Fast and Waveform-Accurate Hazard-Aware SAT-Based TSOF ATPG

Jan Burchard*, Dominik Erb*, Adit D. Singh†, Sudhakar M. Reddy‡ and Bernd Becker*

* University of Freiburg
Freiburg, Germany
{ burchard | erbd | becker }
@informatik.uni-freiburg.de

† Auburn University
Auburn, AL, USA
adsingh@eng.auburn.edu

‡ University of Iowa
Iowa City, IA, USA
sudhakar-reddy@uiowa.edu

***Abstract*—Opens are known to be one of the predominant defects in nanoscale technologies. Especially with an increasing number of complex cells in today's VLSI designs intra-gate opens are becoming a major problem. The generation of tests for these faults is hard, as the timing of the circuit needs to be considered accurately to prevent the invalidation of the generated tests through hazards. Current test generation methods, including new cell aware tests that explicitly target open defects, ignore the possibility of hazard caused test invalidation. Such tests can fail to detect a significant fraction of the targeted opens.**

In this work we present a waveform-accurate hazard-aware test generation approach to target intra-gate opens. Our methodology is based on a SAT-based encoding and allows the generation of tests guaranteed to be robust against hazards. Experimental results for large benchmarks mapped to the state-of-the-art NanGate 45nm cell library including complex cells show the test generation efficiency of the proposed method. Large circuits were efficiently handled – even without the use of fault simulation. Our experiments show that on average, about 10.92 % of conventional hazard-unaware tests will fail to detect the targeted opens because of test invalidation – these are reliably detected by our new test generation methodology. Importantly, our approach can also be applied to improve the effectiveness of commercial cell aware tests.

I. Introduction

Recent silicon data on manufacturing defects in VLSI designs shows that opens are predominant in nanoscale technologies [1], [2], [3]. Opens can be partial, resistive or full. Furthermore, they can occur within gates (intra-gate opens) or between different gates. In this work, we focus on full intra-gate opens. Intra-gate opens have been modeled as transistor stuck-open faults (TSOFs) [4] and cross wire faults [5]. Both can be detected by two-pattern TSOFs tests [5], [6].

The generation of tests to detect TSOFs has been studied extensively in earlier times [7], [8], [9], [10], [4], [11] as well as very recently [12], [13], [14], [5], [6]. Earlier it was shown that tests for TSOFs could be invalidated by circuit delays that cause certain glitches or hazards in the gate containing the fault [15], [16]. Tests could also be invalidated due to charge sharing between intra-gate capacitances and gate outputs. However, the occurrence of such latter invalidations can be expected to be improbable [17]. In this work, we consider the generation of tests for TSOFs that remain valid when considering the presence of circuit delays and, hence, are not based on simplified assumptions.

In order to address test invalidation due to circuit delays, design methods for CMOS circuits have been proposed which ensure that they are testable for TSOFs with tests that cannot be invalidated by circuit delays [17], [16], [18]. To achieve the desired testability, these design methods require adding extra logic to almost all gates of a circuit which may not only increase the circuit area, but also reduce the operation speed. Another DFT approach to implement circuits using CMOS logic such that TSOFs in the circuit are testable by tests that are not invalidated by circuit delays was studied in [19] and [20]. Both methods require the iterative factorization of the function to be realized using Shannon's Expansion Theorems. Still, circuits using these methods will have large number of logic levels as well as higher gate counts, and may not be practical for most applications.

For arbitrary CMOS logic circuits a method to generate what are called robust tests for TSOFs that cannot be invalidated by circuit delays was investigated in [21]. These tests will remain valid under arbitrary circuit delays. The method uses an analysis method for static hazard detection [22] to avoid static hazards in the faulty gate that could invalidate tests being generated. Since hazard analysis is pessimistic it predicts hazards even when the actual circuit delays do not cause hazards and thus may not find tests for some TSOFs even when they exist.

Because of the complexity of generating robust tests, current industrial test generation methods, including new cell aware tests that explicitly target open defects, ignore the possibility of hazard based test invalidation. Such tests can fail to detect a significant fraction of the targeted TSOFs. In this paper we leverage a timing accurate test generation algorithm to efficiently generate two-pattern tests to detect TSOFs that are not invalidated by hazards caused by circuit delays.

In detail we present:

- A new SAT-based waveform-accurate ATPG approach to target intra-gate opens.
- A stability condition which generates test that are robust against many process variations.
- A hybrid encoding that combines the benefits of accurately reflecting the switching activity with the compactness of standard delay fault modeling.
- A fast preprocessing step to identify faults that can be accurately handled without considering the timing information.
- A thorough investigation of the number of invalidated tests in case the accurate timing is not considered within test pattern generation.

To demonstrate the effectiveness of the proposed method we mapped circuits from the largest ITC'99 and IWLS 2005 benchmarks as well as larger industrial circuits from NXP to the state-of-the-art NanGate 45nm cell library using complex cells. Our experimental results show the high efficiency and scalability of the proposed method. Furthermore, our approach reliably creates hazard-aware tests for all TSOFs – while timing-unaware test generation would result in on average 10.92 % of invalidated tests due to glitches. In addition, large circuits with over 100k complex cells and normal gates are efficiently handled – even without the inclusion of fault simulation.

978-3-9815370-8-6/17/$31.00

The remainder of the paper is organized as follows: In Section II we discuss the TSOF model, the circuit structure as well as our TSOF detection library. Section III focuses on our ATPG flow and the timing-unaware as well as waveform-accurate ATPGs. Subsequently, Section IV evaluates the experimental results including the obtained fault coverage, runtime and comparison to a purely timing-unaware approach. Section V concludes with a short summary and outlook onto future work.

II. Preliminaries

A. Transistor Stuck-Open Faults

In the transistor stuck-open fault (TSOF) model a transistor in a gate of a circuit is always open and cannot form a conducting path between its source and drain ports. This can result in the gate's output not being connected to V_{DD} or V_{SS} under certain input conditions, effectively putting the output in a high impedance state. Under these conditions the output will retain its previous value for a certain amount of time due to the gate and line capacitances [4].

To test for the existence of a TSOF a two pattern test $\langle T_1, T_2 \rangle$ is required [15]. The initialization pattern T_1 charges the faulty gate's output to a defined logic level by creating a conducting path to either V_{DD} or V_{SS}. The propagation pattern T_2 makes the fault effect visible at the output by creating a path through the faulty transistor which results in an unchanged gate output if the fault is present and a change to the output value if it is not. When switching from T_1 to T_2 some of the gate inputs may have to be stable (i.e., glitch free) to ensure that the output is not accidentally (dis-)charged and the fault effect masked [15], [16].

As an example consider the AND gate in Figure 1 and assume the transistor M_i_4 is stuck-open. A possible test for this fault would be $\langle 11, 01 \rangle$. The T_1 pattern 11 connects the intermediate output ZN_neg to V_{SS}, resulting in a logic '1' at the output ZN. The T_2 pattern 01 creates a path from V_{DD} to ZN_neg through M_i_4. When M_i_4 is stuck-open, ZN_neg maintains its previous value (since no other path to V_{DD} or V_{SS} exists), otherwise it changes to logic '1' (resulting in '0' at the output).

The input A_2 has to be stable for this test; A '0' glitch at A_2 invalidates the detection because ZN_neg would be connected to V_{DD} through M_i_5, resulting in a logic '0' at the output (which should only occur in the fault free case).

We call a two pattern test with the scope of a single gate a *detection pattern*, whereas the assignment to the circuit's inputs is called a *test pattern*. A test pattern for a TSOF creates the required detection pattern at the gate's inputs while also ensuring the propagation of the fault effect to the circuit's outputs making it observable.

B. Circuit Structure

We consider sequential circuits with full scan based on standard CMOS gates (e.g., AND, OR, NAND, XOR, ...) as well as complex cells (e.g., AND-OR-Invert, ...) in combination with their timing information. Within our algorithms, the complex cells are mapped to standard gates through a Verilog library. This allows for the incorporation of any kind of cell structures, present in a gate library, into the circuit without changes to the ATPG algorithm itself. The timing information for the complex cells is transferred to the mapped circuit's gates, resulting in timing-equivalent circuits.

We assume that the test infrastructure is launch-on-capture. Therefore, the values in the flip-flops in the second time frame are derived from the flip-flop inputs after the first time frame.

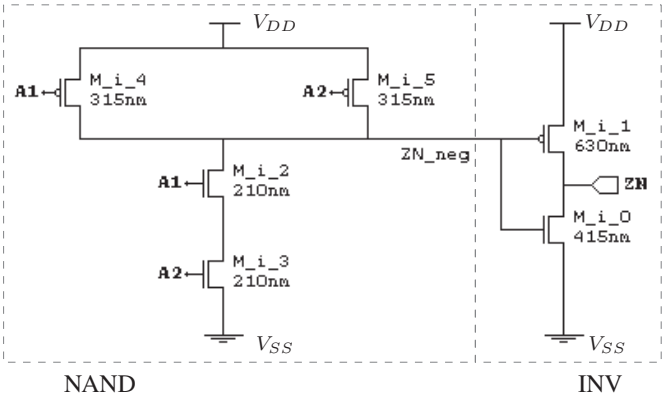

Figure 1. Transistor layout of an AND gate with inputs $A1$ and $A2$ and output ZN as defined by [23]. It is implemented as a NAND gate with an attached inverter.

C. Detection Library

For each possible TSOF in each gate of the considered gate library we pre-compute all $\langle T_1, T_2 \rangle$ detection patterns and store them in a detection library. The ATPG algorithm can then read the required patterns directly from the library, greatly reducing the computation overhead since the pattern creation is performed only once per gate type (instead of once per gate).

1) Automatic Detection Pattern Computation: The detection pattern library is created automatically based on the spice files for each gate provided by the gate library. These spice files contain the circuits consisting of N-FET and P-FET transistors which constitute the gates. For each TSOF all detection patterns are found by exhaustive simulation of the circuit. Due to the low number of inputs of each individual gate the time required for such a simulation is very low (in [23] the highest number of gate inputs is 6).

The overall detection pattern computation for each TSOF works as follows:

1) Pick input patterns T_1 and T_2 which have not been simulated yet.
2) Modify the circuit by disabling the transistor affected by the TSOF.
3) Apply T_1 followed by T_2. If the value of the output changes between the two patterns, the $\langle T_1, T_2 \rangle$ combination is not a detection pattern for the TSOF, continue with the first step.
4) Modify the circuit by enabling all transistors. Apply T_2 again. If the output remains unchanged the $\langle T_1, T_2 \rangle$ combination is not a detection pattern for the TSOF, continue with the first step.
5) Store the current $\langle T_1, T_2 \rangle$ pattern as a detection pattern for the TSOF. Continue with the first step.

There may be many different detection patterns for the same TSOF which are all added to the detection library.

2) Input Stability: The above procedure does not take any stability constraints into account, yet. Every detection pattern is, therefore, further analyzed to determine which (if any) of the inputs that do not change from T_1 to T_2 have to be stable (i.e., glitch free). To this end, all input transitions required to switch from T_1 to T_2 are stored in a transition list t.

For every subset of the remaining inputs a new transition list t' with two transitions representing the glitches is created. Each possible transition sequence of $t \cup t'$ is then simulated to determine if glitches at the current subset of inputs invalidate

the test. Out of these subsets, the algorithm selects the set of inputs that have to be stable to ensure that the pattern is not invalidated.

The final database entry contains the detection pattern $\langle T_1, T_2 \rangle$ and the list of inputs which must be glitch free as well as the gate's output value in the fault free and fault affected case.

It should be noted that some faults do not require any stable inputs. Assume, for example, that the transistor M_i_1 in the AND gate in Figure 1 is stuck open. A detection pattern for this TSOF has to create a conducting path from V_{SS} to the output ZN in T_1 and a path through M_i_1 in T_2. When switching from T_1 to T_2 the output must not be connected to V_{DD} through any other path since this would invalidate the test. However, the only path to V_{DD} leads through M_i_1 which is always open. Thus, no glitch can invalidate the test.

III. Hazard-Aware TSOF ATPG

We developed a TSOF ATPG algorithm which ensures that the test patterns are not invalidated through glitches by taking into account the timing information of the circuit and modeling the propagation of the signal changes induced by switching from T_1 to T_2 with waveform accuracy. The overall ATPG flow is shown in Figure 2.

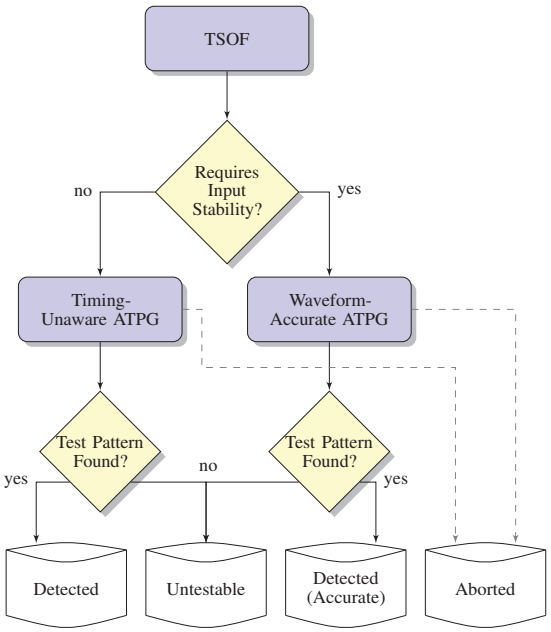

Figure 2. Overview of the ATPG flow.

Depending on the requirements of the detection pattern(s) for the TSOF it is either handled by a faster, timing-unaware ATPG algorithm or by a timing-aware, waveform-accurate variant. Both ATPGs encode the problem as a SAT instance. Each TSOF is classified as detected, untestable or aborted (in case a timeout occurred). The two ATPG versions are described in detail in the following sections.

A. Timing-Unaware ATPG

For the timing-unaware ATPG the circuit is unrolled once to create a circuit instance for the T_1 and T_2 pattern each. The unrolled circuit is converted into a formula in CNF with the Tseitin transformation [24].

To reduce the size of the created propositional formula, only the required parts of the circuit are converted. These parts are determined by analyzing the cones of influence of the fault affected gate within the two time frames (see Figure 3).

The gates in the red secondary input justification cone are only necessary in the first time frame to create the values in the flip-flops that are used in the second time frame. The blue justification cones are the input cones of the gate and are required in both time frames. Additionally, in the second time frame the gray support cone which provide the side inputs for the propagation of the gate's output value as well as the green propagation cone itself need to be modeled. Finally, a second copy of the green cone in T_2 with the fault effect has to be added.

For each circuit output an additional variable encodes whether the output of the circuit with and without the fault differs. A new clause consisting of all of these variables joined by a disjunction is added to the formula, thus ensuring that it is only satisfiable if the fault can be observed on at least one output.

The TSOF affected gate itself is not converted at all, leaving its inputs and outputs as free variables in the formula. These variables are assigned based on the detection pattern for the fault. This guarantees that any satisfying assignment to the formula creates the correct input assignment at the gate and the correct value is propagated in the fault free and fault affected case.

The assignments are added as assumptions to the SAT solver. If the created formula is satisfiable, a test pattern for the TSOF can be extracted. Should the formula be unsatisfiable or a timeout occur, the assumptions are removed and the next detection pattern can be added as new assumptions. Utilizing the solver in such an iterative manner allows it to maintain learned information between individual solver calls. To further increase the effectiveness of this approach, the CNF formula is only created once per fault affected gate. All of the gate's TSOFs without stability constraints are then tested iteratively. The combination of these iterative approaches shows large speedups compared to building individual SAT instances for every TSOF and every single detection pattern.

B. Waveform-Accurate ATPG

The waveform-accurate ATPG is used for all detection patterns which require some of the inputs of the affected gate to be stable. The approach is similar to the previously discussed timing-unaware ATPG. Again, a propositional formula is created only once for each gate. The different faults in this gate and their detection patterns are tested iteratively. However, both the modeling itself as well as the assumptions used for the target gate inputs are more complex:

1) Waveform-Accurate Modeling: To realistically model the switching activity of a circuit the timing information of its gates has to be taken into account.

The timing-accurate modeling, first presented in [25], is based on discretized gate delays. For each line in the circuit the earliest and latest possible switching time is calculated. For every time step between these points in time a new variable is introduced. The set of variables representing the signal on a line is called $Tvars$. The first $Tvar$ represents the signal value under the input pattern T_1, the last $Tvar$ that under T_2. When computing the CNF representation of the circuit, the Tseitin transformation is performed for every time step, using the appropriate $Tvar$. By these means and multiple optimizations which reduce the size of the formula, a waveform-accurate representation of the circuit is created.

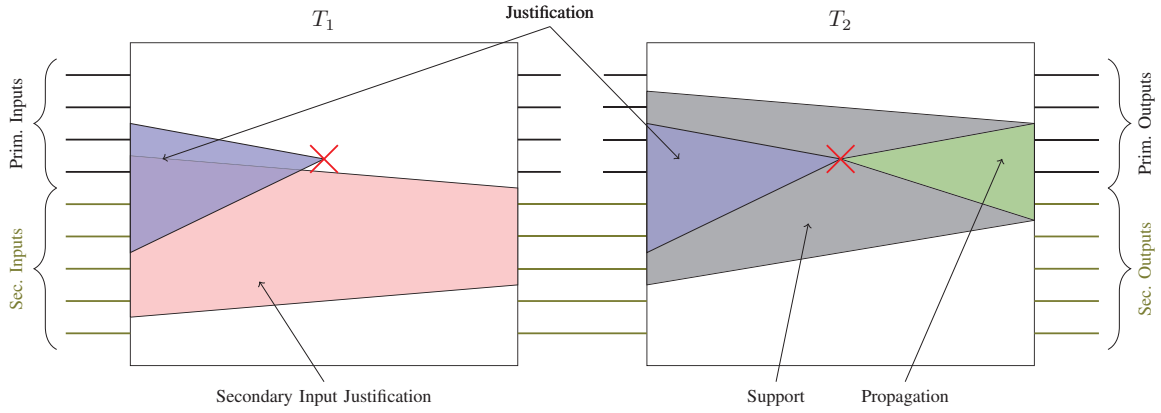

Figure 3. The cones of influence of the fault affected gate in the two considered time frames. The red cross marks the fault site.

2) Hybrid Modeling: Clearly, the waveform-accurate modeling provides the large benefit of accurately reflecting the switching activity, including glitches. However, this comes at the cost of an increased number of variables and clauses in the propositional formula (and, hence, a potentially higher solving time). Luckily, one does not need to encode the entire circuit in this manner to ensure that a TSOF is not invalidated by a glitch: Only the justification cone of the fault site (the blue cone in Figure 3) is relevant. The remainder of the circuit can be modeled similar to the timing-unaware ATPG since glitches cannot invalidate the fault propagation or its support. Therefore the resulting formula is a hybrid of timing-aware, accurate and timing-unaware, efficient modeling.

3) Input Assignments: Similar to the timing-unaware ATPG flow, the target gate itself is not modeled in the formula. Again, the values for the gate inputs in T_1 and T_2 are obtained from the current detection pattern. For a value in T_1, the first $Tvar$ of the corresponding input is added as an assumption, whereas for T_2 the last $Tvar$ is used. The $Tvar$s in between can have different values if there are glitches at the input but these are irrelevant for the initialization and test.

Inputs with a stability constraint are the exception from this rule: Here all $Tvars$ must have the same value. To enforce this condition, new clauses are added to the formula for each input:

$$\bigwedge_{i=0}^{|Tvars|} \left(\neg stable \vee (Tvar_i \Rightarrow Tvar_{(i+1) mod |Tvars|})\right) \quad (1)$$

The new variable $stable$ allows for different stability conditions in each iterative solver call. When $stable$ is not forced to be '1' by an assumption, the solver can assign it to '0', automatically satisfying all added clauses (and, hence, turning them irrelevant and removing the stability constraint on the input).

Conversely, when $stable$ is assigned to '1' through an assumption the implications ensure that all $Tvars$ of the signal have the same value, hence enforcing glitch freedom. In total for each $Tvar$ only one clause with three literals is needed to ensure stability.

From a satisfying assignment to the formula a two timeframe test pattern can be extracted. This test pattern is guaranteed to ensure glitch-freedom on all inputs of the fault affected gate that must be stable and to generate the necessary detection pattern at the gate's inputs. By enforcing such a strong stability condition[1] the generated tests are robust against many process variations because they don't contain any glitches on stable lines.

[1]A weaker stability condition could, for example, only ensure that signals remain stable from a critical point in time when a glitch would surely cause an incorrect gate output onwards.

IV. Evaluation

We evaluated the performance of our proposed ATPG algorithm on the largest ITC'99 benchmark circuits [26], IWLS 2005 cores [27] as well as large industrial designs by NXP. The circuits were synthesized with the 45nm version of the NanGate library [23] which includes many complex cells. Table I summarizes the most important circuit information.

All experiments were performed on an Intel Xeon E5-2643 CPU clocked at 3.3 GHz. As SAT solver *antom* [28] with a timeout 10 seconds was used. The time step size of the discretization of the timing was set to 20 ps.

Table I
Benchmark circuit information.

		Gates			
	Circuit	#Total	% Complex	#Mapped	#TSOF
IWLS 2005	aes_core	11 082	38.43	51 782	71 060
	des_perf	43 854	36.36	199 066	278 846
	pci_bridge_32	6 316	69.18	45 673	58 514
	vga_lcd	31 501	81.18	250 661	315 264
	wb_conmax	15 748	57.64	85 928	108 444
ITC'99	b15	3395	51.04	17 955	23 374
	b17	11 345	50.38	59 148	77 984
	b18	34 936	45.27	168 399	228 636
	b20	5 844	46.92	28 160	37 970
	b21	5 899	47.04	28 157	38 182
	b22	8 144	46.67	39 954	53 612
NXP	p45k	11 413	46.74	58 623	78 838
	p78k	25 740	46.56	174 213	209 950
	p81k	44 559	34.16	163 693	258 576
	p89k	25 209	48.12	127 844	170 922
	p100k	25 633	49.33	136 350	182 354
	p267k	47 986	57.34	294 683	377 210
	p295k	52 366	61.09	333 549	432 420
	p330k	54 287	44.42	296 111	395 182
	p378k	125 824	46.98	881 782	1 088 518
	p388k	118 920	56.35	608 599	843 628

A. General Performance

We computed test patterns for all TSOFs. The resulting fault coverage is summarized in Figure 4. In addition, Figure 5 shows the average computation time per fault. The total runtime of the entire ATPG algorithm is around 44 minutes on average. Even for the largest circuits, the total runtime is very low with a maximum of 238 minutes.

About 20 % of the TSOFs can be tested without requiring any stability on their inputs and are handled by the timing-unaware

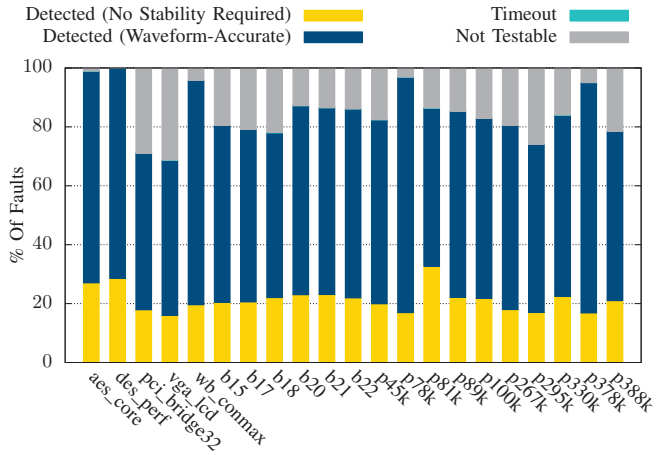

Figure 4. Fault coverage for all analyzed circuits.

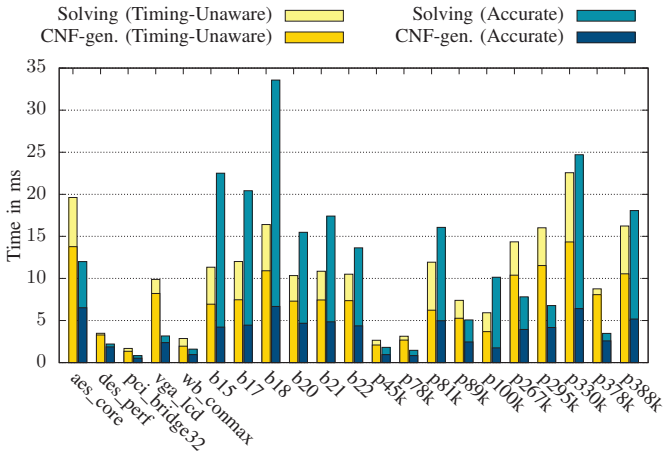

Figure 5. Runtime per fault for the formula generation and solve process.

ATPG. The remaining faults require at least one stable input and, hence, the waveform-accurate ATPG. Overall, our method gives fast computation times with an average test pattern generation time below 35 ms per fault.

The benefit of re-using the same formula for all TSOFs in a gate by utilizing assumptions can be clearly observed: Although the formula generation for the timing-unaware formulas is faster in absolute terms, it is slower in relation to the number of tested faults because the formulas can only be re-used for the TSOFs that do not require input stability (of which there are far fewer). The number of detectable faults per created formula is higher for the more complex waveform-accurate formulas. Even though learned information can be re-used in subsequent solver calls, the solving time is still higher for the waveform-accurate ATPG. However, the entailed decrease in performance is minor which can be attributed to our hybrid approach where only the justification cone is modeled in the more expensive accurate manner.

The number of aborts due to timeouts is very low (at most 0.2 %). Altogether, our algorithm scales very well even with very large circuits with more than 100,000 complex cells and normal gates.

B. Using Only Timing-Unaware ATPG

The complexity of modeling the propagation of signals throughout the justification cone in a waveform-accurate manner gives rise to questions regarding the gains of this approach compared to simply generating all test patterns without any timing information. To gauge the benefits we first created a test without any timing information for every fault with a stability requirement. We then validated whether the test pattern is invalidated through glitches by creating a waveform-accurate SAT formula for the same fault and using the test pattern as input assignment. Through additional constraints, this formula is satisfiable if the test is not invalidated by glitches, otherwise it is unsatisfiable. In order to ascertain the validity of the results, we created up to 10 test patterns for every detection pattern for every fault and checked if the tests remain valid in the presence of hazards.

In addition, we also computed how many of the remaining tests violate our strong stability condition. These patterns might be invalidated in a real circuit by slight changes in the timing due to process variations. Our findings are summarized in Figure 6.

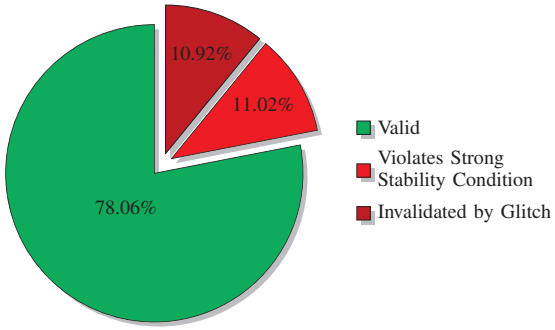

Figure 6. Portion of test patterns that are invalidated by glitches or violate our strong stability condition, averaged over all circuits.

Of all generated test patterns for each circuit, on average 10.92 % are invalidated by glitches. In addition, 11.02 % of patterns violate our strong stability condition and thus might be susceptible to invalidations by slight changes in timing.

This clearly highlights the benefits of our approach which directly generates valid test patterns. Moreover, for on average 3.22 % of testable faults all of the generated patterns were invalidated by glitches. These faults appear to be sensitive to glitches making them hard to detect without timing information even when many different patterns are created.

C. Relaxing the Pattern Conditions

Thus far, the patterns were created for a launch-on-capture test architecture. By relaxing this condition and allowing all possible flip-flop values in T_2 (for example when enhanced scan [29] is used), the fault coverage can be considerably increased (see Figure 7).

Without any restrictions on T_2 the average fault coverage increases to above 97 % . This highlights the adaptability of our SAT-based approach: The formula can simply be modified by including more constraints on signals or their propagation. Such constraints can just as easily be removed from the formula to generate a different set of results. Hence, other test architectures (e.g., launch-on-shift) could be easily integrated if this would be required by the test environment.

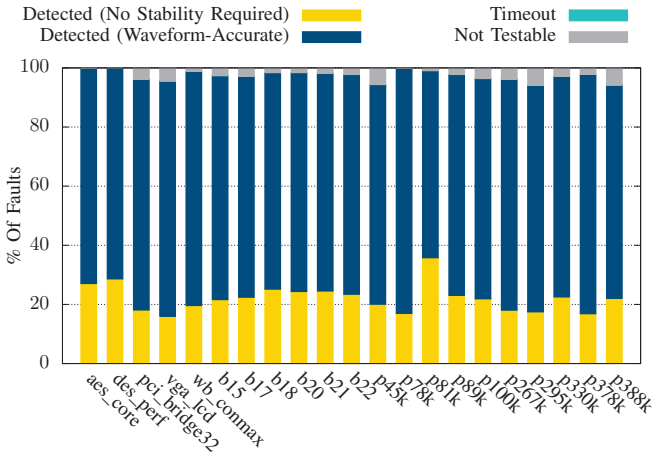

Figure 7. Fault coverage when T_2 can be chosen freely.

V. Conclusion

We introduced a novel SAT-based TSOF ATPG algorithm which generates hazard-aware test patterns that are not invalidated by glitches. Two different approaches were implemented: A timing-unaware ATPG is used for faults which do not require stability conditions. For faults that require input stability a timing-aware, waveform-accurate modeling is chosen. The accurate modeling is restricted to the justification cone of the fault, while the remainder of the circuit is handled in a timing-unaware manner, creating an efficient hybrid circuit model. The detection properties for each TSOF are automatically computed from the spice description of each gate. Hence, gate library specific implementation information is directly incorporated into the test pattern generation, ensuring that the patterns actually match the gates utilized in the circuit under test.

The results show a high performance of our implementation with average test generation times of below 35 ms per fault even for large circuits with more than 100,000 complex cells and normal gates. In addition, our experiments demonstrate the benefit of creating tests in a waveform-accurate manner, as on average 10.92 % of patterns created without timing information were invalidated by glitches.

In the future one could further improve the performance by adding a waveform-accurate TSOF simulator. This would greatly decrease the size of the test set allowing for a more thorough analysis of the patterns. Additionally, the fault coverage could be improved by utilizing tests over multiple time frames [30] which was not considered in this work. Moreover, one could analyze the effect of slight changes in the timing due to process variations in combination with a weaker stability condition. Furthermore, comparing our waveform-accurate ATPG to more pessimistic hazard-aware methods (e.g., robust tests [21]) would show the benefits of our approach even more clearly.

VI. Acknowledgements

The authors thank Matthias Sauer and Tobias Schubert from the University of Freiburg for intensive discussions on SAT solving and modeling techniques.

References

[1] F. Hapke, W. Redemund, A. Glowatz, J. Rajski, M. Reese, M. Hustava, M. Keim, J. Schloeffel, and A. Fast, "Cell-aware test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 9, pp. 1396–1409, Sept 2014.

[2] F. Hapke, M. Reese, J. Rivers, A. Over, V. Ravikumar, W. Redemund, A. Glowatz, J. Schloeffel, and J. Rajski, "Cell-aware production test results from a 32-nm notebook processor," in *IEEE International Test Conference*, Nov 2012, pp. 1–9.

[3] F. Hapke, R. Arnold, M. Beck, M. Baby, S. Straehle, J. F. Goncalves, A. Panait, R. Behr, G. Maugard, A. Prashanthi, J. Schloeffel, W. Redemund, A. Glowatz, A. Fast, and J. Rajski, "Cell-aware experiences in a high-quality automotive test suite," in *19th IEEE European Test Symposium (ETS)*, May 2014, pp. 1–6.

[4] R. L. Wadsack, "Fault modeling and logic simulation of CMOS and MOS integrated circuits," *Bell System Technical Journal*, vol. 57, no. 5, pp. 1449–1474, 1978.

[5] C. Han and A. D. Singh, "Testing cross wire opens within complex gates," in *IEEE 33rd VLSI Test Symposium (VTS)*, April 2015, pp. 1–6.

[6] X. Lin and S. M. Reddy, "On generating high quality tests based on cell functions," in *IEEE International Test Conference (ITC)*, Oct 2015, pp. 1–9.

[7] H. Cox and J. Rajski, "Stuck-open and transition fault testing in CMOS complex gates," in *International Test Conference, 1988*, Sep 1988, pp. 688–694.

[8] S. M. Reddy, V. D. Agrawal, and S. K. Jain, "A gate level model for CMOS combinational logic circuits with application to fault detection," in *21st Conference on Design Automation*, June 1984, pp. 504–509.

[9] Y. M. Elzig, "Automatic test generation for stuck-open faults in CMOS VLSI," in *18th Conference on Design Automation*, June 1981, pp. 347–354.

[10] K.-W. Chiang and Z. G. Vranesic, "On fault detection in CMOS logic networks," in *20th Conference on Design Automation*, June 1983, pp. 50–56.

[11] R. Chandramouli, "On testing stuck-open faults," in *Symposium on Fault-Tolerant Computing*, 1983.

[12] N. Devtaprasanna, A. Gunda, P. Krishnamurthy, S. M. Reddy, and I. Pomeranz, "A unified method to detect transistor stuck-open faults and transition delay faults," in *Eleventh IEEE European Test Symposium (ETS'06)*, May 2006, pp. 185–192.

[13] C. Han and A. D. Singh, "Hazard initialized loc tests for tdf undetectable CMOS open defects," in *22nd Asian Test Symposium*, Nov 2013, pp. 189–194.

[14] ——, "Improving CMOS open defect coverage using hazard activated tests," in *IEEE 32nd VLSI Test Symposium (VTS)*, April 2014, pp. 1–6.

[15] S. K. Jain and V. D. Agrawal, "Test generation for MOS circuits using D-algorithm," in *Proceedings of the 20th Design Automation Conference*, ser. DAC '83. Piscataway, NJ, USA: IEEE Press, 1983, pp. 64–70.

[16] S. M. Reddy, M. K. Reddy, and J. G. Kuhl, "On testable design for CMOS logic circuits." in *Proceedings of the 1983 International Test Conference*, 1983, pp. 435–445.

[17] S. M. Reddy and M. K. Reddy, "Testable realizations for FET stuck-open faults in CMOS combinational logic circuits," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 742–754, Aug 1986.

[18] D. L. Liu and E. J. McCluskey, "Designing CMOS circuits for switch-level testability," *IEEE Design Test of Computers*, vol. 4, no. 4, pp. 42–49, Aug 1987.

[19] S. Kundu and S. M. Reddy, "On the design of robust testable CMOS combinational logic circuits," in *Fault-Tolerant Computing, FTCS-18*, June 1988, pp. 220–225.

[20] S. Chakravarty, "A testable realization of CMOS combinational circuits," in *International Test Conference*, Aug 1989, pp. 509–518.

[21] S. M. Reddy, M. K. Reddy, and V. Agrawal, "Robust tests for stuck-open faults in CMOS combinational logic circuits," in *Fault-Tolerant Computing, FTCS-14*, 1984, pp. 44–49.

[22] M. Yoeli and S. Rinon, "Application of ternary algebra to the study of static hazards," *J. ACM*, vol. 11, no. 1, pp. 84–97, Jan. 1964. [Online]. Available: http://doi.acm.org/10.1145/321203.321214

[23] Si2, "NanGate FreePDK45 generic open cell library, v1.3," http://www.si2.org/openeda.si2.org/projects/nangatelib.

[24] G. Tseitin, "On the Complexity of Derivation in Propositional Calculus," *Studies in Constructive Mathematics and Mathematical Logic*, 1968.

[25] M. Sauer, A. Czutro, I. Polian, and B. Becker, "Small-delay-fault ATPG with waveform accuracy," in *Int'l Conf. on CAD*, 2012.

[26] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Des. Test*, vol. 17, no. 3, pp. 44–53, Jul. 2000.

[27] C. Albrecht, "IWLS 2005 benchmarks," in *International Workshop on Logic Synthesis*, Jun. 2005.

[28] T. Schubert and S. Reimer, "antom," in *https://projects.informatik.uni-freiburg.de/projects/antom*, 2016.

[29] S. Dasgupta, R. G. Walther, T. W. Williams, and E. B. Eichelberger, "An enhancement to lssd and some applications of lssd in reliability, availability and serviceability," in *Symposium on Fault-Tolerant Computing*, 1981.

[30] X. Lin, S. M. Reddy, and J. Rajski, "Using boolean tests to improve detection of transistor stuck-open faults in CMOS digital logic circuits," in *28th International Conference on VLSI Design*, Jan 2015, pp. 399–404.