Mixed 01X-RSL-Encoding for Fast and Accurate ATPG with Unknowns

Erb, Dominik; Scheibler, Karsten; Kochte, Michael A.; Sauer, Matthias; Wunderlich, Hans-Joachim; Becker, Bernd

Proceedings of the 21st Asia and South Pacific Design Automation Conference (ASP-DAC'16) Macao SAR, China, 25-28 January 2016

doi: http://dx.doi.org/10.1109/ASPDAC.2016.7428101

Abstract: Unknown (X) values in a design introduce pessimism in conventional test generation algorithms which results in a loss of fault coverage. This pessimism is reduced by a more accurate modeling and analysis. Unfortunately, accurate analysis techniques highly increase runtime and limit scalability. One promising technique to prevent high runtimes while still providing high accuracy is the use of restricted symbolic logic (RSL). However, also pure RSL-based algorithms reach their limits as soon as millon gate circuits need to be processed. In this paper, we propose new ATPG techniques to overcome such limitations. An efficient hybrid encoding combines the accuracy of RSL-based modeling with the compactness of conventional threevalued encoding. A low-cost two-valued SAT-based untestability check is able to classify most untestable faults with low runtime. An incremental and event-based accurate fault simulator is introduced to reduce fault simulation effort. The experiments demonstrate the effectiveness of the proposed techniques. Over 97% of the faults are accurately classified. Both the number of aborts and the total runtime are significantly reduced compared to the state-of-the-art pure RSL-based algorithm. For circuits up to a million gates, the fault coverage could be increased considerably compared to a state-of-the-art commercial tool with very competitive runtimes.

Preprint

Mixed 01X-RSL-Encoding for Fast and Accurate ATPG with Unknowns

Dominik Erb*, Karsten Scheibler*, Michael A. Kochte‡, Matthias Sauer*, Hans-Joachim Wunderlich‡, Bernd Becker*
*University of Freiburg, Georges-Köhler-Allee 51, 79110 Freiburg, Germany
‡University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Germany

***Abstract*—Unknown (X) values in a design introduce pessimism in conventional test generation algorithms, which results in a loss of fault coverage. This pessimism is reduced by a more accurate modeling and analysis. Unfortunately, accurate analysis techniques highly increase runtime and limit scalability. One promising technique to prevent high runtimes while still providing high accuracy is the use of restricted symbolic logic (RSL). However, also pure RSL-based algorithms reach their limits as soon as millon gate circuits need to be processed.**

In this paper, we propose new ATPG techniques to overcome such limitations. An efficient hybrid encoding combines the accuracy of RSL-based modeling with the compactness of conventional three-valued encoding. A low-cost two-valued SAT-based untestability check is able to classify most untestable faults with low runtime. An incremental and event-based accurate fault simulator is introduced to reduce fault simulation effort. The experiments demonstrate the effectiveness of the proposed techniques. On average, over 99.3% of the considered faults are accurately classified. Both the number of aborts and the total runtime are significantly reduced compared to the state-of-the-art pure RSL-based algorithm. For circuits up to a million gates, the fault coverage could be increased considerably compared to a state-of-the-art commercial tool with very competitive runtimes.

***Index Terms*—Unknown values, test generation, ATPG, Restricted symbolic logic, SAT, Stuck-at fault**

I. Introduction

During test, unknown or X-values can occur in a design at uninitialized memory blocks or non-scan flip-flops, at clock-domain boundaries, or at analog-digital converters. Since these X-values reduce the observability and controllability of signals, the fault coverage is reduced. X-blocking design-for-test structures can be added to limit the propagation of X-values in the design. However, the resulting performance, area, and wiring overhead may prohibit to block all sources of X-values.

Automatic test pattern generation (ATPG) algorithms have to consider these remaining X-values during test generation and fault simulation. Conventional ATPG algorithms for stuck-at faults are based on logic algebras using a limited, fixed number of values to represent signal states in the fault-free and faulty circuits. Examples are the five-valued or nine-valued logic [1, 2]. Typically, one symbol X is used to denote the state of signals with an X-value. The nine-valued logic is also often adapted to SAT-based test generation for stuck-at faults [3]. In these logics, it is not possible to *distinguish* different X-values and to correctly evaluate reconvergences of X-valued signals.

Figure 1 shows a circuit with two X-sources (signals b and d). The X-value originating at signal b fans out and reconverges at output j. Since the negation of the X-value at gate G_2 cannot be captured by a three-valued logic with values $\{0, 1, X\}$, the value of output j is evaluated as X. The stuck-at-0 fault at j is untestable when using three-valued logic since j cannot be justified to value 1. An accurate analysis, however, shows that j has value 1 when $(a, c, e) = (1, 0, 1)$ independent of the value of signal b.

This pessimistic evaluation is a principal shortcoming of logic algebras with a limited, fixed set of values. The accurate computation of signal values in presence of multiple X-sources and reconvergenes is an NP-complete problem and can be performed

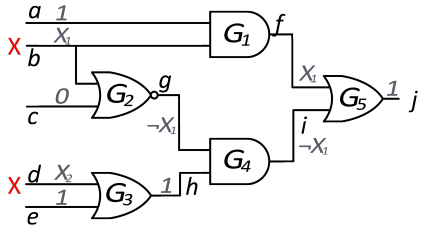

Fig. 1: Stuck-at-0 fault at output j is undetectable in three-valued logic, but detectable using restricted symbolic logic.

by symbolic simulation or by mapping the evaluation to Boolean satisfiability (SAT) [4, 5]. Accurate fault simulation can also be mapped to SAT [6, 7].

The accurate (non-pessimistic) test pattern generation in presence of X-values is an NP-hard problem, which was solved by a mapping to the satisfiability of quantified Boolean formulas (QBF) [8]. However, deciding the satisfiability of a QBF is a PSPACE-complete problem, which is more complex than combinational ATPG or SAT [9]. For larger circuits the time to generate a pattern for a fault or to prove its untestability grows quickly.

In [10], a test pattern generation algorithm based on *restricted symbolic logic* (RSL) achieves lower runtimes than QBF-based ATPG in presence of X-values with minimal impact on the fault coverage: In RSL, different symbols are used to represent X-values and their negation [11, 12]. Logic simulation using RSL (also known as numbered-X or indexed simulation) allows to accurately evaluate local reconvergences of X-valued signals. But RSL may be pessimistic when X-values from *different* X-sources converge at a gate.

In Fig. 1, output j is correctly evaluated to value 1 when logic simulation is based on RSL, since the reconvergence of X-values stemming from signal b at gate G_5 and the resulting X-canceling (here: $X_1 \vee \neg X_1 = 1$) are evaluated accurately. Using RSL, a pattern for the stuck-at-1 fault at j can also be generated.

However, the encoding of the possible X-values at signals in RSL is costly and increases the size of the resulting SAT instance. Furthermore, since RSL-based algorithms are in general unable to prove fault untestability, a dedicated untestability check is required, as proposed in [13] where a mapping to SMT (SAT Modulo Theory) is done.

In this work, ATPG techniques are proposed which overcome the limitations of a pure RSL encoding:

- An *efficient hybrid value encoding* is introduced that combines the accuracy of RSL-based encoding and conventional three-valued encoding. It is based on a thorough analysis of the circuit to limit RSL encoding to the cases when actual reconvergences may cause X-canceling, and uses a combination of pessimistic and RSL encoding. This reduces the size of the SAT instance and the resulting runtime.
- A *low-cost two-valued SAT-based untestability check* is proposed to quickly drop untestable faults.
- *Incremental* accurate fault simulation reduces the simulation

effort significantly by combining event-based accurate evaluation of gates during the fault-free simulation and clever selection of faults to be accurately evaluated by propagation path analysis.

These techniques result in a speedup of the ATPG of up to $88\times$. The number of faults not classified within a timeout of 10 seconds decreases by over 99%, and circuits with more than a million gates can be handled for the first time.

The next section introduces the used terminology. Section III gives a short overview of the prosed ATPG framework. Sections IV, V, and VI present the novel ATPG techniques in detail. Experimental results are discussed in Section VII.

II. Terminology

A. X-Values

An X-value always represents a well defined, but unknown *binary* value of either logic-0 or logic-1. This excludes *undefined* values caused by undefined voltage levels for example by driver contention and corresponds to the semantics of unknown values in Kleene's strong three-valued logic [14]. Signals that generate unknown values are called X-sources – e. g. uncontrolled sequential elements or clock-domain crossings. A signal or gate in the circuit is X-dependent if and only if it is reachable from an X-source.

B. Two-, Three-Valued and Restricted Symbolic Logic (RSL)

Two-valued logic only allows to distinguish logic-0 and logic-1. Three-valued logic extends the two-valued logic by only one additional value to model that a signal has an X-value – and hence is unable to distinguish different X-values.

In contrast, RSL extends the value domain by *indexed and distinguishable* X-values and allows to represent the negation of an X-value. This allows RSL to accurately evaluate simple, local X-reconvergences and the resulting X-canceling (i.e. if correlated X-values occur at several inputs of a gate and as a result a binary value is visible at the gate output). If different X-values reconverge, a new unique X-symbol is created, which has no correlation with the convergent X-sources and hence introduces pessimism.

Nevertheless, [10] showed that for ATPG, the evaluation of local reconvergences already allows to generate a test for almost all detectable faults.

C. Conventional Value Encoding for SAT-Based ATPG

In this work, we map the test pattern generation to a Boolean satisfiability (SAT) problem. Most modern SAT-solvers expect a conjunctive normal form (CNF) as input. A CNF is a conjunction of clauses with each clause being a disjunction of literals – each literal is either a Boolean variable or its negation.

The size of the CNF encoding highly depends on the number of distinguishable values modeled for a signal. In two-valued logic, the possible values are logic-0 and logic-1, and one literal is sufficient for each signal. In order to handle unknowns, three-valued logic additionally supports an X-value. This requires two literals l_1, l_2 to encode the possible signal values [3].

Regarding RSL, [10] uses two literals (n, x) to distinguish binary and X-values and adds $\lceil \log_2 m \rceil$ literals (called b_k) to distinguish up to m possible X-values at a certain signal using a binary encoding of the X-indices.

D. Fault Detection Requirements

According to [6], a stuck-at fault is *detectable* if and only if there is a test pattern (assignment to the controllable inputs) so that there is at least one output at which this fault is observable independent of the values of the X-sources.

III. Overview of the ATPG Framework

The proposed SAT-based ATPG framework consists of a novel signal value and gate encoding for signals that depend on X-sources. The goals are to reduce the variables required for encoding each X-dependent signal and to reduce the number and the complexity of clauses encoding gates with X-dependent inputs. A topological circuit analysis is performed to derive the optimal encoding of each signal and gate in a SAT instance.

Before test generation starts, an efficient untestability check is conducted for each fault. If a fault is not classified as untestable, the SAT instance is constructed to model the fault free and faulty circuits and the relevant propagation of X-values. It is analyzed by a SAT solver. If the instance is satisfiable, a test pattern for the target fault is found. In addition, an incremental accurate fault simulator analyzes patterns and finds addtionally detected faults.

Mixed RSL Encoding of Signal Values and Gates

The proposed *Mixed 01X-RSL* encoding combines the advantages of three-valued and restricted symbolic logic. In a three-valued encoding, only three values need to be distinguished at a signal, but all reconvergences are evaluated pessimistically. In an RSL encoding, a significantly larger number of signal values is distinguished, allowing much more accurate reasoning about reconvergences.

The proposed method encodes only those X-values at a signal as distinguishable (indexed) X-values using RSL that might actually reconverge and cause X-canceling. For all other signals, a three-value encoding is used. This requires to determine the propagation paths of X-values from the X-sources and to compute the set of X-values that might reconverge at a signal in a preprocessing step (cf. Section IV-B). This reduces the number of X-values to be encoded using RSL without reducing the accuracy.

IV. ATPG Algorithm

This section describes the novel value encoding and the construction of the SAT instance for test generation.

A. Details of the Mixed 01X-RSL Encoding

In order to distinguish pessimistic (unnamed) and RSL X-values, we extend the encoding of [10] to allow that a signal in the SAT instance might show both pessimistic and RSL X-values. If only one of both is possible at a signal, the needless case is omitted in order to reduce the encoding overhead.

The following table summarizes the different literal types in our encoding:

Literal	Meaning
n	$n = 1 \leftrightarrow$ signal is negated
x	$x = 1 \leftrightarrow$ signal has X-value
x_p	$x_p = 1 \leftrightarrow$ X-value is encoded pessimistically
b_k	X_i $i = \{1, \ldots, m\}$ binary encoded, $k \in \{1, \ldots, \lceil \log_2 m \rceil\}$

This encoding easily allows the handling of two-valued signals because in this case only the n literal is required, while the x

literal always shows logic-0 and hence neither x_p nor b-literals are required and the corresponding gate clauses is simplified. In addition, if only a pessimistic X-value is possible at a signal, the encoding is only slightly larger than a classical three-valued encoding – as no b-literals are required and the x_p literal is always assigned logic-1.[1] When referring to a literal l of signal s, we use the notation $s[l]$, e. g. $i1[n]$ for the n-literal of signal $i1$.

B. Construction of the Mixed 01X-RSL Instance

Preprocessing for mixed 01X-RSL encoding: At first, all possible X-values per signal that need to be considered in the CNF are computed by a topological netlist analysis. Figure 2 shows the result of the preprocessing proposed in [10]. Already for this small example, four different (indexed) X-values need to be distinguished. Since at gates G_4 and G_5, different X-values may converge, new unique X-symbols X_3, X_4 are introduced. However, only the X-value originating at the X-source at input b might *re*converge and potentially cause X-canceling.

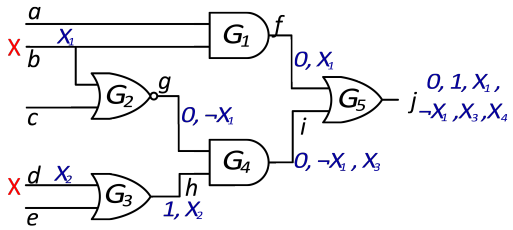

Fig. 2: Result of the preprocessing proposed in [10].

Thus, all other X-values can be replaced by one unique X-value (X_P), which is evaluated pessimistically according to the three-valued logic. This does not reduce the overall accuracy during test generation and leads to the refinement shown in Figure 3.

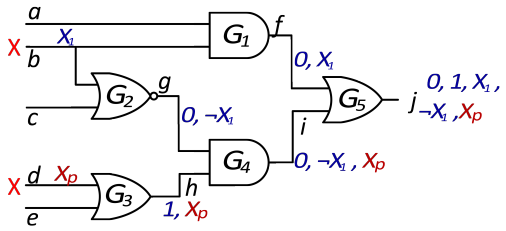

Fig. 3: Refinement for creating a Mixed 01X-RSL CNF.

If only one X-value remains at a signal, the b-literals, which encode the index of an X-value, can be removed and the depending gate clauses significantly simplified.

All gates at which only a pessimistic X-value is possible after the refinement don't need an RSL encoding anymore. This reduces the literals and clauses significantly: For the example above, the number of literals required for gate G_5 and output j is reduced from eleven to only four in the proposed encoding. For the full example, the number of clauses is reduced by over 75% since the expensive comparator circuits used to compare the indices of RSL values can be avoided.

Gate encoding using a mixed 01X-RSL encoding: In contrast to a pure RSL encoding, which only considers n, x and b-literals, pessimistic X-values (x_p literals) need to be considered in the proposed mixed encoding as well. The following cases are considered: (1) no X-values are possible at the inputs of a gate; (2) only one input might show an X-value; (3) two or more inputs might show an X-value. From these three cases, cases (1) and (2) can easily be handled with only a few clauses.

If no X-values are possible at all, a two-valued encoding only using the n-literal is sufficient. Thus, an AND gate with input variables $i1[n]$, $i2[n]$ and output $o[n]$ is represented by three clauses:

$$(i1[n] \vee \neg o[n]) \ , (i2[n] \vee \neg o[n]) \ , (\neg i1[n] \vee \neg i2[n] \vee o[n]).$$

If only one input may have an X-value, we distinguish: If an X-source is modeled, the corresponding x literal is always assigned logic-1. Furthermore, we keep track of the assigned b or x_p literal. If a gate is modeled, an X-value at the input is either blocked or forwarded. Thus, it is sufficient to consider the n and x literals for such a gate. For an two input AND gate with X-dependent input $i1$, this leads to the clauses:

$$\begin{array}{l}(\neg i2[n] \vee o[n] \vee \neg i1[n]) \ , (\neg o[n] \vee i1[n]) \ , \\ (\neg i2[n] \vee o[x] \vee \neg i1[x]) \ , (\neg o[x] \vee i1[x]) \ , \\ (i2[n] \vee \neg o[n]) \ , (i2[n] \vee \neg o[x]).\end{array}$$

If both inputs might show an X-value, three different cases are distinguished: If only RSL X-values are possible, a similar encoding as proposed in [10] is used. However, as we want to save as many clauses as possible, all information from the enhanced preprocessing are directly considered for the creation of the clauses. This leads to the following differences:

1) If only different X-values are possible at the inputs, the comparator circuit to distinguish the X-indices is not required, and no clauses are required to handle the case that a reconvergence might occur.
2) For all b literals that are permanently assigned, the corresponding clauses in the gate encoding are simplified.
3) If a new X-value originates at the output of a gate that will never reconverge, a pessimistic X-value is modeled instead of an RSL X-value.

If only pessimistic X-values are possible at the inputs, no b-literals need to be used for them, and the x_p literals of the inputs are always assigned logic-1. If a new X-value originates at the output and may reconverge according to the preprocessing, then the output is assigned an RSL X-value even though the inputs show only pessimistic X-values. In case only a pessimistic X-value is possible for the output, its x_p literal is permanently assigned logic-1. Consequently only the n and x literals need to be considered in the encoding of the gate. Similarly, also in case of a new RSL X-value at the output, all b literals are already assigned since only one RSL X-value is possible here. As a consequence, both cases can be handled with the same clauses. For an AND gate with two inputs this is expressed by the following implications:[2]

$$\begin{array}{ccc}(i1[n] \wedge i2[n]) & \leftrightarrow & o[n] \\ (\neg i1[n] \wedge \neg i1[x]) & \rightarrow & \neg o[x] \\ (\neg i2[n] \wedge \neg i2[x]) & \rightarrow & \neg o[x] \\ (i1[n] \wedge \neg i1[x] \wedge i2[n] \wedge \neg i2[x]) & \rightarrow & \neg o[x] \\ (o[n] \wedge \neg o[x]) & \rightarrow & (\neg i1[x] \wedge \neg i2[x]) \\ (\neg o[x] \wedge i1[n] \wedge \neg i1[x]) & \rightarrow & \neg i2[x]\end{array}$$

[1] As shown in Section IV-B, in these cases the encoding is adjusted before the creation of the corresponding clauses. Hence, no x_p literal is required.

[2] Please note: this case can also be expressed by only 6 clauses using a classical three-valued encoding. However, this would require to adjust the complete encoding leading to disadvantages in case RSL X-values need to be considered.

$$
\begin{array}{rcl}
(\neg o[x] \wedge i2[n] \wedge \neg i2[x]) & \rightarrow & \neg i1[x] \\
(i1[x] \wedge i2[n] \wedge \neg i2[x]) & \rightarrow & o[x] \\
(i2[x] \wedge i1[n] \wedge \neg i1[x]) & \rightarrow & o[x] \\
(o[x] \wedge i2[n] \wedge \neg i2[x]) & \rightarrow & i1[x] \\
(o[x] \wedge i1[n] \wedge \neg i1[x]) & \rightarrow & i2[x]
\end{array}
$$

If both pessimistic and RSL X-values are possible, a mixture of both encodings is used. If a pessimistic X-value is present at an input, the RSL-clauses are deactivated, and the gate is handled similarly to the case if no b-literals are present. Otherwise, if both inputs do not show a pessimistic X-value, the case is similarly handled as if only RSL X-values are present. If an output might carry either an RSL or pessimistic X-value, the x_p literal is required and cannot be removed in the encoding.

Encoding of the D-Chain: [10] distinguished two different types of D-chains [15] to model possible propagation paths of value differences between the fault-free and faulty circuits. One that allows a higher accuracy (*complete* RSL) and one with a slightly reduced accuracy but the benefit of a notably reduced runtime (*optimized* RSL).

For a mixed 01X-RSL instance, both modes are supported, too. Yet, the results of [10] showed that the difference in the achieved fault coverage between the optimized RSL based and the complete RSL based ATPG is rather small, while the runtime differs significantly. Thus, we omit the case of complete RSL here and focus on optimized RSL.

Optimized RSL restricts the D-Chain and only considers differences that are likely to lead to a valid test pattern. As an example consider a signal that is affected by a fault and shows a logic-1 in the fault-free circuit and an X-value in the faulty circuit. Another signal shows at the same time a logic-1 in the fault-free circuit and the negation of the same X-value in the faulty circuit. A later reconvergence in the faulty circuit may cause X-canceling, but this is typically very rare. Thus, all combinations for which either the fault-free or faulty circuit, but not both, consider an X-value at a signal propagating the fault effect are excluded and not considered a valid difference in the D-chain. This might slightly decrease the accuracy, but the more strict D-chains allow a notably reduced runtime.

To generate such a D-chain within our mixed 01X-RSL encoding, the definition of difference is extended to support pessimistic X-values. Since pessimistic X-values in our encoding never reconverge and never cause X-canceling, they do not propagate a fault effect. Thus, in the D-chain for the mixed 01X-RSL encoding, it is forbidden that a pessimistic X-value is present along the propagation path.

The described techniques are used for ATPG for stuck-at faults, but can also be applied for ATPG targeting transition delay faults by unrolling or time frame expansion.

V. Two-valued SAT-based Untestability Check

According to Section II-D, a fault is detected by a pattern if there is an observable output difference for all possible assignments to the X-sources. Consequently, a fault is untestable if there is an assignment to the X-sources such that there is no test pattern for the controllable inputs that creates an output difference for the target fault.

While [13] implements this untestability check utilizing an SMT-solver, we propose a pure two-value encoded SAT instance to reduce the runtime. The instance considers up to two different assignments to the X-sources. This allows to quickly identify a large fraction of untestable faults and reduces the number of faults for which the more expensive mixed 01X-RSL based analysis is conducted.

In detail: First, we create a pure two-valued CNF of the fault free and faulty circuit and assign randomized values to each X-source (cf. Figure 4 left side). In case the resulting CNF is unsatisfiable, the fault is proven to be untestable. Otherwise, we extend the CNF with a second instance of the circuit – including only those signals and gates, which depend on the X-sources (cf. Figure 4 right side). The X-sources of the second instance are assigned with the negated assignment of the first instance. Furthermore, we add extra clauses in order to force the SAT-solver to propagate the fault effect in both instances to the same output.

Adding a second instance to the CNF strengthens the untestability check, because now we check whether a fault is testable with two different assignments to the X-sources simultaneously. It would also be possible to evaluate even more instances to further strengthen the untestability check. However, experiments showed that two instances trade-off runtime and accuracy very well.

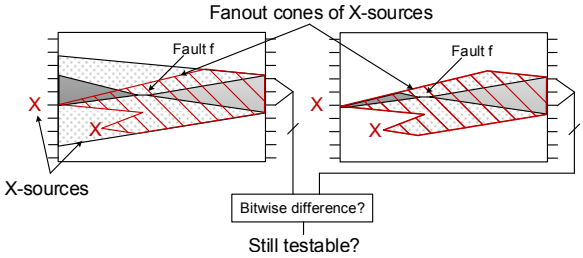

Fig. 4: SAT-based Untestability Check for a fault f considering two different assignments to the X-sources at the same time.

VI. Incremental Accurate Fault-simulation

To implement fault-dropping for each test pattern generated by the proposed mixed 01X-RSL based ATPG, an extended event-based version of the fault simulator proposed in [7] is used. This fault simulator allows an accurate fault classification for a given test pattern in presence of X-values. It maps the decision whether a fault is detected to a Boolean SAT instance.

The principal functionality of the original implementation is as follows: For a pattern to be simulated, first a plain accurate fault-free simulation was performed. Then, for each yet undetected fault it was tested if the fault can be propagated to at least one output. Both fault-free and faulty circuit simulation use a combination of pattern-parallel logic simulation of randomized X-source assignments and RSL-based simulation to classify as many faults as possible accurately without the need to invoke a much more costly SAT-based classification. The simulation of random X-source assignments already allows to prove for many signals that they indeed carry an X-value. The RSL-based simulation on the other hand evaluates many reconvergences accurately. The remaining pessimistically computed signal values are subject to the SAT-based evaluation. More details are given in [7].

Processing the patterns and the resulting input value changes in an event-based manner decreases the overhead of the parallel logic simulation, restricted symbolic simulation, as well as the number of required SAT-solver calls. The improvements in detail are:

1) In the accurate fault-free simulation, input change events are created when input values change from a binary value

to the complementary binary value, or if there is a change from an X-value (binary value) to a binary value (X-value). Only gates that have input events are evaluated.
2) If signal values changed during the event-based fault-free simulation, only those faults are considered in the subsequent fault simulation step for which either the fault-site itself lies in a region with changed values, or for which at least one input of an affected gate changed.
3) The SAT-based classification algorithm invoked for a final value classification is only used if a change in the fault-free simulation also had effects on the propagation of X-values in the circuit.

In this way, the number of costly SAT invocations and resulting runtime is greatly reduced.

VII. Evaluation

The proposed algorithms are implemented in C. All SAT-based approaches use the incremental SAT-solver antom [16]. For comparison a state-of-the-art commercial tool was used. We evaluate the algorithms on full-scan circuits of the largest ISCAS'85 benchmarks (c6288, c7552) and larger industrial designs from NXP. All experiments were conducted on an Intel Xeon CPU with 3.3 GHz. Structural fault collapsing is performed.

We assume that a fixed and randomly selected subset of circuit inputs is used as X-sources.[3] For each circuit, five different X-source assignments are considered. The reported results are the rounded average over these five experiments per circuit.

A. Comparing Mixed 01X-RSL and pure RSL Encoding

Firstly, we evaluate the runtime impact of using a mixed 01X-RSL encoding instead of a pure RSL encoding. Thus, we compare the optimized pure RSL based algorithm of [10] with the proposed optimized mixed 01X-RSL based algorithm. Figure 5 shows the normalized runtime for two industrial designs (p78k and p89k) in two different scenarios. p78k is a circuit with many reconvergencies and hence, for most faults only a few RSL X-values could be replaced by pessimistic X-values (on average less than 5%). In contrast, for circuit p89k on average 40% of the X-values considered in a CNF could be replaced and modeled by pessimistic X-values. In the first scenario (cf. Figure 5 left) all faults are solely handled by the pure RSL or mixed 01X-RSL based algorithm. Two things can be observed: (1) for circuit p89k the runtime reduces by 37.61%, while (2) for circuit p78k the preprocessing runtime overhead completely consumes the benefit of the mixed encoding (especially for the easy-to-detect faults).

In the second scenario (cf. Figure 5 right), the easy-to-detect faults are removed by applying random patterns before using the pure RSL or mixed 01X-RSL based algorithm. As a result, the mixed 01X-RSL based algorithm reduces the runtime for p78k by 35.74% and for p89k by 47.25%.

B. Fault Coverage in Larger Circuits

In order to achieve a high fault coverage while keeping the runtime as low as possible, in experiment two the proposed mixed 01X-RSL based algorithm was combined with the untestability check proposed in Section V and the incremental accurate fault simulator (cf. Section VI) for fault-dropping. Prior to each ATPG run, all faults detected by the used commercial tool are marked as detected and removed from a further analysis. The runtime for removing all easy-to-detect faults with the commercial tool is always included in the reported runtime of the proposed framework.

[3]The effect of clustered X-sources in ATPG has been discussed in [17].

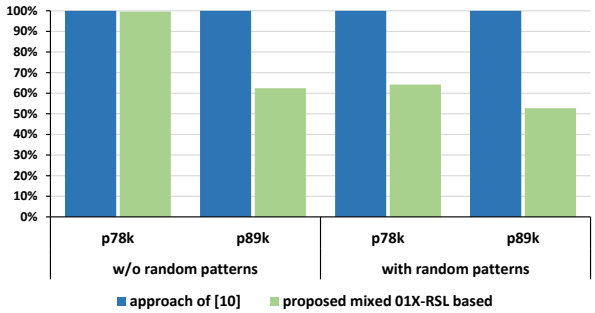

Fig. 5: Normalized runtime of the pure RSL based and mixed 01X-RSL based test generation with and without random patterns.

Table I compares the results of the proposed framework to the fault coverage of the commercial tool as well as the runtimes of the optimized RSL of [10]. For each circuit, an X-ratio of 1%, 2% and 5% was considered. For the academic benchmark c6288 with only 32 inputs, the case of 2% was omitted as it results in a single X-source. Furthermore, the runtime of the commercial tool is also listed to show the fraction of the total runtime of the proposed framework used by the commercial tool.

The proposed ATPG increases the fault coverage compared to the commercial tool considerably. Considering the ISCAS'85 benchmarks, the fault coverage ('FC') increases by up to 34.24% points (c6288 and an X-ratio of 5%).

For the industrial designs from NXP, the proposed ATPG increases fault coverage by up to 7.77% points (p78k and an X-ratio of 5%). On average over all considered industrial designs and an X-ratio of 5%, the proposed ATPG achieves a 3.80% points higher fault coverage than the commercial tool.

Using the proposed ATPG, for almost all circuits no aborts occurred at all. In total only 49 faults were aborted – for the circuit p100k less than 0.005% and for circuit p388k less than 0.002% of the faults were aborted.

Most of the faults that could not be classified as detectable by the proposed approach were classified as untestable by the incorporated untestability check ('UT'). Only a very small number of faults stays unclassified ('UC'). Thus the combination of RSL based ATPG with a fast untestability check classifies for all circuits at least 97% of the faults as detected or proves fault untestability. On average over all tested circuits and X-ratios, only 0.67% stay unclassified.

The runtime of the proposed optimized mixed 01X-RSL framework was always below 2h. In total, 20 611 586 non-equivalent faults were classified in less than 9 hours.

Comparing this optimized mixed 01X-RSL framework to the results of the pure optimized RSL-based one of [10], the number of aborts and the runtime reduces significantly. The number of aborts reduces by 95.91%, and the total runtime reduces by 88.59% (i.e. a speed-up of 8.8×). The highest speed-up of over 88× is achieved for circuit p267k and 1% of the inputs selected as X-sources. Here, the proposed framework reduces the runtime from 11 725s to 132s.

Results from experiments for a mixed 01X-RSL based ATPG version of the *complete* RSL-based approach (cf. Section IV-B, [10]) are in line with the differences observed when comparing the optimized RSL-based versions.

VIII. Conclusions

In this paper we proposed a novel ATPG framework combining new techniques to overcome the limitations of a pure RSL-based encoding for ATPG in presence of unknown values. The proposed mixed value encoding combines the accuracy of RSL with the compactness and efficiency of a three-valued encoding to achieve high accuracy and high scalability at the same time. Furthermore, a low-cost two-valued SAT-based untestability check and an incremental event-based accurate fault simulator is proposed.

The experimental results show the effectiveness and high efficiency of the proposed ATPG framework, reducing the number of aborts by over 95% compared to the state-of-the-art ATPG. At the same time, an average speedup of $8.8\times$ is achieved. This allows the RSL-based evaluation of stuck-at faults for circuits with over one million gates for the first time. The proposed techniques are also applicable for transition delay fault ATPG.

Acknowledgments: This work was partially supported by the German Research Foundation (DFG) under grants BE 1176/14-2, SFB/TR14 AVACS, WU 245/17-1 (ACCESS), and GRK1103.

References

[1] J. P. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM J. Res. Dev.*, vol. 10, no. 4, pp. 278–291, july 1966.

[2] P. Muth, "A nine-valued circuit model for test generation," *IEEE Trans. on Computers*, vol. C-25, no. 6, pp. 630–636, june 1976.

[3] A. Jain, V. Boppana *et al.*, "Testing, verification, and diagnosis in the presence of unknowns," in *Proc. IEEE VLSI Test Symposium (VTS)*, 2000, pp. 263–268.

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

[5] M. A. Kochte, M. Elm, and H.-J. Wunderlich, "Accurate X-propagation for test applications by SAT-based reasoning," *IEEE Trans. CAD*, vol. 31, no. 12, pp. 1908–1919, 2012.

[6] S. Hillebrecht, M. A. Kochte *et al.*, "Exact stuck-at fault classification in presence of unknowns," in *Proc. IEEE European Test Symposium (ETS)*, 2012, pp. 1–6.

[7] D. Erb, M. A. Kochte *et al.*, "Exact logic and fault simulation in presence of unknowns," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 19, no. 3, Jun. 2014.

[8] S. Hillebrecht, M. A. Kochte *et al.*, "Accurate QBF-based test pattern generation in presence of unknown values," in *Proc. Conf. on Design, Automation and Test in Europe (DATE)*, 2013, pp. 436–441.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1979.

[10] D. Erb, K. Scheibler *et al.*, "Test Pattern Generation in Presence of Unknown Values Based on Restricted Symbolic Logic," in *Proc. IEEE International Test Conference (ITC)*, 2014, pp. 1–10.

[11] M. A. Breuer, "A note on three-valued logic simulation," *IEEE Transactions on Computers*, vol. 21, no. 4, pp. 399–402, Apr. 1972.

[12] J. Carter, B. Rosen *et al.*, "Restricted symbolic evaluation is fast and useful," in *Proc. IEEE Int. Conf. Comp.-Aided Design (ICCAD)*, 1989, pp. 38–41.

[13] K. Scheibler, D. Erb, and B. Becker, "Improving Test Pattern Generation in Presence of Unknown Values beyond Restricted Symbolic Logic," in *Proc. IEEE European Test Symposium (ETS)*, May 2015.

[14] S. C. Kleene, *Introduction to Metamathematics*. North-Holland Publishing Co., Amsterdam, 1952.

[15] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Trans. CAD*, vol. 11, no. 1, pp. 4–15, jan 1992.

[16] T. Schubert, M. Lewis, and B. Becker, "Antom—solver description," *SAT Race*, 2010.

[17] D. Erb, M. A. Kochte *et al.*, "Accurate QBF-based Test Pattern Generation in Presence of Unknown Values," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2015.

TABLE I: Results of the proposed ATPG framework in contrast to a state-of-the-art commercial tool.

circuit	gates	faults	X-Ratio	Commercial Tool		**Proposed Optimized Mixed 01X-RSL & accurate fault sim.**							Time[s]
			[%]	FC[%]	Time[s]	FC[%]	ΔFC [%pt.]	UT[%]	UC[%]	Aborts	Time[s]	Speedup to [10]	of [10]
c6288	2 416	8 704	1.0	82.46	41	97.38	14.92	2.60	0.02	0	53	0.17X	12
			5.0	60.80	81	95.04	34.24	4.75	0.21	0	91	0.24X	22
c7552	4 043	10 816	1.0	91.78	4	92.73	0.94	7.23	0.04	0	4	0.57X	2
			2.0	88.50	5	90.20	1.70	7.80	2.00	0	5	0.75X	4
			5.0	68.01	9	74.41	6.40	22.68	2.90	0	9	1.17X	10
p78k	74 243	225 476	1.0	97.29	10	98.66	1.37	1.22	0.12	0	17	1.00X	17
			2.0	93.59	20	96.94	3.35	2.72	0.34	0	36	1.00X	36
			5.0	84.24	38	92.01	7.77	5.99	2.01	0	81	1.40X	114
p89k	88 726	239 090	1.0	91.94	78	92.11	0.16	7.70	0.19	0	79	19.98X	1 579
			2.0	86.32	84	86.58	0.26	13.07	0.34	0	122	15.73X	1 924
			5.0	71.74	38	72.75	1.02	26.36	0.89	0	260	10.72X	2 789
p100k	96 685	259 322	1.0	95.28	121	96.12	0.84	3.74	0.15	0	135	15.39X	2 080
			2.0	91.40	181	93.43	2.03	6.23	0.34	1	349	7.36X	2 565
			5.0	80.29	468	83.76	3.53	14.72	1.46	13	2 604	9.50X	24 745
p141k	172 686	452 599	1.0	95.86	40	96.24	0.37	3.39	0.38	0	1 304	2.79X	3 645
			2.0	93.90	44	94.51	0.61	5.02	0.47	0	1 811	2.63X	4 755
			5.0	85.82	77	87.62	1.80	11.26	1.12	0	4 137	2.53X	10 475
p267k	271 538	658 395	1.0	94.71	42	95.01	0.31	4.84	0.15	0	132	88.77X	11 725
			2.0	90.95	49	91.33	0.38	8.17	0.50	0	299	50.43X	15 061
			5.0	80.00	84	81.95	1.95	16.25	1.80	0	779	32.02X	24 944
p378k	371 538	1 127 364	1.0	96.83	65	98.51	1.69	1.39	0.10	0	97	-	-
			2.0	93.59	104	96.95	3.36	2.72	0.34	0	192	-	-
			5.0	84.46	195	92.16	7.70	5.92	1.92	0	590	-	-
p388k	489 271	1 352 303	1.0	93.86	360	97.37	3.51	2.54	0.09	0	1 491	-	-
			2.0	91.71	634	95.61	3.90	4.17	0.22	10	2 026	-	-
			5.0	83.03	2 729	88.29	5.26	10.74	0.97	24	6 508	-	-
p951k	1 002 883	2 539 361	1.0	96.39	474	96.73	0.35	3.22	0.05	0	1 399	-	-
			2.0	93.69	797	94.30	0.60	5.54	0.16	0	2 540	-	-
			5.0	85.28	1 107	86.69	1.40	12.56	0.76	0	4 977	-	-