Efficient BDD-based Fault Simulation in Presence of Unknown Values

Kochte, Michael A.; Kundu, S.; Miyase, Kohei; Wen, Xiaoqing; Wunderlich, Hans-Joachim

Proceedings of the 20th IEEE Asian Test Symposium (ATS'11) New Delhi, India, 20-23 November 2011

doi: http://dx.doi.org/10.1109/ATS.2011.52

Abstract: Unknown (X) values, originating from memories, clock domain boundaries or A/D interfaces, may compromise test signatures and fault coverage. Classical logic and fault simulation algorithms are pessimistic w.r.t. the propagation of X values in the circuit. This work proposes efficient hybrid logic and stuck-at fault simulation algorithms which combine heuristics and local BDDs to increase simulation accuracy. Experimental results on benchmark and large industrial circuits show significantly increased fault coverage and low runtime. The achieved simulation precision is quantified for the first time.

Preprint

Efficient BDD-based Fault Simulation in Presence of Unknown Values

Michael A. Kochte[†,‡], Sandip Kundu[§], Kohei Miyase[‡], Xiaoqing Wen[‡], Hans-Joachim Wunderlich[†]
†ITI, University of Stuttgart, Germany
‡Kyushu Institute of Technology, Iizuka, Japan
§Dep. Electrical & Computer Engineering, University of Massachusetts at Amherst, USA
kochte@iti.uni-stuttgart.de, kundu@ecs.umass.edu, k_miyase@cse.kyutech.ac.jp, wen@cse.kyutech.ac.jp, wu@informatik.uni-stuttgart.de

***Abstract*—Unknown (X) values, originating from memories, clock domain boundaries or A/D interfaces, may compromise test signatures and fault coverage. Classical logic and fault simulation algorithms are pessimistic w.r.t. the propagation of X values in the circuit. This work proposes efficient hybrid logic and stuck-at fault simulation algorithms which combine heuristics and local BDDs to increase simulation accuracy. Experimental results on benchmark and large industrial circuits show significantly increased fault coverage and low runtime. The achieved simulation precision is quantified for the first time.**

***Index Terms*—Unknown values, X propagation, precise fault simulation, symbolic simulation, BDD**

I. Introduction

During test application unknown signal values in a circuit, termed X values, may corrupt the test signature and result in a loss of fault coverage, especially in test compression and BIST architectures [1]. X values may originate from uncontrolled flip-flops, memories, clock domain boundaries or tristate logic. They can be controlled by insertion of additional design-for-test circuitry to block their propagation within the circuit. However, blocking logic increases overhead and inherently limits observability of parts of the design, reducing test quality. Typically, a fraction of X sources cannot be blocked and the resulting X values are handled for example by X-tolerant compactors [2, 3] or X masking logic [4, 5] in test compression architectures. With increasing number of Xs to be handled, hardware overhead and test time increases as more patterns have to be applied to reach the targeted fault coverage.

To estimate the propagation of X values in the circuit, n-valued logic simulation algorithms with a fixed number of logic symbols, for instance the 3-valued logic with $\{0, 1, X\}$, are used. However, these classical algorithms overestimate the number of Xs in the circuit since they are inherently pessimistic w.r.t. X propagation. These algorithms cannot uniquely identify different X values in the circuit and thus, fail to correctly evaluate reconvergences of Xs. Figure 1 (a) gives an example of a simple circuit and input stimuli with one X source. 3-valued simulation computes the value of the output signal as X, while its actual value is 0 because of X canceling at the reconvergence at the AND gate.

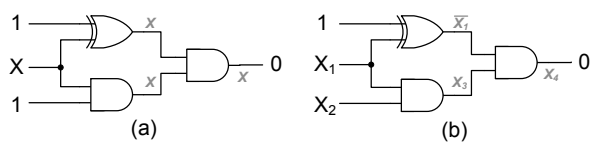

Fig. 1: Pessimism in (a) 3-valued and (b) Restricted symbolic simulation

Restricted symbolic simulation [6, 7] uses a higher number of symbols to encode the states of the signals. It is able to track simple inversions of X sources and allows to evaluate some reconvergences correctly. While restricted symbolic simulation correctly evaluates the state of the output signal in figure 1 (a), it fails in the case (b) since it is not able to encode the conjunction of the two unique X states X_1 and X_2.

Apart from logic simulation, this pessimism in X propagation affects a wide range of other EDA algorithms as well. Fault simulation as well as ATPG suffer from the same shortcoming because the underlying logic reasoning is unable to track unique X states. The result in fault simulation is an underestimation of fault coverage, resulting in increased unnecessary test costs and reduced product quality.

By applying SAT-based formal methods, a precise analysis of X reconvergences can be conducted [8]. This method can also be applied to fault simulation [9]. However, the computational cost is high if a large number of test patterns is evaluated. A precise symbolic simulation which represents all possible signal states on X valued signals can also be implemented using binary decision diagrams (BDDs, [10]). Yet for larger circuits, especially with arithmetic structures like multipliers, building the corresponding BDD is not trivial. This problem has been solved in verification and equivalence checking applications by partitioning the circuit and building partial BDDs using cutpoints [11].

The contribution of this work are novel logic and fault simulation algorithms, which leverage heuristic techniques and partial BDD based symbolic simulation to significantly increase the accuracy of logic and fault simulation in presence of Xs with low computational effort. The result is a more precise computation of fault coverage and increased test and product quality at no cost.

The next section introduces the used terminology, followed by a discussion of related work in section III. The proposed algorithm is presented in section IV and evaluated on benchmark and industrial circuits in section V.

II. Terminology

Due to the inherent pessimism, n-valued logic simulation algorithms compute a superset of the signals which actually have an unknown value. In this work we distinguish the following three types of X values:

Pessimistic X (PEX): The value of a signal is called PEX if and only if 3-valued logic simulation determined that the signal value is an X for a given input stimulus. Note that this computation is pessimistic.

Real X (REX): The value of a signal is called REX if and only if its value depends on at least one of the X sources of the circuit for a given input stimulus. The set of REX signals is a subset of the set of PEX signals. By definition, all X sources are REX.

False X (FEX): The value of a signal is called FEX if and only if its value can be proven to be independent of any of the X sources in the circuit for a given input stimulus. While all signals with binary value $\in \{0, 1\}$ could be considered FEX, we constrain the set of FEX signals to a subset of the PEX signals. FEXs originate at reconvergences of signals with REX value if X canceling occurs.

From these definitions, it follows that the set of signals with PEX values is the union of the sets of signals with REX and FEX values: $PEX = REX \cup FEX$.

III. Related Work

The pessimism of logic simulation w.r.t. X propagation can be reduced by approximate as well as precise analysis. Approximate techniques comprise static and structural analysis as well as restricted symbolic simulation. Precise techniques are based on formal methods, typically mapping the problem to decision diagrams or satisfiability instances.

In [12], the authors showed how knowledge gained from a structural static learning analysis [13] can be used for a limited increase of the accuracy of logic and fault simulation. The simulation approach based on circuit partitioning [14] augments n-valued logic simulation by extracting small X valued regions of the circuit with X reconvergences for separate simulation to learn about local X propagation. The result is fed back into the simulation of the whole circuit.

In restricted symbolic simulation [6], also called partial symbolic simulation [15] or distinguishing X simulation [16], the number of symbols to track unique X states is significantly increased. At low computational cost, simple inversions of X states and their reconvergences can be accurately computed. Depending on the circuit and input stimuli, restricted symbolic simulation allows to significantly reduce the pessimism in logic simulation. However, more complex operations of X states (e.g. conjunction of two unique X states) are still evaluated pessimistically. In [7], restricted symbolic simulation is applied to the implication step of ATPG.

A formal analysis implemented as symbolic simulation of the circuit is able to represent *all* possible X states of signals, depending on the X sources. If the implementation is based on reduced ordered BDDs [17] and each signal is assigned a BDD describing its Boolean function w.r.t. the input stimulus, then any non-constant function denotes a REX state. All other signals have constant binary values for the given stimulus, independent of the X sources. By interpreting X sources as defect locations with unknown behavior, this symbolic simulation algorithm can be applied to fault diagnosis [18] as well. Still, building BDDs for larger circuits, or even parts of larger circuits, may prove difficult especially if certain arithmetic structures, e.g. multipliers, are contained.

BDD-based techniques have also been applied to sequential fault simulation with unknown or don't care values to reduce the explosion of X states in the circuit over multiple test cycles [19]. To allow the simulation of larger circuits, [20] proposed a method that switches between BDD-based and pessimistic simulation in different simulation cycles, depending on memory requirements.

More recent approaches map the problem of precise logic simulation to the satisfiability (SAT) domain. The approach in [21] maps an RTL design to a quantified Boolean formula (QBF) to reason about X propagation stemming from uninitialized registers over a limited number of simulation cycles. In [9], the 3-valued simulation is extended by a precise SAT-based analysis of reconvergences of REX values. The result is an accurate computation of REX values for the given stimulus at the cost of increased runtime (in the order of ATPG for the same circuit). This technique is also used to increase the accuracy of stuck-at fault simulation.

The problem of accurately handling unknown values is also known in verification and equivalence checking. Here, X values mainly stem from black boxes in the design. [22] suggests to explicitly encode X values in the SAT instance allowing to represent one single X state. Yet the approach is not able to track multiple unique X states originating from different X sources or Boolean operations on signals with X states. [23, 24] use BDDs to conduct a symbolic simulation of the design for verification with X states. [23] switches to a SAT-based reasoning when the memory requirement of the BDDs exceeds the available memory and trades off memory and runtime. For logic and fault simulation, thousands of patterns need to be simulated and runtime must be as small as possible.

In this work, we propose to increase the accuracy of both logic and fault simulation by combining heuristics and efficient fault simulation techniques with symbolic simulation of limited subpartitions of the circuit by use of BDDs. Instead of mapping the whole circuit to a BDD to perform the symbolic simulation, only local parts of the circuit are represented by BDDs. A similar use of partial BDDs is known from equivalence checking [11]. The following sections presents the proposed algorithms for logic and fault simulation.

IV. Enhanced Fault Simulation using Local BDDs

The proposed stuck-at fault simulation algorithm partitions the circuit into fanout free regions (FFRs) and separately computes fault activation in the FFR as well as the observability of the corresponding fanout stem [25–27]. For the logic simulation of the fault-free circuit and the observability computation of fanout stems, a hybrid logic simulation algorithm is used. This algorithm combines local BDDs bounded by a node limit to evaluate X-reconvergences, and restricted symbolic simulation to reduce the size of BDDs and the number of BDD operations as much as possible. The algorithm is explained in detail in the next two subsections, followed by the discussion of the fault simulation algorithm in section IV-C.

A. X-Reconvergence Analysis Using Local BDDs

Reduced ordered BDDs allow the symbolic simulation of the Boolean function of a circuit. To correctly classify PEX values into REX and FEX values for a given input stimulus and a set of X sources, the function of each PEX signal is represented as a shared BDD. Due to canonicity of reduced ordered BDDs, a simple comparison with the tautology function (constant 1) or its inverse reveals whether the signal under consideration carries a FEX value or not. If the resulting BDD is not equal to a constant function, the signal value depends on the X sources and thus, is a REX value.

Listing 1: Pseudo code of hybrid logic simulation

```
// S: set of signals in the circuit
// val_s: value of signal s determined by pessimistic simulation

S_PEX := {s ∈ S | val_s ∉ {0,1}}
for s ∈ S_PEX in topological order, starting from X sources
    g := driving_gate(s)
    if (∃s' ∈ inputs(g) : val_s' = controlling_value(g)) or
       (∀s' ∈ inputs(g) : val_s' ∈ {0,1}) then
        val_s := table_lookup(g)
    else
        if (s is an X source) then
            bdd_s := create_bdd_variable(s)
        else
            bdd_s := build_bdd(g)
            if (bdd_s = 0) then
                val_s := 0
            else if (bdd_s = 1) then
                val_s := 1
            else if (size(bdd_s) > limit) then
                delete_bdd(bdd_s)
                bdd_s := create_bdd_variable(s)
            end
        end
    end
end
```

The actual comparison of the constructed BDD of a signal is a constant time operation. However, constructing the BDD for larger structures and certain sensitive logic functions may require an exponential amount of memory in the number of inputs. Assuming that X canceling occurs locally, we build local BDDs limited in size by partitioning the circuit at cutpoints [11, 23] to overcome excessive memory requirements.

The BDD-based hybrid logic simulation algorithm is depicted in listing 1. For a given input stimulus, the set of signals S_{PEX} with a PEX value is computed, for example by 3-valued logic simulation. As a by-product, the binary values of all other signals are known.

The PEX signals are then processed in topological order, starting from the X sources in the circuit. For each X source, a BDD variable is introduced For all other signals s to be processed, the Boolean function of the preceding signals have already been computed due to the topological processing. If one of the predecessors of s has the controlling value of the driving gate g, or all the predecessors have binary values, then the value val_s of s can be derived by a fast table lookup. If val_s cannot be derived by simple reasoning, a BDD is constructed for the Boolean function at s. The method build_bdd in the pseudo-code constructs the BDD for s depending on the driving gate g and the Boolean functions of its inputs.

If the resulting BDD is a constant function, the signal state is set to FEX with either 0 or 1 as value val_s. This value is then used for the simulation of the fanout of s. Otherwise, the signal state is kept as PEX. If the size of the BDD at signal s exceeds a given limit w.r.t. the number of nodes, we stop describing the function at s as BDD and introduce a new free variable which is used in the construction of the following BDDs in the output cone of s. The BDD with excessive size is not used any more and deleted. Since relations between the BDD variables are not analyzed, the variable order in the BDD is given by the topological graph traversal.

An example of the constructed BDDs for a small circuit is given in figure 2. Assume signal lines b, c, d are three X sources, and a, b have the value 1. In figure 2b, the non-trivial BDDs of the Boolean functions at signals f, g, h, i, k are depicted. bdd_f represents the inversion of signal b and bdd_g represents the conjunction of signals b and c. bdd_h collapses to the constant-0 function. bdd_i^* results from the exclusive disjunction of the functions at d and g. Assuming a node limit of 5, this BDD exceeds the limit and a new variable is introduced (bdd_i). bdd_k finally reflects the inversion of bdd_i due to the negated conjunction at k.

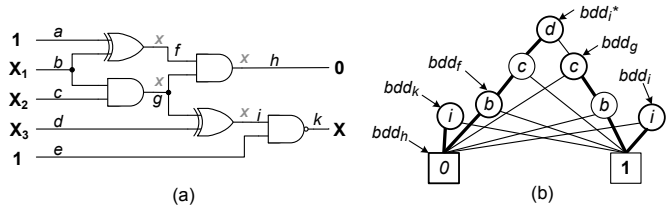

Fig. 2: BDD construction in the hybrid logic simulation flow

Using local BDDs with limited size, high memory requirements are avoided at the cost of a certain degree of simulation accuracy. X-reconvergences might be evaluated pessimistically if a correlation between signal states is lost due to the introduction of new variables when a BDD exceeds the node limit.

B. Hybrid Logic Simulation with Increased Accuracy

To reduce the computational effort of the algorithm in the previous section, we employ a restricted symbolic simulation [6] instead of 3-valued simulation to reduce the number of PEX signals for which BDDs are constructed. In the restricted symbolic simulation two symbols are used to encode the binary states $\{0, 1\}$. In addition, each X source is assigned a unique symbol encoded as positive integer larger than 1. Inverted X states are assigned the negated integer value of the original X state. This encoding allows to correctly track propagation of Xs stemming from X sources unless they are combined with different X sources. In particular, the X state does not loose its identity by propagation through inverting gates. If two or more X states from different X sources are combined at a gate and X canceling does not occur, the output value is assigned a new symbol which has not been used in the simulation so far. This symbol is then used for the simulation in the output cone of that signal. By using a 32-bit machine word per symbol, about 2.1 billion different symbols are available.

Table I shows the truth table of a two-input AND gate for restricted symbolic simulation. For combinations of X values (a_p, $-a_p$) which result in X canceling, a binary value is computed. If uncorrelated X values are combined (a_p, a_q), a new unique X symbol is generated (denoted as X').

A large fraction of simple X reconvergences are evaluated correctly with restricted symbolic simulation. Still, X reconvergences may be evaluated pessimistically if X canceling occurs after combination of multiple different X states. Only for these cases, local BDDs are constructed in the algorithm of the preceding section. Reducing the number of PEX signals for which a BDD analysis is required, significantly improves the efficiency of the overall hybrid logic simulation.

TABLE I: Two-input AND gate truth table (adopted from [7])

$\wedge$	0	1	a_p	$\text{-}a_p$	a_q	$\text{-}a_q$
0	0	0	0	0	0	0
1	0	1	a_p	$\text{-}a_p$	a_q	$\text{-}a_q$
a_p	0	a_p	a_p	0	X'	X'
$\text{-}a_p$	0	$\text{-}a_p$	0	$\text{-}a_p$	X'	X'
a_q	0	a_q	X'	X'	a_q	0
$\text{-}a_q$	0	$\text{-}a_q$	X'	X'	0	$\text{-}a_q$

C. Efficient Fault Simulation with Increased Accuracy

The enhanced stuck-at fault simulation is based on the hybrid logic simulation algorithm. For a given test pattern, this algorithm accurately classifies a higher number of faults as detected compared to classical algorithms based on a 3-valued logic like the concurrent algorithm [28] or the PPSFP algorithm [25–27]. As in state-of-the-art PPSFP algorithms, computation of fault activation and fault propagation to an observable output or pseudo output is done separately to increase efficiency. However, patterns are not evaluated in a bit-parallel manner since the hybrid logic simulation algorithm processes a single pattern at a time. The overall flow of the proposed fault simulation algorithm is depicted in figure 3.

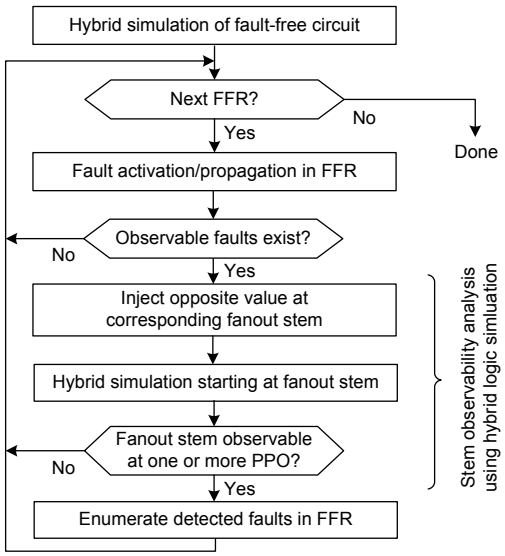

Fig. 3: Enhanced fault simulation flow

For each pattern, fault simulation starts with hybrid logic simulation of the fault-free circuit. The FFRs of the circuit are then processed one at a time. The algorithm determines fault activation and fault propagation within the FFR using the computed high precision signal values of the fault-free circuit. Within the FFR, the fault effect propagates only along a single path towards the fanout stem. However, this path may be part of a branch of an X valued fanout and reconvergence. The resulting value of such a reconvergence might be impacted by the faulty value. In this step, such rare cases of reconvergences are evaluated pessimistically.

If there are any yet undetected faults in the FFR which are activated by the test pattern and observable at the corresponding fanout stem, the algorithm proceeds with the stem observability analysis. In contrast to [9], where the stem observability analysis is conducted using a pessimistic 3-valued logic simulation, the hybrid logic simulation algorithm increases the accuracy of this step as well. The simulation starts at the fanout with the opposite value of the fault-free circuit, and stops at observable outputs. The resulting logic values are compared with the values of the fault-free simulation. If the binary values differ, the fanout is observable and therefore, the activated and locally propagated faults in the FFR are detectable with the particular pattern.

V. Evaluation and Results

This section presents experimental results for benchmark and industrial circuits. The achieved accuracy of logic and stuck-at fault simulation is discussed. The algorithm is implemented in Java except for the BDD part. The BDD package BuDDy [29] version 2.4 is used to construct and perform any operations on BDDs. The results are verified by exhaustive classical logic and fault simulation for smaller circuits. All experiments are conducted on a 3.0GHz Intel Core i7 workstation.

A. Simulation Precision in Presence of Unknown Values

Using the ISCAS'85 circuits, the achievable precision of logic and fault simulation is assessed. In the experiments, 64 random patterns are evaluated. Per pattern, a fixed number of circuit inputs (4, 8 and 16 inputs) are randomly selected as X sources (the implementation also allows to configure internal signals as X source).

To compute the precision, an exhaustive simulation of the patterns representing all possible values of the X sources is conducted for each target pattern. In the case of 4 X sources, 16 fully specified patterns are generated per target pattern. For logic simulation, we define precision as the ratio between the number of found FEX signals over 3-valued simulation and the number of FEX signals determined by exhaustive simulation. For fault simulation, precision is the ratio of additionally detected faults over a PPSFP fault simulation by the proposed algorithm, and the number of additionally detected faults computed by exhaustive fault simulation. For the exhaustive fault simulation, the fully specified patterns generated per target pattern are simulated using the PPSFP algorithm and the intersection of the detected faults is computed.

Table II lists the results for the circuits and different numbers of X sources. For each case, column 3 and 4 give the average and maximum ratio of FEX signals w.r.t. the PEX signals computed by 3-valued simulation. Up to 72% of the PEX signals found by 3-valued simulation can be proven to actually have a binary value (c2670).

Column 5 to 9 list the precision of the proposed hybrid logic simulation depending on the BDD node limit. Column 5 (RSS) gives the result for the restricted symbolic simulation without using any BDDs. With increasing BDD size the precision increases as well. For most circuits, small BDDs with at most 50 nodes already achieve 100% precision.

The results for fault simulation are listed in column 10 to 14. The precision is given for restricted symbolic simulation as well as for different BDD limits. In general, precision grows with increasing BDD limit and reaches very high values already for a BDD limit of 50 nodes. A higher BDD limit of 1000 nodes improves the precision only in few cases.

B. Experimental Results for Industrial Circuits

In a second series of experiments, larger ITC'99 benchmark and industrial circuits (kindly provided by NXP) have been used in a commercial design-for-test and test generation flow.

TABLE II: Precision of hybrid logic and enhanced fault simulation

Circuit	Num. X sources	Avg. rat. FEX sig. [%]	Max. rat. FEX sig. [%]	Logic sim. prec. for BDD node limit [%] RSS	5	15	50	1000	Fault sim. prec. for BDD node limit [%] RSS	5	15	50	1000
c1355	4	33.3	48.3	22.7	58.1	100.0	100.0	100.0	0.0	45.7	98.9	98.9	98.9
	8	21.3	42.9	32.4	33.0	53.0	98.9	100.0	0.0	0.0	31.6	98.1	99.9
	16	5.1	13.2	97.8	97.8	97.8	99.7	100.0	0.0	0.0	0.0	100.0	100.0
c1908	4	5.7	37.2	13.3	86.0	100.0	100.0	100.0	10.5	81.5	91.6	91.6	91.6
	8	4.5	32.2	0.5	32.3	79.9	100.0	100.0	4.9	41.7	81.1	94.8	94.8
	16	2.2	23.2	0.1	38.5	40.5	82.5	100.0	3.4	42.9	46.3	88.0	98.3
c2670	4	20.5	71.9	98.1	100.0	100.0	100.0	100.0	97.1	98.3	98.3	98.3	98.3
	8	12.5	50.1	93.5	100.0	100.0	100.0	100.0	95.3	99.0	99.0	99.0	99.0
	16	15.4	62.4	86.5	100.0	100.0	100.0	100.0	73.8	98. 4	98.4	98.4	98.4
c3540	4	28.5	71.0	87.0	100.0	100.0	100.0	100.0	88.6	96.1	96.1	96.1	96.1
	8	17.9	48.2	71.8	96.8	100.0	100.0	100.0	61.1	85.4	89.1	89.1	89.1
	16	8.7	24.8	44.3	85.5	96.7	100.0	100.0	24.8	66.3	84.5	89.2	89.2
c5315	4	27.7	51.5	99.4	100.0	100.0	100.0	100.0	98.6	98.6	98.6	98.6	98.6
	8	23.4	55.5	95.1	100.0	100.0	100.0	100.0	95.2	98.0	98.0	98.0	98.0
	16	19.7	34.4	88.5	99.9	100.0	100.0	100.0	88.6	98.6	98.6	98.6	98.6
c6288	4	38.4	53.3	86.8	98.6	100.0	100.0	100.0	60.3	91.5	99.4	99.4	99.4
	8	23.7	36.9	88.9	97.0	99.6	100.0	100.0	51.1	74.4	93.0	98.4	99.1
	16	11.8	21.1	92.0	96.9	99.4	99.7	99.9	39.0	46.0	65.5	66.2	93.0
c7552	4	22.2	44.8	97.6	100.0	100.0	100.0	100.0	97.0	97.7	97.7	97.7	97.7
	8	20.8	42.4	92.8	100.0	100.0	100.0	100.0	94.1	98.0	98.0	98.0	98.0
	16	19.7	38.4	90.5	100.0	100.0	100.0	100.0	93.3	97.9	97.9	97.9	97.9

For each circuit, a fixed randomized set of flip-flops is assumed to be unscannable and the respective pseudo primary inputs are replaced by X sources. All other flip-flops are configured in scan chains. 16 of these X source configurations are generated per circuit and the one with average behavior is used in the following. For this configuration, the random pattern resistant stuck-at faults in the output cones of the X sources are computed and test patterns are generated with a commercial, X aware ATPG. For the generated test patterns, a logic simulation using the proposed hybrid algorithm, as well as a fault simulation with the proposed algorithm is conducted. The resulting number of computed FEX signals and additionally detected faults are shown in table III. Due to the circuit size (up to 1 million gates), the exhaustive simulation used in the previous section could not be applied.

Column 2 and 3 of the table show the size of the circuit in gate primitives as well as the percentage of flip-flops (pseudo primary inputs) selected as X sources. The next column lists the number of computed FEX signals w.r.t. 3-valued logic simulation for the restricted symbolic simulation (RSS). The following two columns give the absolute increase in the number of FEX signals w.r.t. the result of restricted symbolic simulation. Very small BDDs are already sufficient to significantly improve the result of restricted symbolic simulation.

The last three columns relate to fault simulation. The number of additionally detected faults using restricted symbolic simulation is found in column 7 (RSS). Column 8 and 9 give the increase of detected faults over restricted symbolic simulation for BDD node limits of 5 and 50. RSS already classifies a large number of faults as detected compared to classical algorithms. Using small local BDDs, several thousands of faults are detected in addition to RSS. Note that for most of these faults ATPG was not able to generate patterns. For highest product quality and 0-DPM requirements, fault simulation accuracy must be as high as possible.

The required runtime for the logic and fault simulation depends on the circuit size, the particular X configuration and the number of patterns. Due to limited space, runtimes (in seconds per pattern) are only given for larger circuits and the 5% X configuration in table IV. Even for largest design with about one million gates, the fault simulation time per pattern is less than 2 seconds. Runtime increases only slightly with the higher BDD limits. Compared to the SAT-based method of [9], fault simulation runtime is reduced by a factor of 43X for p418k. Memory consumption for the largest circuit reaches 3210 MByte.

VI. CONCLUSIONS

This paper presented novel hybrid logic and fault simulation algorithms based on restricted symbolic simulation and local BDDs. The efficient algorithms reduce the pessimism of X propagation during simulation and thus achieves more precise results in presence of unknown values. The enhanced fault simulation with increased accuracy is able to correctly classify a very high number of faults as detected compared with classical fault simulation algorithms. This increase in fault coverage and product quality comes at no cost in terms of test time, test data or hardware overhead. The experimental results show the applicability to 1 million gate industrial circuits. Already with small local BDDs and very low runtimes, the achievable precision is very high.

ACKNOWLEDGMENT

This work was partly supported by the German Research Foundation (DFG) as part of the priority program "SPP 1500 Dependable Embedded Systems". M. Kochte was supported by the German Academic Exchange Service (DAAD) as a Visiting Researcher at the Kyushu Institute of Technology.

REFERENCES

[1] G. Hetherington, T. Fryars *et al.*, "Logic BIST for large industrial designs: Real issues and case studies," in *Proc. ITC*, 1999, pp. 358–367.

[2] S. Mitra and K. S. Kim, "X-compact: an efficient response compaction technique," *IEEE Trans. CAD*, vol. 23, no. 3, pp. 421–432, March 2004.

[3] M. Sharma and W.-T. Cheng, "X-filter: Filtering unknowns from compacted test responses," in *Proc. IEEE ITC*, 2005, paper 42.1.

[4] M. Naruse, I. Pomeranz *et al.*, "On-chip compression of output responses with unknown values using LFSR reseeding," in *Proc. ITC*, 2003, pp. 1060–1068.

[5] J. Rajski, J. Tyszer *et al.*, "X-press: Two-stage x-tolerant compactor with programmable selector," *IEEE Trans. CAD*, vol. 27, no. 1, pp. 147–159, Jan. 2008.

[6] J. Carter, B. Rosen *et al.*, "Restricted symbolic evaluation is fast and useful," in *Proc. ICCAD*, 1989, pp. 38–41.

TABLE III: Results of proposed logic and fault simulation for ITC'99 benchmark (b17, b18, b19) and industrial circuits

Circuit	Num. gate primitives	X input ratio	Num. FEX for BDD node limit			Num. add. faults for BDD node limit		
			RSS	5 (Δ to RSS)	50 (Δ to RSS)	RSS	5 (Δ to RSS)	50 (Δ to RSS)
b17	38513	1 %	24167	82	82	0	0	0
		2 %	55246	4292	4292	6	0	0
		5 %	266100	45146	45182	391	50	50
b18	131673	1 %	459523	67	67	500	0	0
		2 %	783480	191796	192780	703	851	851
		5 %	1263002	998238	1043837	1482	240	240
b19	265071	1 %	4194551	599503	599503	6753	1402	1402
		2 %	6016033	1162300	1219382	2055	2235	2415
		5 %	5319376	1860802	1967974	2654	2039	2046
p286k	368981	1 %	388645	1408	1408	565	0	0
		2 %	634547	34834	34834	1287	0	0
		5 %	1432102	70962	71121	3638	70	70
p330k	348152	1 %	3951633	389716	389716	1860	99	99
		2 %	5685365	480231	480231	4169	299	299
		5 %	2611189	319328	321115	7953	2680	2680
p378k	374467	1 %	332450	18517	18517	10151	604	604
		2 %	1068676	112618	112856	19864	1971	1992
		5 %	2821627	454346	466423	41772	10922	11178
p388k	482497	1 %	535026	73770	73770	2324	414	414
		2 %	1150819	37218	37218	6364	49	49
		5 %	4316667	1038516	1107619	11372	5284	5543
p418k	442888	1 %	1620174	2188	2188	4267	0	0
		2 %	473569	16349	16349	4691	52	52
		5 %	2090221	559747	650040	7057	422	447
p533k	653015	1 %	2284341	153558	153558	4987	82	82
		2 %	8044449	950278	951414	12562	2960	2960
		5 %	41481309	9965507	10242439	60373	24250	24919
p874k	802410	1 %	1968587	40095	40095	5210	75	75
		2 %	3021585	262992	262992	5127	397	397
		5 %	4310243	894282	903653	10372	2188	2190
p951k	1012970	1 %	1247156	120666	120666	4079	203	203
		2 %	4574620	342388	342388	5788	424	424
		5 %	9250782	1943976	1996095	15084	3405	3405

TABLE IV: Runtime per pattern (in sec.)

Circuit	Logic sim.			Fault sim.		
	RSS	5	50	RSS	5	50
b19	0.39	0.84	0.88	1.21	1.22	1.23
p418k	0.77	1.29	1.30	1.14	1.26	1.22
p533k	0.95	1.41	1.41	1.75	2.12	1.94
p874k	1.17	1.62	1.61	1.72	1.78	1.80
p951k	1.43	1.69	1.74	1.82	1.97	2.01

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

[8] M. Elm, M. A. Kochte, and H.-J. Wunderlich, "On determining the real output Xs by SAT-based reasoning," in *Proc. ATS*, 2010, pp. 39–44.

[9] M. A. Kochte and H.-J. Wunderlich, "SAT-based fault coverage evaluation in the presence of unknown values," in *Proc. DATE*, 2011.

[10] R. E. Bryant, "Symbolic simulation - techniques and applications," in *Proc. ACM/IEEE Design Automation Conference*, 1990, pp. 517–521.

[11] A. Kuehlmann and F. Krohm, "Equivalence checking using cuts and heaps," in *Proc. Design Automation Conference*, 1997, pp. 263–268.

[12] 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.

[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] 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.

[15] X. Wen, T. Miyoshi *et al.*, "On per-test fault diagnosis using the X-fault model," in *Proc. ICCAD*, 2004, pp. 633–640.

[16] N. Sridhar and M. Hsiao, "On efficient error diagnosis of digital circuits," in *Proc. ITC*, 2001, pp. 678–687.

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

[18] S.-Y. Huang, "Speeding up the byzantine fault diagnosis using symbolic simulation," in *Proc. IEEE VTS*, 2002, pp. 193–198.

[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] 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.

[22] A. Jain, V. Boppana *et al.*, "Testing, verification, and diagnosis in the presence of unknowns," in *Proc. VTS*, 2000, pp. 263–268.

[23] C. Wilson, D. Dill, and R. Bryant, "Symbolic simulation with approximate values," in *Formal Methods in Computer-Aided Design*, ser. Springer LNCS, 2000, vol. 1954, pp. 507–522.

[24] C. Scholl and B. Becker, "Checking equivalence for partial implementations," in *Proc. Design Automation Conference*, 2001, pp. 238–243.

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

[26] 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.

[27] 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.

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

[29] J. Lind-Nielsen, "BuDDy - A binary decision diagram package," Technical University of Denmark, 1997.