefit the test quality in terms of fault coverage (i.e. reduced test escapes) and test overhead. In this paper, we propose an efficient SAT-based method for the accurate identification of X-propagation in logic simulation. Based on this method, we investigate the influence of the pessimism in state of the art $n$-valued simulation in logic simulation and its application to X-masking schemes. Furthermore, the algorithm is applied to stuck-at fault simulation to evaluate the impact of simulation pessimism on fault coverage.

The rest of the paper is organized as follows. Section II formalizes the problem of accurate simulation. Section III reviews the state of the art in X-propagation assessment and fault simulation in presence of X-values. Additionally, an overview is given over recent X-handling schemes. Section IV describes our approach for accurate X-propagation assessment in logic simulation, while section V extends this approach for stuck-at fault simulation with increased accuracy. Section VI explains the experimental setup and presents the case study on the impact of the simulation pessimism in general, and its impact on different X-handling schemes. The section also evaluates the pessimism of 3-valued stuck-at fault simulation. Finally, section VII concludes this work.

## II. Problem Statement

This work targets the accurate logic simulation of fullscan or combinational circuits in presence of unknown values for a given set of input stimuli, without the pessimism introduced by $n$-valued simulation algorithms due to propagation of unknown values. Unknown values may result from X-sources internal to the circuit or from only partially specified input stimuli.

Classical 3-valued logic simulation distinguishes between the binary logic values 0 and 1, and a state with unknown value, usually denoted U or X. For the following discussion, we distinguish three types of X-values, namely

- Real X-values (REX): A signal state which can be proven to depend on the assignments to X-sources
- Pessimistic X-values (PEX): A superset of the REX values as computed for instance by 3-valued logic simulation
- False X-values (FEX): PEX values which do not depend on the assignments to X-sources. FEX values have a logic value $\in \{0, 1\}$.

It follows that PEX = FEX $\cup$ REX and FEX $\cap$ REX = $\emptyset$ as illustrated in Fig. 2.

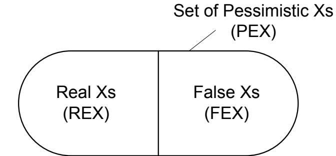


Fig. 2. PEX, REX and FEX values

To determine whether the value of a signal is a REX for a given input stimulus, the Boolean difference w.r.t. the X-sources must be computed. This, for example, can be done by computing and comparing all possible values of the signal by enumerating all assignments to the X-sources. If the signal value is not constant for all assignments, it is a REX for the given input stimulus.

In X-masking algorithms, the occurrence of X-values at the pseudo primary outputs or inputs of compaction hardware impacts the overhead in terms of hardware, test data, or time. In addition, overmasking may result in a loss of failure data and fault coverage. An accurate logic simulation algorithm can be employed to accurately classify which pseudo primary outputs capture REX values in which pattern.

To accurately compute whether a fault in the circuit with X-sources is detected by a given test pattern, the test responses of the fault free and faulty circuit must be computed and compared. The exact response of the fault free circuit can be obtained by an accurate logic simulation as outlined above. In the same way, the response of the faulty machine can be computed. If there is at least one output pair with constant, but opposite values in the fault free and faulty circuit, the corresponding fault is detected by the pattern. This method is able to compute the exact fault coverage of a test pattern set if the number of X-sources is very small.

## III. Related Work

Both approximate and exact methods have been proposed to overcome the pessimism in logic and stuck-at fault simulation, trading off accuracy and runtime.

### A. Logic Simulation with Increased Accuracy

Accurate logic simulation in presence of X-values can be mapped to symbolic simulation or expressed as a satisfiability instance. In a symbolic or indexed simulation of the circuit [5], the Boolean function of each signal is expressed in dependence of the free variables. This can be implemented by use of reduced ordered binary decision diagrams (ROBDDs) [6]. ROBDDs provide a canonical representation of Boolean functions and allow to compute whether a signal depends on an X-source or not. In the latter case, the signal actually has a fixed binary value $\in \{0, 1\}$ and is represented by a terminal node in the ROBDD. However, the memory requirements for BDDs may prohibit their use for some circuit structures, such as multipliers.

A different approach for exact logic simulation is based on computing the forward implication at each gate by analyzing the intersection of the input cube with the implicants of the

function and its inverse [7]. While the authors report results for circuits with up to 10 inputs, it is unclear whether this cube-based algorithm scales to circuits of thousands of inputs.

The recent work in [8] investigates the propagation of X-values resulting from uninitialized registers in high-level RTL models by mapping the design to a quantified Boolean formula (QBF) and solving it with a QBF solver. The proposed algorithm can accurately identify whether uninitialized register values are propagated over a limited number of clock cycles and whether they are observable at sequential elements.

Reasoning about the circuit behavior in presence of X-values is also known in the domain of formal verification of designs with black boxes where parts of the implementation are unknown. In the SAT-based work of [9], the state of signals driven by black boxes is modeled using two binary variables such that an X-state can be represented. However, this technique is unable to express correlations of X-states, similar to classical 3-valued logic simulation. In [10], an accurate BDD-based symbolic simulation to handle outputs of black boxes is proposed, and it is shown for smaller circuits that this technique increases the accuracy of the verification algorithm.

The mixed approach of [11] combines BDDs and reasoning techniques similar to SAT solvers for system-level verification. The algorithm distinguishes internal variables for which the exact state needs to be kept and variables for which a pessimistic value is sufficient in the context of the design verification. If the available memory is exceeded, the method conducts an explicit case splitting analysis, which increases runtime in exchange for lower memory consumption.

Since the computational effort for formal methods may render the application to larger circuits too expensive or even impossible, pessimistic algorithms have been proposed which still offer increased accuracy compared to classical $n$-valued logic simulation. In restricted symbolic simulation [12] with a limited domain of symbols, also termed distinguishing X-simulation [13], or applied to test generation as in [14], the simulation result remains pessimistic. However, runtime and space requirements are much lower than for unbound symbolic simulation.

Indirect implications which are derived during static learning can also be used to increase the accuracy of logic simulation [15, 16]. The idea stems from the ATPG domain and is based on evaluating the contrapositive of signals in the circuit. A learning criterion selects a subset of indirect implications which are not trivially found by following all transitive direct implications. While methods based on static learning require only moderate computational effort, the number of the resulting indirect implications may be very high and increase the size of the circuit representation significantly.

The approximation algorithm of [17] is based on partitioning the circuit into reconvergent regions of limited size starting at X-valued fanout stems. Partitions are simulated separately to learn about X-propagation. This information is then fed back into the simulation of the whole circuit. Due to the size limitation on partitions, the result is still pessimistic.

### B. X-Handling and Test Response Compaction in Presence of X-Values

Different kinds of compaction schemes show different vulnerability to Xs and, in consequence, different techniques have been developed to prevent Xs from corrupting the compactor signature. They can be classified as X-tolerant space compaction, X-tolerant time compaction and X-masking for both space and time compaction. All of them benefit if the number of Xs to be masked can be reduced.

X-tolerant space compactors tolerate Xs by construction of the compactor [18–23]. Two types of time compactors and according X-tolerance schemes can be distinguished: finite impulse response (FIR) compactors as convolutional compactors [24] and infinite impulse response (IIR) compactors as MISR compactors (Multiple Input Shift Registers). Xs fed into FIR compactors corrupt only a subset of bits of the signature and can thereby inherently be tolerated and extracted on the tester [25]. Xs fed into IIR compactors may affect all signature bits generated in all future compaction cycles. They can be tolerated by X-canceling schemes [26].

X-tolerant compactors are only suitable if the amount of Xs is bound to a certain number $p$. If the amount of Xs fed into the compactor exceeds $p$, Xs will corrupt the signature. The limit $p$ can be traded off against the compaction ratio. In order to optimize the compaction ratio, it is crucial to identify the signals with REX values instead of those with PEX values.

X-masking logic is synthesized in between the circuit outputs, i.e. scan outs, and the inputs of any arbitrary compaction logic [27–32]. During scan-out a predetermined X can be converted into a specified value by feeding it e.g. through a NAND gate and controlling the value of the output by an extra *mask input*. Most masking approaches mask a superset of those scan cells exposed to X-propagation, where a trade-off between control data for masks and overmasked PPOs has to be found. With fewer Xs to be masked, better solutions for the trade-off between overhead and overmasking can be found, and less area is required for the implementation of the masking function [27, 32].

For all the aforementioned X-handling schemes it is beneficial to identify the circuit outputs carrying the real Xs instead of the pessimistic super set only. Thereby, the efficiency of the schemes can be improved in terms of applicability, compaction ratio and information content of the signatures.

### C. Stuck-at Fault Simulation with Increased Accuracy

The use of indirect implications in stuck-at fault simulation has been proposed by [33]. Due to the limitations of static learning methods, the achievable accuracy is limited, i.e. only a small subset of actually detectable faults is identified in addition to classical fault simulation algorithms.

As in logic simulation, ROBDDs can also be used for a symbolic simulation of the fault-free and faulty circuit and fault classification. The application for symbolic fault simulation of MOS circuits is described in [34].

Especially for sequential circuits, where Xs originating from uninitialized sequential elements may spread over the

whole circuit and significantly impair fault coverage, symbolic simulation has been applied. Restricted symbolic simulation is used in [35] for synchronous sequential circuits.

The use of BDDs for the computation of synchronizing (reset) sequences which set an uninitialized sequential circuit to a defined state was proposed in [36]. The technique was applied to test generation in synchronous sequential circuits in [37]. For larger circuits, the symbolic traversal and search of states by BDDs may exceed the available memory. To trade off memory requirements and accuracy of state traversal, the hybrid algorithm for fault simulation of [38] conducts exact BDD-based symbolic simulation until the BDD memory requirement exceeds a given limit. In that case, the hybrid approach switches to conventional 3-valued simulation and introduces pessimism. Switching between the pessimistic and accurate simulation is performed between simulation cycles.

## IV. Accurate Logic Simulation

In contrast to pessimistic $n$-valued simulation, which only computes the set of PEX values, the proposed accurate logic simulation algorithm is able to efficiently distinguish all REX and FEX values. This is performed by firstly computing a pessimistic set of X-valued signals (PEXs) which is then analyzed by a SAT solver. Only reconvergences of REX signals have to be handled by the SAT solver since only these can generate FEX values. The results are the sets of REX and FEX valued signals in the fault-free circuit.

This step is correct and complete, i.e. firstly it correctly identifies *all* REX valued signals in the set of PEX valued signals, and secondly, it determines the actual binary value for all FEX valued signals. This value is independent of assignments to the X-valued inputs or other X-sources.

### A. Overview of the Accurate Logic Simulation Algorithm

The simulation algorithm is a hybrid approach that combines restricted symbolic simulation, reconvergence analysis and exact SAT-based reasoning. It extends the method of [39]. FEX signals can only emerge if a REX is propagated along multiple paths which *reconverge* at a gate and X-canceling occurs as illustrated in the example in Fig. 1. The convergence of unrelated X-values cannot produce a FEX state.

Starting with a fast 3-valued logic simulation of the netlist with signals $S$, inputs $I$ and outputs $O$ under a test pattern $\mathcal{P} : I \mapsto \{0, 1, X\}$ and $I_x \subseteq (I \cup S)$ the set of X-sources, the values of the internal signals $S$ and outputs $O$ are determined. Let $S_x$ and $O_x$ denote the subsets of signals respectively outputs with a $PEX_{3V}$ value determined by 3-valued simulation. To reduce the number of X-reconvergences to be processed by the SAT solver, an event-based restricted symbolic simulation is conducted on the signals $S_x$. The result is the set $PEX_{RSS} \subseteq PEX_{3V}$.

Following a reconvergence analysis of the $PEX_{RSS}$ signals, a SAT instance is generated which is used for the exact computation of the signal states at REX reconvergences under the given input pattern $\mathcal{P}$. During topological traversal of the $PEX_{RSS}$-valued signals starting from the X-sources $I_x$, the SAT-based evaluation at REX reconvergences is invoked. At all other gates where the value can be correctly and quickly computed without SAT-based analysis, the netlist traversal performs direct forward implications of the signal states. The flow of the algorithm is depicted in Fig. 3.

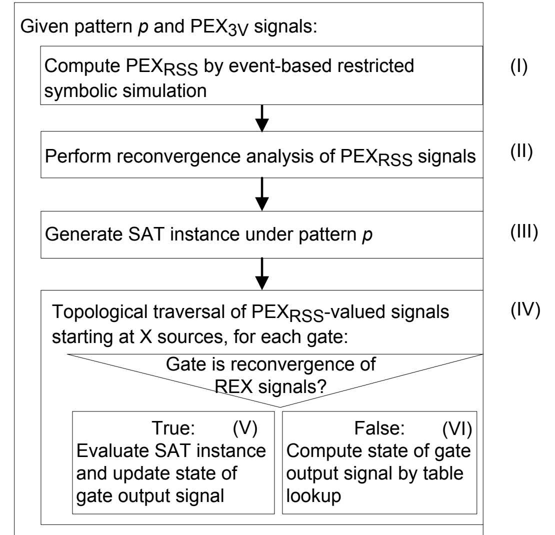


Fig. 3. Accurate logic simulation flow

### B. Restricted Symbolic Simulation

Restricted symbolic simulation (RSS) simulates a circuit under a given pattern with a higher, but still limited number of symbols [12]. Apart from symbols for the binary values $\{0, 1\}$, additional symbols represent different X-states in the circuit. Symbols are usually encoded as integers in the machine word. Each X-source is assigned a unique symbol. Inversions of X-states are computed by negation of the numerical value of the encoded symbol.

Reconvergences of X-valued signals can result in X-canceling as e.g. the conjunction of an X-state and its inversion yielding 0 irrespective of the actual value of the unknown. This evaluation is easily performed based on the integer encoding. If two or more unrelated X-states are combined at a gate, the resulting symbolic output value cannot be represented in the encoding and a yet unused X-symbol is introduced. At this point, restricted symbolic simulation introduces pessimism into the computation by losing the identity of the combined X-states.

Here, restricted symbolic simulation is implemented as an event-based simulation which only evaluates the set of $PEX_{3V}$ signals. The separation of 3-valued simulation (e.g. implemented as pattern parallel simulation) and event-based restricted symbolic simulation reduces the runtime especially for low numbers of X-sources in the circuit.

### C. Reconvergence Analysis

From the previous step (I), the set of $PEX_{RSS}$-valued signals is known. At $PEX_{RSS}$-valued fanout stems, a reconvergence analysis is performed. This analysis is implemented

as an event-based forward traversal of the $PEX_{RSS}$ signals starting from the fanout stem. It derives the set of gate output signals $S_x^R$ where disjoint paths reconverge. Found reconvergences are stored for subsequent generation of the SAT instance.

### D. Generation of the SAT Instance for Signal Classification

A SAT solver is used to exactly classify the $PEX_{RSS}$-valued signals in the circuit in presence of X-sources. For a considered REX reconvergence, a SAT instance is constructed that is satisfiable if and only if the reconverged signal is a REX. This is achieved by searching for two assignments $\mathcal{P}_1$ and $\mathcal{P}_2$ from $I_x \mapsto \{0,1\}$ of the X-sources which result in complementary values at the reconverged signal. If such assignments do not exist, then the reconverged signal has a constant value $\in \{0,1\}$ irrespective of the value assignments to $I_x$.

A single SAT instance is generated per pattern and evaluated under different constraints such that all REX reconvergences for that pattern can be classified iteratively. The SAT instance does not comprise all gates and signals of the circuit, but is restricted to those which generate $PEX_{RSS}$-values at their output. The SAT instance models two copies of these gates and signals, termed $s$ and $s'$. The set of clauses of the gates in the input cone of $s$ ($s'$) is denoted $C_s$ ($C'_s$). Additional clauses are introduced at reconvergences for comparison as illustrated in Fig. 4. For each reconvergence $s \in S_x^R$, the two clauses $\{s, s'\}, \{\neg s, \neg s'\}$ are satisfied only if the values of signal $s$ in the circuit and $s'$ in the copy have different values. To evaluate each reconvergence $s$ separately, the clauses for comparison can be directly satisfied via an additional selector variable $s_{SEL}$ per reconvergence. The resulting clauses

$$D_s := \{\{s, s', \neg s_{SEL}\}, \{\neg s, \neg s', \neg s_{SEL}\}\}$$

are added to the SAT instance for each reconvergence, resulting in the SAT instance $\mathcal{P}_{SAT}$ for pattern $\mathcal{P}$:

$$\mathcal{P}_{SAT} := \bigcup_{s \in S_x^R} (C_s \cup C_{s'} \cup D_s).$$

To evaluate a particular reconvergence $s$, the corresponding selector variable $s_{SEL}$ is constrained to *true* and all other selector variables are constrained to *false*. The SAT solver then searches for two assignments to the X-sources which cause complementary values at $s$ and $s'$. If such assignments exist, then the signal value is a REX which depends on the value of the X-sources. If the SAT solver proves that no such assignments exist, then $s$ has a FEX value independent of the assignment to any of the X-sources.

### E. Exact Logic Simulation

The exact logic simulation of the circuit with a given pattern is performed by a topological traversal of the $PEX_{RSS}$ valued signals in the circuit, starting from the inputs (step IV in Fig. 3). The PEX signals are classified either by invoking the SAT solver (step V) or by a table look-up based logic simulation. The SAT solver is invoked at a reconvergence if and only if multiple inputs of the corresponding gate have been classified as REX values. The result of the SAT-based computation (step V) is then used in the evaluation of the subsequent, yet unclassified PEX signals. If only one of the inputs of a gate is a REX and the others have a defined value including FEX, the correct output value is derived by a table look-up without the SAT solver.

The result of the accurate logic simulation is the correct classification of all PEX signals in the circuit into FEX and REX for the considered pattern $\mathcal{P}$. This classification includes all primary and pseudo primary outputs of the circuit. Consequently, this information can be used for X-masking schemes as outlined in section VI.

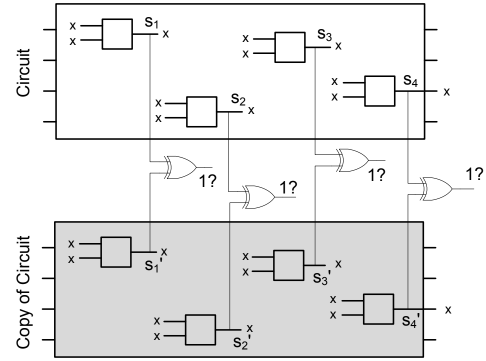


Fig. 4. Principle of signal evaluation at PEX reconvergences $s_i \in S_x^R$

## V. Improved Computation of Stuck-at Fault Coverage

The computational effort of an accurate logic simulation of every faulty machine per pattern is computationally very expensive. The proposed algorithm uses the SAT-based accurate logic simulation method of the previous section for the fault-free circuit as basis for the stuck-at fault coverage computation. This way, the serial analysis of all yet undetected faults for each pattern can be avoided. By trading-off computing time and pessimism in the fault classification step, a significant improvement in accuracy compared to conventional $n$-valued fault simulation is achieved with reasonable computational effort.

The next section outlines the algorithm, followed by a presentation of the fault classification method. The remaining pessimism of the proposed method is discussed in section V-C.

### A. Overview

Fig. 5 shows the overall flow of the fault analysis algorithm which incorporates ideas from PPSFP fault simulation [2–4]. First, each pattern is accurately simulated (step I). To analyze the faulty machines, the circuit is partitioned into fanout-free regions as shown in Fig. 6. The fanout-free regions are processed one at a time. Based on the accurate logic simulation, the exact logic values in the fault free case and the activated faults in the fanout-free regions are known.

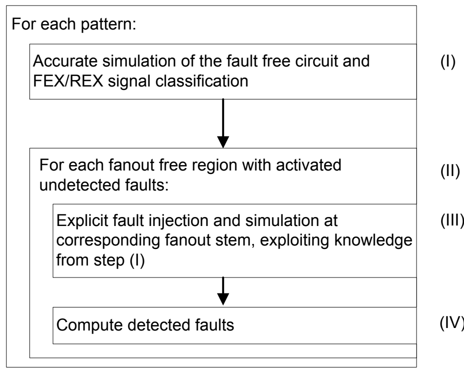


Fig. 5. Overview of stuck-at fault coverage computation in presence of Xs

To determine the observability of the activated faults, the corresponding fanout stems are evaluated by explicit fault injection and simulation (step III). For the simulation of the transitive fanout of the stems, a pessimistic but fast 3-valued logic simulation is performed, trading off computational effort and accuracy.

### B. Stuck-at Fault Classification

A fault effect is propagated through the circuit and may turn an X-value into a defined value of 0 or 1, or vice versa. If an output is altered from a defined value to an X, a *possible detect* can be reported. However, if the fault is propagated along reconverging paths with X-values, the output in the faulty case might be a FEX as well, and in principle it is possible to decide about a *definite detect* or a *definite undetect* in the same way as we decide about REX and FEX in the fault free case. Yet this computation has to be performed for each fault separately and is only possible for small circuits due to complexity limitations.

By applying algorithmic optimizations as found in state-of-the-art fault simulators, the efficiency of the fault classification is increased: Fault activation in fanout-free regions is evaluated separately from fault propagation from the corresponding fanout stem [2–4]. To avoid excessive simulation times for the stems, a fast 3-valued logic simulation is performed in step III.

For each fanout-free region, the activation of each fault in the region is determined using the signal classification from the accurate logic simulation. For the local propagation of the faults to the corresponding fanout stem, these exact values are used as well. Within the fanout-free region, the fault effect can only propagate along a single path towards the next fanout stem. Off-path signals are not affected by the fault. However, a faulty signal may influence a reconvergence of X-values in the output cone of the fault. While fault activation can be correctly derived based on the values of the accurate logic simulation, the local propagation to the corresponding fanout stem can only be pessimistically computed, unless another accurate analysis is conducted.

If faults are activated and propagated to a stem, an explicit fault injection and simulation originating at this stem is conducted to determine stem observability at the circuit outputs or at an intermediate signal dominator [4]. Here, signal reconvergences may be affected by the fault. We trade-off the computational requirements and the accuracy of the fault classification by using 3-valued logic simulation for the stem simulation. This introduces some pessimism compared to the exact solution. At the boundary of the transitive fanout of the considered stem, the results of the accurate simulation of step I are used as shown in Fig. 6.

Finally, using stem observability and fault activation information, the detected faults are enumerated (step IV).

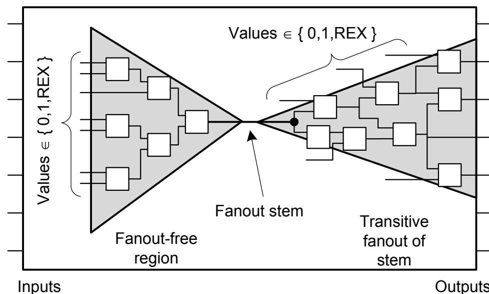


Fig. 6. Computation of fanout stem observability and fault detection in fanout free regions

### C. Pessimism in the Fault Coverage Computation

While an accurate logic simulation of the fault-free and each faulty machine allows to compute the exact fault coverage of a given test pattern, the required computational effort is prohibitively high. The algorithm discussed in the preceding section uses 3-valued logic simulation to evaluate the logic gates in the fanout of the fault to speed up the computation. Consequently, X-reconvergences reachable from the fault site are evaluated pessimistically.

This may affect local fault propagation in fanout-free regions as well as fanout stem observability at observable circuit outputs. An example is given in Fig. 7 where the stuck-at-1 fault influences the X-reconvergence at the XOR gate. In the proposed algorithm, the detection of this fault cannot be definitely classified. This small degree of pessimism

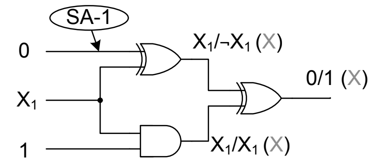


Fig. 7. Fault-free and faulty circuit state with accurate computation of fault propagation and the proposed algorithm (in brackets)

which is introduced by trading-off runtime and accuracy may cause *definite detects* of an accurate analysis to be classified as *possible detects*. In the following experimental part, the accuracy of the algorithm is evaluated.

### D. Computation of Transition Fault Coverage

The proposed method can be extended to the computation of transition fault coverage by separate analysis of the launch and capture time frame. An exact logic simulation of the launch time frame computes transition fault activation, and a stuck-at fault simulation of the capture time frame computes fault propagation and observability. This approximately doubles the computational effort compared to stuck-at fault simulation.

## VI. Evaluation and Results

The proposed algorithm has been verified and applied to numerous benchmark and industrial circuits. This section presents a comparison with related work and a discussion of the achieved accuracy. Firstly we briefly explain the verification of the correctness and computation of accuracy of the results. We investigate the amount of FEXs with respect to a 3-valued logic simulation and former approaches for enhancing the accuracy of X-propagation evaluation in section VI-B. Then we assess the influence of an exact logic simulation on X-masking schemes preceding a response compaction. Section VI-C and VI-D present the improvements in stuck-at fault coverage of the proposed fault simulation algorithm. Section VI-E compares the result to restricted symbolic simulation based algorithms. Finally, we discuss the computing time of the proposed algorithm.

To show the impact of the exact analysis in dependence on the number of X-sources, we conducted the experiments by injecting Xs into the primary and pseudo-primary inputs (PPIs) of the designs. We conducted experiments with 0.5%, 1.0% and 2.0% Xs in the PPIs, which results in X-densities in the PPOs roughly corresponding to the numbers reported in [40].

### A. Verification by Exhaustively Filled Patterns

The correctness of the algorithm is verified using a simulation based strategy. For input patterns with less than 16 X-valued inputs, the partially specified patterns are filled exhaustively, i.e. all possible assignments for these X-valued inputs are enumerated. The resulting patterns are subject to logic respectively fault simulation. If signals carry a constant value of 0 or 1 for all the patterns, they cannot be a REX, and if a stuck-at fault is detected by all the patterns, it is a definite detect.

For larger circuits and patterns with more than 16 X-valued inputs, the required computing effort prohibits exhaustive verification. Here, a validation approach is chosen. Instead of enumerating all possible patterns that originate from a partially specified one, only 128 patterns are chosen. From these 128 patterns, two are chosen deterministically (all X-valued inputs set to 0, and to 1) while the rest is randomly filled.

In addition, the fault simulation of 128 filled patterns (originating from a single partially specified pattern), followed by intersection of the sets of faults detected by each pattern, allows to derive an upper bound of the number of additionally detectable faults in the circuit due to accurate analysis. This information is used to assess the accuracy of the proposed algorithm (c.f. section VI-D). Because of the small sample size of only 128 patterns, the computed coverage is an optimistic upper bound. Thus, the derived accuracy of the proposed algorithm is a pessimistic lower bound of its actual accuracy.

### B. Accurate Logic Simulation

*1) Improving Simulation Accuracy:* Firstly, we investigate the degree of pessimism found in 3-valued logic simulation using the largest ITC'99 benchmark circuits (b17, b18, b19) and a set of industrial circuits kindly provided by NXP (named p*k) as shown in Table I. The number of REX valued outputs is determined for different configurations of X-sources at the circuit inputs. Secondly, we compare the amount of found FEX outputs with the results of the method based on indirect implications presented in [33].

| Circuit | #Inp. | #Outp. | #Gates | #Stuck-at faults | Scan conf. #chains | length |
|---|---|---|---|---|---|---|
| b17 | 1452 | 1512 | 35549 | 81330 | — | — |
| b18 | 3357 | 3364 | 124952 | 277976 | — | — |
| b19 | 6666 | 6713 | 251692 | 560696 | — | — |
| p77k | 3487 | 3400 | 65015 | 121262 | 143 | 28 |
| p78k | 3148 | 3484 | 68263 | 163258 | 195 | 22 |
| p81k | 4029 | 3952 | 106450 | 223764 | 144 | 29 |
| p89k | 4632 | 4557 | 80963 | 155794 | 306 | 57 |
| p100k | 5902 | 5829 | 84356 | 166960 | 270 | 53 |
| p141k | 11290 | 10502 | 152808 | 287552 | 264 | 45 |
| p239k | 18692 | 18495 | 224597 | 455992 | 360 | 61 |
| p259k | 18713 | 18495 | 298796 | 607536 | 360 | 61 |
| p267k | 17332 | 16621 | 238697 | 372140 | 360 | 62 |
| p269k | 17333 | 16621 | 239771 | 374296 | 360 | 62 |
| p279k | 18074 | 17827 | 257736 | 493744 | 385 | 59 |
| p295k | 18508 | 18521 | 249747 | 478996 | 330 | 62 |
| p330k | 18010 | 17468 | 312666 | 547808 | 320 | 64 |
| p378k | 15732 | 17420 | 341315 | 816274 | 325 | 64 |
| p388k | 25005 | 24065 | 433331 | 856678 | 400 | 66 |
| p418k | 30430 | 29809 | 382633 | 688808 | 576 | 93 |

TABLE I
Circuit characteristics.

For the three different numbers of X-sources, 16 different configurations are investigated per circuit. In each configuration, the X-sources are selected randomly and 32 random patterns are assigned to the circuit. The propagation of the X-values to the outputs is computed by 3-valued logic simulation. Then, the REX outputs are determined for each pattern. Table II lists for the three cases the average X-density at the outputs and the minimum, average and maximum REX ratio over the 16 configurations. The X-density denotes the average ratio of PEX valued outputs computed by 3-valued simulation and the number of circuit outputs. The REX ratio at the circuit outputs $O$ for a configuration $c$ is the ratio of the number of REX outputs and the number of PEX outputs determined in 3-valued simulation over all patterns:

$$\text{REX ratio}_c := \frac{\sum_{Pat.\ k} |REX_c^k \cap O|}{\sum_{Pat.\ k} |PEX_c^k \cap O|},$$

with $REX_c^k$ ($PEX_c^k$) being the set of REX (PEX) outputs under pattern $k$.

For the majority of the circuits, a high percentage of X-valued outputs can be proven to be false X. For circuit p259k,

| Circuit | X-input ratio 0.5% | | | | X-input ratio 1.0% | | | | X-input ratio 2.0% | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | X-den. [%] | REX ratio [%] min | avg | max | X-den. [%] | REX ratio [%] min | avg | max | X-den. [%] | REX ratio [%] min | avg | max |
| b17 | 0.98 | 42.5 | 90.1 | 100.0 | 2.24 | 47.0 | 86.1 | 100.0 | 3.55 | 56.5 | 82.5 | 100.0 |
| b18 | 0.68 | 52.3 | 91.0 | 100.0 | 1.57 | 63.2 | 87.4 | 99.2 | 3.50 | 65.3 | 82.1 | 94.8 |
| b19 | 0.81 | 63.8 | 84.8 | 99.6 | 1.65 | 69.1 | 81.8 | 93.6 | 3.35 | 64.2 | 80.6 | 92.3 |
| p77k | 1.94 | 17.2 | 83.1 | 100.0 | 0.45 | 35.0 | 84.2 | 100.0 | 2.51 | 64.8 | 86.6 | 97.6 |
| p78k | 2.25 | 50.3 | 57.2 | 66.1 | 4.97 | 50.9 | 56.9 | 65.4 | 9.31 | 53.6 | 58.9 | 66.8 |
| p81k | 1.67 | 47.5 | 52.9 | 56.6 | 3.28 | 49.0 | 56.5 | 73.7 | 6.12 | 53.7 | 59.0 | 62.3 |
| p89k | 0.50 | 69.6 | 86.7 | 98.9 | 0.81 | 55.5 | 80.1 | 97.9 | 2.50 | 56.7 | 82.5 | 96.0 |
| p100k | 0.50 | 58.8 | 86.1 | 94.3 | 0.89 | 72.0 | 86.3 | 97.0 | 3.45 | 58.1 | 82.8 | 93.6 |
| p141k | 1.60 | 25.0 | 73.9 | 87.9 | 2.23 | 64.5 | 76.6 | 86.0 | 3.81 | 67.3 | 76.6 | 83.8 |
| p239k | 1.40 | 47.2 | 63.2 | 82.2 | 3.03 | 60.2 | 66.8 | 78.3 | 4.89 | 62.1 | 69.4 | 78.6 |
| p259k | 1.21 | 44.9 | 47.1 | 51.4 | 2.40 | 46.2 | 48.8 | 55.0 | 6.34 | 49.6 | 53.7 | 96.5 |
| p267k | 0.75 | 81.9 | 90.4 | 98.9 | 1.78 | 56.3 | 86.4 | 96.7 | 3.21 | 69.2 | 84.9 | 94.6 |
| p269k | 0.66 | 50.7 | 89.4 | 99.4 | 1.70 | 55.1 | 83.7 | 95.5 | 3.01 | 76.4 | 88.2 | 93.3 |
| p279k | 0.80 | 72.6 | 81.6 | 90.6 | 1.27 | 74.1 | 85.9 | 94.2 | 2.81 | 75.9 | 83.8 | 89.6 |
| p295k | 0.26 | 83.3 | 90.3 | 98.6 | 3.54 | 84.4 | 92.5 | 98.8 | 4.23 | 48.9 | 77.2 | 99.2 |
| p330k | 0.83 | 64.6 | 79.7 | 90.3 | 1.96 | 65.2 | 78.8 | 94.6 | 3.45 | 74.1 | 80.4 | 94.5 |
| p378k | 2.52 | 51.2 | 55.7 | 63.1 | 5.42 | 52.5 | 55.3 | 57.3 | 9.28 | 56.7 | 58.3 | 59.5 |
| p388k | 2.01 | 41.4 | 56.2 | 91.7 | 3.27 | 44.1 | 50.8 | 91.0 | 4.89 | 38.7 | 51.4 | 73.8 |
| p418k | 0.79 | 75.2 | 85.3 | 95.8 | 1.42 | 70.7 | 84.5 | 91.3 | 2.47 | 78.8 | 85.5 | 90.8 |

TABLE II

RATIO OF REX OUTPUTS FOR DIFFERENT X-SOURCES CONFIGURATIONS. THE SMALLER THE PERCENTAGE, THE HIGHER IS THE PESSIMISM OF 3-VALUED LOGIC SIMULATION AND THE HIGHER IS THE GAIN OF EMPLOYING THE EXACT ANALYSIS.

in average only about 50% of the PEX valued outputs are actually REXs. Over all circuits, only about 75% of the X-valued outputs are REX outputs.

In the second experiment, we compare the number of detected FEX outputs with the indirect implication based method from [33]. We report results for the set of ISCAS'85/89 circuits evaluated in [33]. The experiment assumes that 50% of the circuit inputs are X-sources. Each circuit is evaluated with 32 random input patterns. Table III lists the average and maximum number of FEX outputs per pattern for the method of [33] and the exact method proposed here.

| Circuit | Indir. Impl. [33] avg #FEX | max #FEX | Proposed method avg #FEX | max #FEX |
|---|---|---|---|---|
| c2670 | 1.03 | 2 | 1.88 | 5 |
| c5315 | 0.22 | 2 | 3.44 | 7 |
| s5378 | 3.00 | 11 | 5.84 | 13 |
| s9234 | 2.40 | 6 | 8.90 | 19 |
| s13207 | 2.50 | 11 | 13.72 | 29 |
| s15850 | 7.00 | 22 | 18.91 | 41 |
| s35932 | 0.06 | 1 | 0.88 | 8 |
| s38417 | 1.90 | 11 | 44.66 | 68 |
| s38584 | 24.00 | 53 | 68.91 | 130 |

TABLE III

AVERAGE AND MAXIMUM NUMBER OF FEX OUTPUTS FOUND BY [33] AND THE PROPOSED METHOD.

For all circuits, the average and maximum number of identified FEX outputs in the exact method exceeds the number of the indirect implication based method. For circuit s38417, the average and maximum number of identified false X-outputs is more than 20x respectively 6x higher.

*2) Impact on Chain and Vector Masking:* The exact analysis of REX values is beneficial for all X-tolerating and X-masking architectures. In this section, we focus on the most simple ones, scan vector and scan chain masking, where either a complete vector or complete chain is masked per pattern. More complex schemes may benefit even more. The experiment is restricted to the industrial circuits, because scan configurations were not available for the ITC'99 benchmark circuits.

Table I gives an overview of the scan configurations of the circuits. The first columns give the number of primary and pseudo-primary inputs and outputs, as well as the number of gates. Column five and six show the number of scan chains and the maximum scan chain length. The scan configurations are chosen to have many but very short scan chains.

Three different X-source distributions with 1.0% Xs in the inputs are chosen from the previous experiments: The distribution with the minimum number of REX outputs, an average one and the one with the maximum number of REX outputs. For these distributions we generate X-aware test patterns with a commercial ATPG tool and apply them to analyze the percentage of overmasked scan chains or vectors.

Table IV reports the results for the three X-distributions. The table reports the percentage of scan chains (respectively vectors) with at least one REX output in the exact analysis w.r.t. the chains (vectors) with at least one PEX output according to 3-valued analysis:

$$\text{Mask ratio} = \frac{\#\text{XChains}_{\text{Exact}}}{\#\text{XChains}_{\text{3-valued}}} \cdot 100\%.$$

This mask ratio quantifies the fraction of chains (vectors) which are masked according to the exact analysis, compared to masking based on 3-valued analysis. For circuit p77k and the minimum configuration, for instance, only 70.1% of the scan chains masked based on 3-valued analysis (91.2% of the vectors), actually require masking when the exact analysis is applied. The results show that up to 30% of the masked chains (vectors) are masked unnecessarily when only a pessimistic analysis is applied.

In order to assess the negative impact of this overmasking on the observability of scan cells, another set of experiments is conducted for the five largest circuits in the set. Based on the ATPG test patterns, the number of observable scan cells is

| Circuit | Min., Mask ratio [%] Chains | Vectors | Avg., Mask ratio [%] Chains | Vectors | Max., Mask ratio [%] Chains | Vectors |
|---|---|---|---|---|---|---|
| p77k | 70.1 | 91.2 | 96.0 | 94.4 | 97.6 | 97.1 |
| p78k | 82.9 | 99.2 | 88.0 | 98.9 | 98.2 | 100.0 |
| p81k | 88.7 | 95.8 | 92.7 | 92.6 | 87.9 | 92.7 |
| p89k | 87.7 | 69.5 | 97.6 | 92.6 | 96.9 | 98.0 |
| p100k | 91.8 | 88.4 | 96.5 | 93.4 | 97.8 | 94.5 |
| p141k | 94.4 | 100.0 | 81.7 | 95.0 | 87.5 | 96.1 |
| p239k | 84.1 | 99.1 | 84.7 | 97.5 | 86.5 | 99.1 |
| p259k | 67.5 | 93.1 | 67.4 | 93.7 | 65.0 | 90.8 |
| p267k | 96.6 | 90.2 | 99.4 | 97.1 | 99.7 | 99.7 |
| p269k | 96.3 | 96.3 | 98.7 | 99.9 | 99.7 | 99.5 |
| p279k | 95.9 | 99.2 | 97.6 | 98.1 | 98.3 | 95.9 |
| p295k | 95.8 | 98.3 | 97.7 | 99.0 | 99.1 | 99.4 |
| p330k | 98.1 | 99.1 | 98.4 | 99.1 | 98.2 | 99.5 |
| p378k | 86.3 | 100.0 | 87.2 | 100.0 | 88.8 | 99.0 |
| p388k | 76.0 | 98.2 | 75.6 | 97.2 | 78.3 | 98.0 |
| p418k | 98.3 | 97.3 | 98.2 | 97.1 | 98.6 | 99.9 |

TABLE IV

PERCENTAGE OF SCAN CHAINS AND VECTORS TO BE MASKED ACCORDING TO EXACT ANALYSIS W.R.T. PESSIMISTIC SIMULATION FOR MIN., AVG., MAX. CONFIGURATION WITH 1.0% X-INPUT RATIO.

| Circuit | Indir. Impl. [33] avg #faults | max #faults | Proposed method avg #faults | max #faults |
|---|---|---|---|---|
| c2670 | 5.0 | 6.0 | 8.3 | 17.0 |
| c5315 | 0.4 | 4.0 | 8.0 | 29.0 |
| s5378 | 6.0 | 28.0 | 12.8 | 76.0 |
| s9234 | 3.0 | 8.0 | 14.7 | 39.0 |
| s13207 | 3.0 | 17.0 | 28.5 | 50.0 |
| s15850 | 16.0 | 77.0 | 45.7 | 114.0 |
| s35932 | 0.5 | 1.0 | 3.7 | 25.0 |
| s38417 | 10.0 | 79.0 | 78.5 | 159.0 |
| s38584 | 27.0 | 51.0 | 81.8 | 161.0 |

TABLE V

ADDITIONALLY DETECTED FAULTS IN PRESENCE OF X-VALUES IN ENHANCED FAULT SIMULATION BASED ON INDIRECT IMPLICATIONS [33] AND THE PROPOSED METHOD.

computed for X-masking based on 3-valued and exact analysis. The results show that the number of observable cells increases by the exact analysis, depending on the circuit. Generally, the increase is higher the higher the X-density is. While for p388k the increase is quite high at 8.22%, other circuits show only insignificant gain for any of the X-configurations.

The results show that a pessimistic X evaluation overestimates the amount of Xs in the circuit outputs. Applying an exact analysis is beneficial for all X-tolerating and X-masking schemes and helps to reduce overmasking significantly.

### *C. Comparison with Enhanced Fault Simulation based on Indirect Implications*

The stuck-at fault simulation method of [33] exploits indirect implications found by static learning. In an experiment similar to section VI-B1 and Table III, the authors of [33] investigate the number of additionally detected faults for a subset of ISCAS'85 and 89 circuits. The experiments consider 32 random patterns per circuit. The probability of an X-value at a particular circuit input is set to 50%.

These experiments are repeated with the proposed fault simulation algorithm. Table V presents the average and maximum number of additionally detected stuck-at faults per pattern w.r.t. 3-valued fault simulation for [33] and the proposed method. For instance, for circuit c5315 the method of [33] detects in average 0.4 faults per pattern in addition to 3-valued fault simulation, while the proposed method detects 8.0 additional faults. Similarly, the maximum number of additionally detected faults per pattern improves by a factor of 7. For all circuits, the average and maximum number of additionally detected faults is improved.

### *D. Increase of Stuck-at Fault Coverage*

The fault simulation algorithm has been applied to the industrial circuits provided by NXP. For these circuits, we investigate the fault classification of random pattern resistant stuck-at faults. For each circuit, the X-source configurations of Table II are evaluated.

A collapsed list of stuck-at faults in the support [41] of the X-valued inputs is generated. The faults are restricted to this subset since the accurate analysis of the X-valued signals only affects the support of X-valued inputs. Random pattern testable faults are removed by 3-valued fault simulation of 10000 random patterns. For the remaining hard faults, a commercial tool is used to generate X-aware deterministic patterns with high abort limit. The fault coverage of the test set is computed with classical 3-valued fault simulation. A loose upper bound of fault coverage is computed using the technique discussed in section VI-A. Finally, the fault coverage is computed using the proposed algorithm.

Table VI shows the results for three X-source ratios. The columns show the number of additionally detected stuck-at faults in the proposed algorithm compared to 3-valued fault simulation (FC Inc.) and the achieved accuracy w.r.t. the method of section VI-A (Acc.) for the considered X-source configurations.

The algorithm is able to classify a large number of additional hard faults as detected by the generated test set when compared to 3-valued fault simulation. The number reaches up to 20955 faults for circuit p378k, where ATPG could only reach a very low fault coverage. For many circuits and X-source configurations, the accuracy of the proposed algorithm exceeds 70% w.r.t. the loose upper bound. Results for higher X-source ratios are reported in [42].

### *E. Comparison with Restricted Symbolic Simulation*

Restricted symbolic simulation (RSS) is very effective in the analysis of simple local reconvergences of REX values. However, with higher number of X sources in the circuit, the pessimism of RSS increases as well. For the larger circuits of the considered benchmarks, Table VII shows for X-input ratios 1.0%, 2.0% and 5.0% the per-cent increase of identified FEX valued outputs of the proposed exact logic simulation algorithm w.r.t. RSS. The X-input configurations have been computed as described in section VI-B1. The increase of identified FEX valued outputs is typically higher for larger numbers of X-sources and ranges from 0.0% to 22.7% for the considered circuits. The maximum increase of 162.5% has been observed for circuit p77k and 1.0% X-input ratio, where

| Circuit | X-inp. [%] | Conf. min FC Inc. | Conf. min Acc. | Conf. avg FC Inc. | Conf. avg Acc. | Conf. max FC Inc. | Conf. max Acc. |
|---|---|---|---|---|---|---|---|
| | 0.5 | 0 | 0.0 | 0 | 100.0 | 0 | 100.0 |
| b17 | 1.0 | 0 | 0.0 | 1 | 1.6 | 1 | 25.0 |
| | 2.0 | 0 | 0.0 | 1 | 1.9 | 5 | 38.5 |
| | 0.5 | 0 | 100.0 | 1 | 8.3 | 1 | 5.0 |
| b18 | 1.0 | 662 | 74.7 | 480 | 68.3 | 14 | 46.7 |
| | 2.0 | 261 | 55.2 | 753 | 27.2 | 348 | 47.4 |
| | 0.5 | 3 | 1.8 | 0 | 0.0 | 1 | 1.1 |
| b19 | 1.0 | 0 | 0.0 | 6 | 6.4 | 213 | 54.2 |
| | 2.0 | 2573 | 59.9 | 314 | 37.6 | 51 | 9.0 |
| | 0.5 | 23 | 57.5 | 76 | 65.5 | 0 | 0.0 |
| p77k | 1.0 | 32 | 45.1 | 3 | 9.7 | 2 | 4.3 |
| | 2.0 | 65 | 31.9 | 751 | 70.8 | 513 | 75.1 |
| | 0.5 | 66 | 62.3 | 484 | 53.7 | 503 | 46.6 |
| p78k | 1.0 | 440 | 42.3 | 367 | 44.2 | 1960 | 55.8 |
| | 2.0 | 1827 | 48.4 | 981 | 43.2 | 3444 | 53.0 |
| | 0.5 | 144 | 11.7 | 9 | 0.7 | 37 | 3.5 |
| p81k | 1.0 | 72 | 2.9 | 12 | 0.6 | 173 | 7.6 |
| | 2.0 | 96 | 2.3 | 210 | 5.1 | 241 | 6.7 |
| | 0.5 | 168 | 53.2 | 98 | 23.5 | 5 | 5.4 |
| p89k | 1.0 | 107 | 48.9 | 192 | 28.3 | 288 | 63.0 |
| | 2.0 | 141 | 47.2 | 78 | 25.2 | 1996 | 70.4 |
| | 0.5 | 93 | 74.4 | 265 | 58.5 | 19 | 7.7 |
| p100k | 1.0 | 20 | 12.8 | 83 | 24.8 | 543 | 68.4 |
| | 2.0 | 166 | 21.9 | 123 | 20.9 | 1867 | 80.9 |
| | 0.5 | 303 | 78.1 | 189 | 44.3 | 2937 | 96.1 |
| p141k | 1.0 | 258 | 58.9 | 1058 | 92.3 | 479 | 75.7 |
| | 2.0 | 1119 | 82.3 | 821 | 65.6 | 3677 | 94.3 |
| | 0.5 | 1101 | 29.6 | 1089 | 37.6 | 820 | 29.3 |
| p239k | 1.0 | 1520 | 23.2 | 663 | 13.9 | 4797 | 39.8 |
| | 2.0 | 2025 | 18.0 | 3821 | 28.8 | 4431 | 28.0 |
| | 0.5 | 506 | 26.1 | 1246 | 25.9 | 1535 | 37.7 |
| p259k | 1.0 | 3149 | 37.0 | 2741 | 33.8 | 2649 | 31.2 |
| | 2.0 | 851 | 0.8 | 2142 | 18.0 | 3688 | 28.2 |
| | 0.5 | 52 | 16.6 | 331 | 36.7 | 76 | 40.9 |
| p267k | 1.0 | 274 | 44.7 | 286 | 66.7 | 1405 | 62.0 |
| | 2.0 | 9 | 1.2 | 996 | 42.3 | 1906 | 46.1 |
| | 0.5 | 2 | 1.6 | 262 | 35.6 | 634 | 29.8 |
| p269k | 1.0 | 13 | 2.8 | 229 | 34.5 | 260 | 33.6 |
| | 2.0 | 1907 | 63.9 | 132 | 11.1 | 1265 | 48.1 |
| | 0.5 | 13 | 15.1 | 191 | 72.6 | 205 | 85.8 |
| p279k | 1.0 | 47 | 23.3 | 305 | 63.9 | 1688 | 81.1 |
| | 2.0 | 319 | 48.9 | 1986 | 81.5 | 3607 | 82.2 |
| | 0.5 | 19 | 57.6 | 36 | 100.0 | 90 | 83.3 |
| p295k | 1.0 | 28 | 48.3 | 110 | 57.6 | 1504 | 95.6 |
| | 2.0 | 132 | 68.0 | 731 | 62.9 | 825 | 77.8 |
| | 0.5 | 942 | 86.2 | 358 | 52.7 | 4597 | 85.1 |
| p330k | 1.0 | 2553 | 73.7 | 4421 | 65.0 | 2875 | 79.1 |
| | 2.0 | 3815 | 72.4 | 4654 | 68.9 | 18110 | 86.4 |
| | 0.5 | 1332 | 45.4 | 3886 | 49.9 | 3793 | 55.8 |
| p378k | 1.0 | 6971 | 55.1 | 7415 | 57.4 | 10484 | 54.0 |
| | 2.0 | 20955 | 60.4 | 18789 | 58.7 | 13841 | 55.2 |
| | 0.5 | 2456 | 80.8 | 735 | 46.4 | 3130 | 75.2 |
| p388k | 1.0 | 3862 | 57.5 | 5583 | 68.7 | 3875 | 76.9 |
| | 2.0 | 2416 | 52.0 | 5658 | 46.1 | 9029 | 59.4 |
| | 0.5 | 91 | 14.9 | 606 | 57.4 | 646 | 57.7 |
| p418k | 1.0 | 1141 | 47.8 | 4277 | 83.0 | 2282 | 66.8 |
| | 2.0 | 1724 | 58.5 | 2514 | 59.5 | 1905 | 48.3 |

TABLE VI

INCREASE IN DETECTED STUCK-AT FAULTS AND ACHIEVED ACCURACY FOR THE X-SOURCE CONFIGURATIONS OF TABLE II

.

an exact analysis uncovers more than 2.6 times as many FEX valued outputs than RSS.

With regard to fault simulation, RSS can only identify a subset of the detectable faults classified by the proposed algorithm correctly. For the larger circuits, the fault coverage is computed using an RSS-based algorithm without the proposed exact analysis of the fault-free circuit. The number of additional detectable faults for the test set over the RSS-based

| Circuit | X-input ratio 1.0% FEX increase [%] min | avg | max | X-input ratio 2.0% FEX increase [%] min | avg | max | X-input ratio 5.0% FEX increase [%] min | avg | max |
|---|---|---|---|---|---|---|---|---|---|
| b19 | 5.8 | 8.3 | 0.0 | 16.8 | 9.9 | 8.5 | 12.7 | 5.0 | 12.7 |
| p378k | 6.0 | 3.4 | 4.2 | 5.5 | 15.1 | 8.9 | 19.5 | 18.0 | 22.7 |
| p388k | 1.6 | 3.2 | 1.2 | 5.3 | 2.4 | 3.1 | 5.4 | 0.6 | 3.1 |
| p418k | 0.1 | 3.2 | 0.0 | 4.7 | 3.3 | 0.9 | 14.9 | 5.9 | 5.8 |

TABLE VII

INCREASE IN FOUND FEX VALUED OUTPUTS IN EXACT LOGIC SIMULATION W.R.T. RSS.

algorithm is given in Table VIII. Depending on circuit and X-configuration, the proposed SAT-based algorithm classified up to 42270 faults more than the RSS-based computation of fault coverage.

| Circ. | X-input ratio 1.0% $\Delta_{RSS}$ det. faults min | avg | max | X-input ratio 2.0% $\Delta_{RSS}$ det. faults min | avg | max | X-input ratio 5.0% $\Delta_{RSS}$ det. faults min | avg | max |
|---|---|---|---|---|---|---|---|---|---|
| b19 | 0 | 0 | 0 | 0 | 0 | 0 | 900 | 408 | 97 |
| p378k | 275 | 127 | 724 | 1869 | 2761 | 840 | 42270 | 5712 | 5331 |
| p388k | 379 | 110 | 192 | 256 | 785 | 2055 | 1222 | 2801 | 3692 |
| p418k | 0 | 6 | 0 | 1 | 104 | 55 | 138 | 313 | 356 |

TABLE VIII

NUMBER OF ADDITIONAL DETECTABLE FAULTS IN THE PROPOSED ALGORITHM COMPARED TO RSS-BASED FAULT COVERAGE COMPUTATION

### F. Computing Time

The algorithm has been implemented in a Java based EDA framework using the SAT4J SAT solver [43]. All experiments have been conducted on an Intel Xeon CPU with 2.8 GHz frequency. The runtime is in the order of ATPG, ranging from a few minutes for smaller circuits up to a few hours. The runtime for fault simulation for a single pattern including the construction of the SAT instance (in sec.) and memory consumption (in MB) is given in Table IX for the X-source configurations of Table II. The runtime of the proposed algorithm depends on the circuit size, the number of patterns to be evaluated, and the number of X-sources. As shown in the table, runtime increases with the number of X sources. For very high numbers of X-sources, many internal signals have a REX value. This increases both the size of each SAT instance, as well as the number of instances to be evaluated. This causes the relatively high runtime for circuit p378k and p388k for the 2.0% X-input ratio. In the worst case, a redundancy evaluation is performed for each signal reconvergence in the circuit for each considered pattern.

The cones with X-valued signals which are mapped to a SAT instance are rather small. In the experiments with industrial circuits, the number of gates included in a SAT instance ranges on average per circuit from 453 gates in circuit p89k to 22779 gates in circuit p378k, including the cone copy. The resulting SAT instances including the clauses for comparison range from 1826 to 115662 clauses. Maximum memory consumption reaches 2.4GB for circuit p378k.

## VII. CONCLUSION

In this paper, the influence of pessimistic evaluation of X-propagation in $n$-valued logic and stuck-at fault simulation

| Circuit | X-inp. [%] | Runtime per pattern (sec) Conf. min | Conf. avg | Conf. max | Mem. (MB) |
|---|---|---|---|---|---|
| b17 | 0.5 | 0.7 | 0.8 | 1.2 | 475 |
| | 1.0 | 0.8 | 0.8 | 0.8 | 779 |
| | 2.0 | 0.7 | 0.6 | 0.8 | 609 |
| b18 | 0.5 | 1.3 | 8.0 | 4.1 | 1136 |
| | 1.0 | 2.9 | 3.6 | 9.4 | 1494 |
| | 2.0 | 7.1 | 9.5 | 16.8 | 1890 |
| b19 | 0.5 | 12.4 | 13.5 | 15.7 | 1278 |
| | 1.0 | 6.0 | 12.6 | 33.7 | 1224 |
| | 2.0 | 27.5 | 23.2 | 58.0 | 1573 |
| p77k | 0.5 | 0.8 | 3.8 | 3.1 | 624 |
| | 1.0 | 3.8 | 6.2 | 1.3 | 1158 |
| | 2.0 | 3.8 | 5.6 | 49.4 | 1019 |
| p78k | 0.5 | 2.7 | 3.8 | 4.4 | 621 |
| | 1.0 | 3.5 | 6.1 | 6.1 | 1005 |
| | 2.0 | 5.0 | 2.6 | 7.6 | 1356 |
| p81k | 0.5 | 5.7 | 5.1 | 7.9 | 1086 |
| | 1.0 | 6.9 | 6.6 | 7.0 | 1168 |
| | 2.0 | 11.4 | 9.5 | 8.3 | 1445 |
| p89k | 0.5 | 2.8 | 2.9 | 2.9 | 581 |
| | 1.0 | 1.8 | 2.4 | 3.2 | 550 |
| | 2.0 | 0.8 | 1.3 | 1.3 | 1108 |
| p100k | 0.5 | 3.4 | 4.8 | 5.0 | 984 |
| | 1.0 | 4.1 | 5.3 | 6.5 | 989 |
| | 2.0 | 3.4 | 3.7 | 4.7 | 1445 |
| p141k | 0.5 | 10.6 | 12.4 | 11.1 | 1119 |
| | 1.0 | 9.0 | 9.6 | 11.4 | 801 |
| | 2.0 | 4.9 | 7.0 | 12.3 | 1846 |
| p239k | 0.5 | 6.1 | 4.7 | 10.5 | 1427 |
| | 1.0 | 10.7 | 10.9 | 12.9 | 1443 |
| | 2.0 | 62.7 | 24.5 | 30.6 | 1248 |
| p259k | 0.5 | 10.8 | 14.3 | 11.1 | 1289 |
| | 1.0 | 14.5 | 18.0 | 96.2 | 1379 |
| | 2.0 | 98.2 | 46.1 | 46.3 | 1551 |
| p267k | 0.5 | 17.5 | 23.3 | 19.5 | 1301 |
| | 1.0 | 23.0 | 16.8 | 18.0 | 1097 |
| | 2.0 | 3.6 | 5.8 | 3.9 | 1599 |
| p269k | 0.5 | 16.7 | 19.9 | 20.5 | 1437 |
| | 1.0 | 22.2 | 18.4 | 16.9 | 1109 |
| | 2.0 | 8.8 | 9.0 | 10.1 | 1486 |
| p279k | 0.5 | 18.4 | 21.2 | 25.6 | 1389 |
| | 1.0 | 24.9 | 24.0 | 24.8 | 1306 |
| | 2.0 | 7.5 | 12.7 | 15.2 | 1674 |
| p295k | 0.5 | 18.1 | 19.9 | 20.0 | 1280 |
| | 1.0 | 5.1 | 5.6 | 6.0 | 1309 |
| | 2.0 | 4.4 | 7.6 | 7.9 | 1856 |
| p330k | 0.5 | 7.6 | 7.1 | 23.8 | 1769 |
| | 1.0 | 17.2 | 18.8 | 16.4 | 1643 |
| | 2.0 | 34.7 | 24.7 | 43.9 | 1722 |
| p378k | 0.5 | 12.1 | 20.2 | 27.9 | 1794 |
| | 1.0 | 45.7 | 58.6 | 67.6 | 2429 |
| | 2.0 | 151.1 | 220.3 | 92.5 | 2063 |
| p388k | 0.5 | 22.0 | 13.3 | 14.5 | 2085 |
| | 1.0 | 31.1 | 30.6 | 27.0 | 2182 |
| | 2.0 | 86.0 | 81.8 | 163.2 | 2282 |
| p418k | 0.5 | 11.9 | 12.9 | 13.8 | 1829 |
| | 1.0 | 15.7 | 17.2 | 19.7 | 1958 |
| | 2.0 | 15.1 | 22.2 | 30.2 | 2042 |

TABLE IX
FAULT SIMULATION RUNTIME IN SEC PER PATTERN AND MEMORY CONSUMPTION FOR THE X-SOURCE CONFIGURATIONS OF TABLE II.

was examined. An accurate logic simulation algorithm using SAT-based reasoning to exactly identify the signals with real X-values in the circuit was presented. This algorithm was extended to compute the fault coverage of a test set with high accuracy in presence of unknown values.

In the experiments the algorithm was applied to industrial circuits. It was shown that the stuck-at fault coverage of test sets can be significantly higher than estimated by classical fault simulation, improving product quality without incurring any additional test or hardware overhead. The logic simulation algorithm was used to analyze the impact of pessimistic simulation on X-handling schemes. It was shown that X-handling efficiency can be improved significantly, if exact methods are used for X-propagation assessment.

## REFERENCES


[1] E. Ulrich and T. Baker, “The concurrent simulation of nearly identical digital networks,” in *Proc. 10th Workshop on Design Automation*, 1973, pp. 145–150.

[2] J. Waicukauski, E. Eichelberger *et al.*, “Fault simulation for structured VLSI,” *VLSI Systems Design*, vol. 6, no. 12, pp. 20–32, 1985.

[3] K. Antreich and M. H. Schulz, “Accelerated fault simulation and fault grading in combinational circuits,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 6, no. 5, pp. 704–712, 1987.

[4] H. K. Lee and D. S. Ha, “An efficient, forward fault simulation algorithm based on the parallel pattern single fault propagation,” in *Proc. IEEE International Test Conference*, 1991, pp. 946–955.

[5] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital systems testing and testable design*. IEEE Press, 1994.

[6] R. E. Bryant, “Graph-based algorithms for boolean function manipulation,” *IEEE Trans. on Computers*, vol. 35, no. 8, pp. 677–691, 1986.

[7] S. Chandra and J. Patel, “Accurate logic simulation in the presence of unknowns,” in *Proc. International Conference on Computer-Aided Design*, 1989, pp. 34–37.

[8] H.-Z. Chou, K.-H. Chang, and S.-Y. Kuo, “Accurately handle don’t-care conditions in high-level designs and application for reducing initialized registers,” *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 29, no. 4, pp. 646–651, 2010.

[9] A. Jain, V. Boppana *et al.*, “Testing, verification, and diagnosis in the presence of unknowns,” in *Proc. VLSI Test Symposium*, 2000, pp. 263–268.

[10] C. Scholl and B. Becker, “Checking equivalence for partial implementations,” in *Proc. Design Automation Conference (DAC)*, 2001, pp. 238–243.

[11] C. Wilson, D. Dill, and R. Bryant, “Symbolic simulation with approximate values,” in *Formal Methods in Computer-Aided Design*, ser. Lecture Notes in Computer Science, W. Hunt and S. Johnson, Eds. Springer Berlin / Heidelberg, 2000, vol. 1954, pp. 507–522.

[12] J. Carter, B. Rosen *et al.*, “Restricted symbolic evaluation is fast and useful,” in *Proc. International Conference on Computer-Aided Design*, 1989, pp. 38–41.

[13] N. Sridhar and M. Hsiao, “On efficient error diagnosis of digital circuits,” in *Proc. IEEE International Test Conference*, 2001, pp. 678–687.

[14] S. Kundu, I. Nair *et al.*, “Symbolic implication in test generation,” in *Proc. Conference on European Design Automation*, 1991, pp. 492–496.

[15] M. H. Schulz, E. Trischler, and T. M. Sarfert, “Socrates: a highly efficient automatic test pattern generation system,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 126–137, 1988.

[16] W. Kunz, D. Stoffel, and P. Menon, “Logic optimization and equivalence checking by implication analysis,” *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 16, no. 3, pp. 266 –281, 1997.

[17] S. Kang and S. A. Szygenda, “Accurate logic simulation by overcoming the unknown value propagation problem,” *Simulation*, vol. 79, no. 2, pp. 59–68, 2003.

[18] Y. Han, Y. Hu *et al.*, “Theoretic analysis and enhanced X-tolerance of test response compact based on convolutional code,” in *Proc. ASP-DAC.*, 2005, pp. 53–58.

[19] S. Mitra and K. S. Kim, “X-compact: an efficient response compaction technique,” *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 23, no. 3, pp. 421–432, 2004.

[20] M. Sharma and W.-T. Cheng, “X-filter: Filtering unknowns from compacted test responses,” in *Proc. IEEE International Test Conference*, 2005, pp. 1090–1098.

[21] W. Rajski and J. Rajski, “Modular compactor of test responses,” in *Proc. IEEE VLSI Test Symposium.*, 2006, pp. 242–251.

[22] P. Wohl, J. Waicukauski, and S. Ramnath, “Fully X-tolerant combinational scan compression,” in *Proc. IEEE International Test Conference*, 2007, pp. 1–10.

[23] T. Rabenalt, M. Goessel, and A. Leininger, “Masking of X-values by use of a hierarchically configurable register,” in *Proc. 14th IEEE European Test Symposium*, 2009, pp. 149 –154.

[24] J. Rajski, J. Tyszer *et al.*, “Finite memory test response compactors for embedded test applications,” *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 622–634, 2005.

[25] G. Mrugalski, A. Pogiel *et al.*, “Diagnosis with convolutional compactors in presence of unknown states,” in *Proc. of the IEEE International Test Conference*, 2005, paper 16.3.

[26] R. Garg, R. Putman, and N. Touba, “Increasing output compaction in presence of unknowns using an X-canceling MISR with deterministic observation,” in *Proc. 26th IEEE VLSI Test Symposium*, 2008, pp. 35–42.

[27] M. Naruse, I. Pomeranz *et al.*, “On-chip compression of output responses with unknown values using LFSR reseeding,” in *Proc. of the IEEE International Test Conference*, 2003, pp. 1060–1068.

[28] M.-T. Chao, S. Wang *et al.*, “Response shaper: a novel technique to enhance unknown tolerance for output response compaction,” in *Proc. Int’l Conference on Computer-Aided Design*, 2005, pp. 80–87.

[29] V. Chickermane, B. Foutz, and B. Keller, “Channel masking synthesis for efficient on-chip test compression,” in *Proc. of the IEEE International Test Conference*, 2004, pp. 452–461.

[30] J. Rajski, J. Tyszer *et al.*, “X-press: Two-stage X-tolerant compactor with programmable selector,” *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 147–159, 2008.

[31] H. Tang, C. Wang *et al.*, “On efficient X-handling using a selective compaction scheme to achieve high test response compaction ratios,” in *Proc. International Conference on VLSI Design*, 2005, pp. 59–64.

[32] Y. Tang, H.-J. Wunderlich *et al.*, “X-masking during logic BIST and its impact on defect coverage,” *IEEE Trans. VLSI Systems*, vol. 14, no. 2, pp. 442–451, 2006.

[33] S. Kajihara, K. K. Saluja, and S. M. Reddy, “Enhanced 3-valued logic/fault simulation for full scan circuits using implicit logic values,” in *Proc. IEEE European Test Symposium (ETS)*, 2004, pp. 108–113.

[34] K. Cho and R. E. Bryant, “Test pattern generation for sequential MOS circuits by symbolic fault simulation,” in *Proc. ACM/IEEE Design Automation Conference*, 1989, pp. 418–423.

[35] M. Mojtahedi and W. Geisselhardt, “New methods for parallel pattern fast fault simulation for synchronous sequential circuits,” in *Proc. International Conference on Computer-Aided Design*, 1993, pp. 2–5.

[36] C. Pixley, S.-W. Jeong, and G. D. Hachtel, “Exact calculation of synchronization sequences based on binary decision diagrams,” in *Proc. Design Automation Conference (DAC)*, 1992, pp. 620–623.

[37] H. Cho, S.-W. Jeong *et al.*, “Synchronizing sequences and symbolic traversal techniques in test generation,” *Journal of Electronic Testing: Theory and Applications*, vol. 4, no. 1, pp. 19–31, 1993.

[38] B. Becker, M. Keim, and R. Krieger, “Hybrid fault simulation for synchronous sequential circuits,” *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 15, no. 3, pp. 219–238, 1999.

[39] M. Elm, M. A. Kochte, and H.-J. Wunderlich, “On determining the real output Xs by SAT-based reasoning,” in *Proc. IEEE Asian Test Symposium*, 2010, pp. 39–44.

[40] J. Rajski, J. Tyszer *et al.*, “X-press compactor for 1000x reduction of test data,” in *Proc. IEEE International Test Conference.*, 2006, pp. 1–10.

[41] I. Hamzaoglu and J. H. Patel, “New techniques for deterministic test pattern generation,” *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 15, no. 1/2, pp. 63–73, 1999.

[42] M. A. Kochte and H.-J. Wunderlich, “SAT-based fault coverage evaluation in the presence of unknown values,” in *Proc. Design, Automation and Test in Europe (DATE’11)*, 2011, pp. 1–6.

[43] D. Le Berre and A. Parrain, “The Sat4j library, release 2.2,” *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 7, 2010.