SAT-Based Fault Coverage Evaluation in the Presence of Unknown Values

Michael A. Kochte, Hans-Joachim Wunderlich
University of Stuttgart
Institute of Computer Architecture and Computer Engineering
Pfaffenwaldring 47, 70569 Stuttgart, Germany

***Abstract*—Fault simulation of digital circuits must correctly compute fault coverage to assess test and product quality. In case of unknown values (X-values), fault simulation is pessimistic and underestimates actual fault coverage, resulting in increased test time and data volume, as well as higher overhead for design-for-test. This work proposes a novel algorithm to determine fault coverage with significantly increased accuracy, offering increased fault coverage at no cost, or the reduction of test costs for the targeted coverage. The algorithm is compared to related work and evaluated on benchmark and industrial circuits.**

***Index Terms*—Unknown values, fault coverage, precise fault simulation**

I. Introduction

In Design-for-Test and test generation, as well as in test application, signal lines may take unknown (X) values. These X values result from partially specified test patterns, or stem from X sources within the circuit. X sources comprise uninitialized or uncontrollable sequential elements in the circuit, clock domain crossings or tristate circuitry. In sequential ATPG, the problem of uninitialized sequential elements and resulting X valued signals is known and has been solved by heuristic, precise and hybrid test generation algorithms, which are able to generate test sequences even in presence of X values.

Fault simulation algorithms such as the PPSFP (parallel pattern single fault propagation) and the concurrent algorithm [1–4] can be used to pessimistically estimate fault coverage for stuck-at and transition faults in presence of X values. Principally, these algorithms use an n-valued logic with limited number of symbols to compute the signal states in the fault-free and faulty circuit.

In presence of X values at circuit inputs or internal signals, a logic with a limited number of symbols is unable to correctly compute the propagation of X values in the circuit as it fails to correctly evaluate all reconvergences of X valued signals. An example is given in figure 1, where a 3-valued simulation computes the state of the output signal as X, while the signal actually takes the value 1. Consequently, in fault simulation

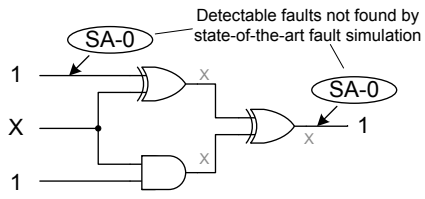

Fig. 1. Pessimism in 3-valued logic simulation and fault simulation

less faults can be classified as detected by a particular pattern since fault activation and propagation require well-defined logic values. State-of-the-art fault simulation cannot classify a single stuck-at fault in figure 1 as detected, because the faults are either not activated or propagated. A precise analysis however reveals that two stuck-at-0 faults are actually detectable with the given input pattern.

Correctly evaluating fault coverage in these cases is important to guarantee product quality with low defective part level. A less pessimistic analysis may also *increase fault coverage without inferring any costs* in terms of test data volume, test time or design-for-test hardware overhead. In contrast, this knowledge gives an additional potential for optimization during test generation and design-for-test, promising reduced overhead, or increased test quality without cost increase, or a combination thereof.

Yet the computational effort of a precise analysis is very high. Accurate logic simulation is already an NP-complete problem [5], and each faulty machine must be evaluated precisely w.r.t. each pattern. Here, we propose a novel algorithm to compute fault coverage with significantly increased accuracy and which is able to correctly classify a large number of faults otherwise marked as undetected by state-of-the-art fault simulation. This work focusses on the evaluation of fault coverage and test quality of test sets for BIST, embedded deterministic or external test in presence of X values resulting from both partially specified patterns and X sources internal to the circuit under test. In ATPG each fault needs to be considered only once for the pattern under construction. In contrast, fault simulation has to analyze all the generated patterns w.r.t. all faults, or at least all patterns until fault detection if fault dropping is used. Hence, for each fault, not a single NP-complete problem has to be solved, but multiple, and in the worst case as many as patterns are applied.

The reminder of this paper is organized as follows: The next section presents the related work in logic and fault simulation, followed by a concise problem statement. Section IV introduces the details of the proposed algorithm for improved fault coverage computation. Section V compares the proposed algorithm with related work, determines an upper bound on fault detection and evaluates the approach by using benchmark and industrial circuits. Finally, a short conclusion summarizes the work.

II. Related Work

Both approximative and exact methods have been proposed to overcome the pessimism in logic and fault simulation.

A. Logic Simulation with Increased Accuracy

Precise logic simulation in presence of X values can be mapped to symbolic simulation or expressed as a satisfiability instance. In a symbolic simulation of the circuit, the Boolean function of each signal is expressed symbolically. This can be implemented by use of reduced ordered binary decision diagrams (ROBDDs) [6]. ROBDDs provide a canonical representation of Boolean functions and allow a fast test whether

978-3-9810801-7-9/DATE11/©2011 EDAA

a signal depends on an X source or not. In this 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 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 an 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.

Determining the circuit behavior in presence of X values is related to the problem of formal verification of circuits with black boxes, where methods based on additional X constraints and symbolic simulation techniques are applied [9, 10].

Since the computation effort for formal methods may render the application to larger circuits impossible, pessimistic algorithms have been proposed which still offer increased precision compared to classical n-valued logic simulation. If the domain of symbols is restricted, as e.g. in [11] or applied to test generation as in [12], the simulation result cannot be exact. 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 [13, 14]. The idea stems from the ATPG domain and is based on evaluating the contrapositive of signals in the circuit. A learning critereon 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 approximative algorithm of [15] is based on circuit partitioning of reconvergent regions. A partition starts at an X-valued fanout stem and comprises all gates in the transitive fanout until the fanout signal reconverges. If the reconverged signal has an X-value in 3-valued simulation, the partition is subject to 3-valued logic simulated twice, once with the fanout stem set to 0 and then 1. The result at the reconvergence is then fed back to the simulation of the whole circuit.

B. Fault Simulation with Increased Accuracy

The use of indirect implications in fault simluation has been proposed by [16]. Due to the limitations of static learning methods, the achievable precision is limited, i.e. only a small subset of actually detectable faults is identified.

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 [17].

Especially for sequential circuits, where Xs originating from uninitialized sequential elements spread over the whole circuit and significantly impair fault coverage, symbolic simulation has been applied. Restricted symbolic simulation is used in [18] for synchronous sequential circuits.

The use of ROBDDs for exact simulation in synchronous sequential circuits has been proposed in [19]. Since this approach is limited by the memory requirements of the BDD, the authors of [20] propose a hybrid fault simulation approach which uses exact ROBDD-based symbolic simulation until the ROBDD memory requirement exceeds a given limit. In that case, the hybrid approach switches to conventional 3-valued simulation and potentially loses accuracy.

III. Problem Statement

This work targets the correct classification of stuck-at faults in fullscan or combinational circuits for a given set of test patterns in presence of unknown values from either partially specified patterns or X sources internal to the circuit. This section explains the used terminology and outlines an exact solution for this problem.

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

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

It follows that PEX = FEX $\cup$ REX and FEX $\cap$ REX = $\emptyset$.

To compute whether a fault in the circuit is detected by a given test pattern, the test responses of the fault free and faulty circuit must be computed and compared. One way to compute the exact response of the fault free circuit is enumerating all possible assignments of the X sources. If the output is constant for all assignments, it is a FEX and the corresponding output stuck-at 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 small. Symbolic approaches based on ROBDDs, e.g., may allow a larger number of X sources. However, they are not applicable for all circuits due to known shortcomings of ROBDDs, as for example dependence on variable ordering or exponential growth for multipliers. The following section describes a hybrid algorithm for the computation of fault coverage which is more pessimistic than the exact method but practically feasible even for large circuits.

IV. SAT-Based Evaluation of Fault Coverage

To reduce the high computational effort of a precise analysis of every fault in a circuit with unknown values as described in the previous section, the proposed algorithm uses an efficient SAT-based method to classify the signal states in the fault-free circuit under a given input pattern. The accurately computed signal states are then used as basis for the fault classification. This way, the serial analysis of all yet undetected faults for each pattern can be avoided. By trading-off computing time and precision 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 overall algorithm. Section IV-B explains the SAT-based classification of signal states, followed by the optimizations in the fault classification.

A. Overview

In contrast to pessimistic n-valued simulation, which only computes the set of PEX values, the proposed logic simulation algorithm is able to efficiently distinguish all REX and FEX values. This information is then exploited during fault analysis. The fault analysis step incorporates ideas from PPSFP fault simulation [2–4]. Figure 2 shows the overall flow of the algorithm. To determine the detected faults of a test set, each pattern is precisely simulated. This is performed by an initial pessimistic 3-valued logic simulation and by generating a SAT instance which is then processed by a SAT solver. Only

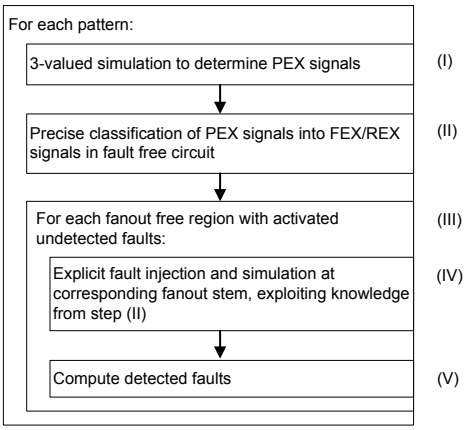

Fig. 2. Overview of fault coverage computation in presence of Xs

reconvergences of PEX values have to be handled by the SAT solver. The results are the sets of REX and FEX valued signals in the fault-free circuit (step I and II in the figure). Based on this information, the activated faults and the exact logic values in the fault free case are known. To determine the observability of activated faults in fanout-free regions of the circuit, the corresponding fanout stems are evaluated by explicit fault injection and simulation. For the simulation of the transitive fanout of the stems, however, computational effort and precision of fault simulation are traded off.

A fault effect will be propagated through the circuit and may turn an X-value into a constant 0 or 1, or vice versa. If an output is altered from a constant 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 false X 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.

To avoid excessive simulation times for the stems, a fast 3-valued logic simulation is performed in step IV. For the input values of the transitive fanout of the stem, the PEX classification from step II is used. The values of PEX signals whose state may be impacted by a fault are thus evaluated pessimistically. Finally, using stem observability and fault activation information, the detected faults are enumerated.

B. Accurate 3-Valued Simulation

As the first step of the proposed algorithm, a precise 3-valued logic simulation is performed which classifies all PEX signals accurately as either a FEX or REX signal. 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.

The simulation algorithm is a hybrid approach that combines reconvergence analysis, event-based logic simulation and exact SAT-based analysis. FEX signals can only emerge if a REX is propagated along multiple paths which reconverge at a gate as illustrated in the example in Fig. 1. The convergence of unrelated unknown values cannot produce a FEX state. With the information of a reconvergence analysis, a SAT instance is generated which is used for the exact computation of the signal states at FEX reconvergences under the given input stimuli. A breadth-first traversal of the subset of PEX-valued signals, starting at the circuit inputs, invokes the SAT-based evaluation at REX reconvergences. 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.

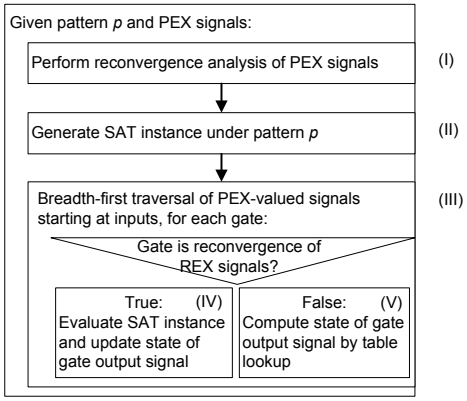

Fig. 3. Precise logic simulation flow

1) Reconvergence Analysis: From 3-valued simulation, the set of PEX-valued signals is known. At PEX-valued fanout stems, a quick reconvergence analysis is executed. This analysis is implemented as an event-based forward traversal of the PEX signals starting from the fanout stem. It derives the list of gates where disjoint paths reconverge. Found reconvergences are stored for subsequent generation of the SAT instance.

2) Generation of the SAT Instance for Signal Classification: A SAT solver is used to exactly classify the PEX-valued signals in the circuit in presence of X-sources. For a considered PEX reconvergence, a SAT instance is constructed that is satisfiable if and only if the reconverged signal is a real X (REX). This is achieved by searching for two assignments to the X sources of the circuit that result in complementary values at the reconverged signal.

A single SAT instance is generated for each pattern and evaluated under different unitary constraints such that all PEX reconvergences for that pattern are classified. The SAT instance does not comprise all gates of the circuit, but is restricted to those gates which generate PEX-values at their output. The SAT instance models two copies of these gates. Additional clauses are introduced at PEX 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 CUT and s' in the copy have different values. To allow 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 $\{s, s', \neg s_{SEL}\}, \{\neg s, \neg s', \neg s_{SEL}\}$ are added to the SAT instance for each PEX reconvergence.

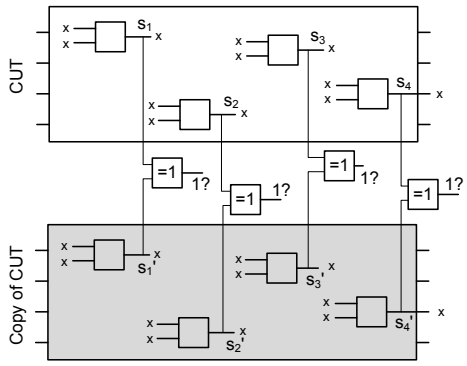

Fig. 4. Principle of signal evaluation at PEX reconvergences $s_i \in S_x^R$

To evaluate a particular PEX reconvergence s, the corresponding selector variable s_{SEL} is contrained to false and all other selector variables are constrained to true. 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 real X (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.

3) Exact Logic Simulation: The exact logic simulation of the circuit with a given pattern is performed by a breadth-first traversal of the PEX-valued signals in the circuit, starting from the inputs. The PEX signals are classified either by invoking the SAT solver (c.f. step II) or by forward implication based logic simulation. The SAT solver is invoked at a reconvergence if and only if at least two inputs of the corresponding gate have been classified as REX values. If only one of the inputs is a REX and the others are constant signals including FEX, the correct output value can be derived without the SAT solver.

The result of the SAT-based computation (step IV) is then propagated along the unclassified PEX signals towards the circuit outputs as far as possible. A table-based lookup simulation is used for this evaluation. Propagation stops at convergences with X-valued signals or at circuit outputs. The exact simulation ends once all PEX signals have been visited and classified.

C. Fault Classification

The signal classification gained in the previous step is used in the fault classification step to increase the accuracy compared to n-valued fault simulation. By applying algorithmic optimizations as found in state-of-the-art fault simulators, the efficiency of this step is increased: Fault activation in fanout-free regions is evaluated separately from fault propagation from the corresponding fanout stem [2–4].

For each fanout-free region, the activation of each fault in the region is determined using the signal classification from the exact 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. Thus, local propagation is correctly computed using the values from the exact logic simulation.

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 pessimistic 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 values from the exact logic simulation of step are used.

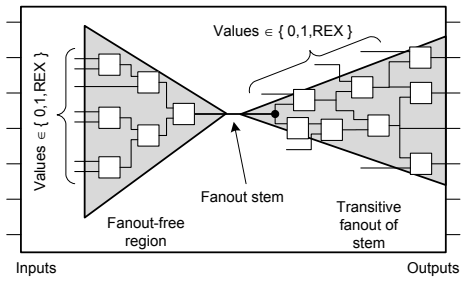

Fig. 5. Computation of fanout stem observability and fault detection in fanout free regions

V. 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. Here we assume that X values occur at the pseudo primary inputs of the circuit.

A. Verification by Exhaustively Filled Patterns

To verify the correctness of the algorithm, a simulation based strategy is applied. 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 resp. fault simulation. If lines carry a constant signal for all the patterns, they cannot be a REX, and if a fault is detected by all the patterns, it is a definite detect. For both cases, verification of the proposed algorithm was successful.

For larger circuits and patterns with more than 16 X-valued inputs, the required computing effort prohibits the exhaustive verfication. 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. Again, no mismatches between the result of the proposed algorithm and the logic resp. fault simulation were found.

In addition, the fault simulation of 128 filled patterns (originating from a single partially specified pattern), followed by intersection of the set of faults detected by each pattern, allows to derive an upper bound of the number of additionally detectable faults in the circuit due to precise analysis. This

information is used to assess the precision of the proposed algorithm (c.f. section V-C). Due to the small sample size of 128 patterns only, the computed coverage is an optimistic upper bound. Thus, the derived precision of the proposed algorithm is a pessimistic lower bound of its actual precision.

B. Comparison with Enhanced Fault Simulation based on Indirect Implications

The fault simulation method of [16] exploits indirect implications found by static learning. In [16] the authors report the number of identified FEX signals, FEX outputs and additionally detected faults for a subset of ISCAS'85 and 89 circuits. The conducted experiments consider 32 randomized input patterns per circuit. The probability of an X-value at a particular circuit input is set to 50%. The authors report the average and maximum number of FEX signals and outputs over these 32 patterns and the improvement of fault coverage over 3-valued fault simulation.

These experiments are repeated with the proposed algorithm. However, the number of patterns is increased to 128 to limit the impact of outliers in the much smaller set of 32 patterns. Table I presents the results w.r.t. identified FEX values at all the signals, at the (pseudo primary) outputs, and the improved fault coverage. The first number in each column is the ratio of the average number of identified FEX values per pattern of the 128 patterns evaluated with the proposed algorithm, and the 32 patterns of [16]. The second number is the best value out of the 128 or 32 patterns, resp., which is biased in favor of our approach due to the larger sample size. For instance, the last entry of circuit c5315 denotes that in average we can classify 17.01X times more possibly detected faults as definitely detected than [16], and if we compare the best outcomes of both series, the improvement is 9.25X.

Circuit	FEX signals	FEX outputs	Fault detection
c2670	2.33, 8.52	2.34, 5.50	1.60, 18.50
c5315	18.06, 13.60	15.46, 7.50	17.01, 9.25
s5378	1.97, 1.49	2.08, 1.90	1.89, 2.85
s9234	6.40, 8.44	3.35, 4.66	5.06, 6.37
s13207	4.64, 3.29	6.79, 3.00	10.21, 6.58
s15850	2.54, 1.75	2.80, 2.31	2.64, 2.02
s35932	6.79, 5.65	24.28, 24.00	5.40, 34.00
s38417	4.40, 2.73	23.40, 6.81	7.87, 2.16
s38584	2.59, 4.33	2.80, 2.54	3.77, 5.05

TABLE I
IMPROVEMENT OVER ENHANCED FAULT SIMULATION BASED ON INDIRECT IMPLICATIONS [16]

For all circuits, the proposed algorithm is able to classify more FEX signals and outputs, for c5315 as example, the average number of FEX signals increases by 1706% over the result of [16]. For this circuit, the average fault coverage increase over all patterns rises by 1601% over the fault coverage increase by [16].

C. Results on Industrial Circuits

The algorithm has been applied to a wide range of industrial circuits provided by NXP. For these circuits, we investigated the fault classification of random pattern resistant faults for multiple configurations.

For each circuit, different X-source configurations are evaluated. In each X-source configuration, a fixed set of pseudo primary inputs is randomly selected as X-sources. Then, a collapsed list of stuck-at faults in the support [21] of the X-valued inputs is generated. The faults are restricted to this subset since the precise 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. An upper bound of fault coverage is computed using the validation technique discussed in section V-A. Finally, the fault coverage is computed using the proposed algorithm.

In the configurations, 0.5%, 1.0%, 2.0% and 5.0% of the inputs are randomly selected as X-sources. 16 different configurations are generated for each of these four X-source ratios. From these 16 configurations per circuit and X input ratio, results for two configurations A and B are selected and reported here. The selection of A and B is based on the number of discovered FEX valued signals in the circuit during the precise logic simulation. The number of discovered FEXs is a coarse indicator of the number of faults that can be detected in addition to n-valued fault simulation. In configuration A the number of discovered FEXs is close to the average number of discovered FEXs over all 16 configurations. Configuration B has the maximum number of discovered FEX signals.

Table II shows the results for the industrial circuits and the four different X-source ratios. For each circuit, the number of pseudo primary inputs and gates are given in column 2 and 3. The following columns present the number of additionally detected faults in the proposed algorithm compared to 3-valued fault simulation and the achieved precision w.r.t. the method of section V-A for each X-source ratio.

The algorithm is able to classify a large number of additional hard faults as detected by the generated test set compared to 3-valued fault simulation. The number of additionally detected faults depends on the particular circuit, X-source configuration as well as the test set. The number reaches up to 209031 uncollapsed faults for circuit p286k, where ATPG could only reach a fault coverage of 10.19%. For many circuits and X-source configurations, the precision of the proposed algorithm exceeds 70% w.r.t. the optimistic upper bound.

The runtime of the proposed method depends on the circuit size, the number of patterns to be evaluated, and the number of X sources. The runtime is in the order of ATPG for most circuits, e.g. 452 up to 7042 sec for circuit p100k for configurations with 0.5% and 5% X-input ratio. With increasing circuit size, patterns and X sources, the runtimes reach a couple of hours for larger circuits, e.g. 48676 sec for circuit p418k and a configuration with 2% X-input ratio. Memory usage from 262 up to 1849 MB. The algorithm has been implemented in JAVA. First experiments with a C++ SAT solver promise a speedup of 2x-3x.

VI. CONCLUSIONS

Computation of fault coverage of a test set in presence of unknown / X values results in a pessimistic under-estimation if algorithms based on n-valued logic are used. This work presented a SAT-based algorithm to compute fault coverage with high precision. The algorithm is applicable to large, recent industrial circuits and substantially improves the state-of-the art. Experiments show that fault coverage of test sets can be significantly higher than estimated by classical fault simulation. The algorithm can be easily modified to target transition fault coverage.

Circuit	Num. PPI	Num. Gates	Conf. Type	0.5% FC Inc.	0.5% Prec.	1.0% FC Inc.	1.0% Prec.	2.0% FC Inc.	2.0% Prec.	5.0% FC Inc.	5.0% Prec.
p35k	2912	41443	A	20	90.9%	8	100.0%	187	55.0%	239	88.2%
			B	33	100.0%	159	42.0%	379	99.0%	271	53.8%
p45k	3739	38811	A	28	19.6%	16	27.6%	244	36.4%	633	29.4%
			B	231	67.5%	369	55.7%	13	20.0%	94	40.2%
p77k	3487	65015	A	0	0.0%	260	21.3%	127	49.8%	1043	73.6%
			B	482	17.1%	19774	99.7%	608	22.5%	545	66.0%
p78k	3148	68263	A	492	43.5%	768	43.6%	2318	49.4%	5463	51.9%
			B	255	49.6%	4759	86.5%	2697	53.8%	8173	55.6%
p81k	4029	106450	A	23	2.3%	77	4.1%	99	2.6%	611	6.6%
			B	8	0.8%	51	2.0%	79	2.5%	553	6.3%
p89k	4632	80963	A	113	60.4%	328	66.4%	204	33.6%	1841	80.5%
			B	68	67.3%	1737	94.6%	2434	91.6%	10593	96.0%
p100k	5902	84356	A	12	6.2%	113	26.3%	378	37.6%	79	5.7%
			B	575	98.6%	9462	95.5%	6933	70.2%	2546	62.6%
p141k	11290	152808	A	568	91.2%	850	87.6%	827	81.2%	5090	89.0%
			B	2154	94.7%	20417	91.8%	1914	62.8%	20626	89.1%
p239k	18692	224597	A	473	25.8%	1657	25.4%	2861	22.3%	7609	26.3%
			B	844	20.5%	908	21.3%	8251	35.2%	5951	19.6%
p259k	18713	298796	A	384	18.2%	2795	33.4%	5594	37.3%	11205	28.6%
			B	493	19.6%	2279	29.4%	4534	33.1%	6495	23.8%
p267k	17332	238697	A	97	36.7%	1336	51.3%	1201	60.4%	3869	71.1%
			B	320	53.2%	1682	17.2%	604	48.1%	6054	75.0%
p269k	17333	239771	A	550	63.1%	91	18.0%	275	26.5%	4812	64.5%
			B	771	61.9%	469	59.4%	1678	51.6%	928	60.8%
p279k	18074	257736	A	215	83.7%	660	65.0%	2003	68.7%	4292	74.1%
			B	623	64.4%	1158	85.9%	1989	89.6%	15100	76.3%
p286k	18351	332726	A	321	75.7%	1944	90.5%	1324	89.2%	3188	77.6%
			B	220	98.7%	209031	83.3%	4596	92.0%	13276	92.7%
p295k	18508	249747	A	138	61.3%	269	89.1%	595	87.1%	1863	45.3%
			B	1129	98.7%	738	88.6%	244	59.5%	122	95.3%
p330k	18010	312666	A	2712	85.0%	2803	70.0%	3246	65.6%	10899	63.0%
			B	3448	87.0%	9459	89.3%	5940	75.1%	18044	67.3%
p378k	15732	341315	A	2314	43.9%	5873	53.0%	14386	56.6%	34203	57.7%
			B	4591	53.1%	10534	58.1%	15579	56.7%	17814	57.8%
p388k	25005	433331	A	1935	74.1%	3995	77.7%	6423	52.3%	14986	64.0%
			B	1049	77.7%	6424	86.2%	58665	93.4%	68691	89.2%
p418k	30430	382633	A	553	46.6%	685	51.9%	3000	61.1%	4996	46.5%
			B	865	56.3%	1969	52.2%	5577	57.6%	7233	57.3%

TABLE II
INCREASE IN DETECTED FAULTS AND ACHIEVED PRECISION FOR DIFFERENT X-SOURCE RATIOS

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] H. P. Chang and J. A. Abraham, "The complexity of accurate logic simulation," in *Proc. Int'l Conference on Computer-Aided Design*, 1987.

[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*, 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. VTS*, 2000, pp. 263–268.

[10] T. Nopper, C. Scholl, and B. Becker, "Computation of minimal counterexamples by using black box techniques and symbolic methods," in *Proc. ICCAD*, 2007, pp. 273–280.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.