Efficient SAT-based Dynamic Compaction and Relaxation for Longest Sensitizable Paths

Matthias Sauer* Sven Reimer* Tobias Schubert* Ilia Polian† Bernd Becker*

* Albert-Ludwigs-Universität Freiburg
Georges-Köhler-Allee 051
79110 Freiburg, Germany
{ sauerm | reimer | schubert | becker }@informatik.uni-freiburg.de

† University of Passau
Innstraße 43
94032 Passau, Germany
ilia.polian@uni-passau.de

***Abstract*— Comprehensive coverage of small-delay faults under massive process variations is achieved when multiple paths through the fault locations are sensitized by the test pair set. Using one test pair per path may lead to impractical test set sizes and test application times due to the large number of near-critical paths in state-of-the-art circuits.**

We present a novel SAT-based dynamic test-pattern compaction and relaxation method for sensitized paths in sequential and combinational circuits. The method identifies necessary assignments for path sensitization and encodes them as a SAT-instance. An efficient implementation of a bitonic sorting network is used to find test patterns maximizing the number of simultaneously sensitized paths. The compaction is combined with an efficient lifting-based relaxation technique. An innovative implication-based path-conflict analysis is used for a fast identification of conflicting paths.

Detailed experimental results demonstrate the applicability and quality of the method for academical and industrial benchmark circuits. Compared to fault dropping the number of patterns is significantly reduced by over 85% on average while at the same time leaving more than 70% of the inputs unspecified.

I. Introduction

High-performance and low-power design methods often result in circuits with a large number of near-critical paths. By employing versions of primitive cells with different V_T, or by using body biasing techniques, designers can realize local power-performance trade-offs, choosing for each gate whether it should be slow and power-efficient or fast and power-hungry. Slow versions of gates are consequently used on short paths, such that the delays of such paths become comparable to the critical path delay. High-quality testing of small-delay faults under process variations requires sensitization of several, ideally all, near-critical paths through the fault location. Since the number of such paths is so high, using one test pair per path will imply excessive pattern count and test application time. Therefore, it is desirable to generate compact test pair sets where one pair sensitizes multiple long paths and thus detects many small-delay faults along these paths. This problem is related to test compaction.

Static compaction [1], [2] starts with existing test patterns and tries to merge them by finding common non-conflicting inputs. In contrast to static compaction, *dynamic compaction* [3], [4] yields new test patterns that target several faults at the same time. Often, these methods are combined with *fault dropping* [5], where simulation is used to detect random fault detection. Previous work on test pattern compaction [6], [7] for stuck-at-faults, which use SAT-solvers [8], creates SAT-instances that iteratively target additional faults individually.

The compaction of delay fault models is especially challenging, as multiple timeframes and stringent test requirements need to be considered. Approaches in [9], [10] keep a pool of paths and try to generate test patterns that sensitize several paths from the pool together by combining a structural compatibility analysis with ATPG. The procedure in [11] aims at selecting paths with matching crosspoints to yield a compact test set with high fault coverage.

A central requirement for any practical test strategy is compatibility with state-of-the-art *test compression* methods [12], [13], [14]. Encoded test data is transmitted from the tester to the chip where it is decompressed by on-chip decoders and fed into the scan chains. The efficiency of test compression heavily depends on a large number of don't-care values (Xes) being present in the test data. Unfortunately, compaction tends to eliminate many Xes, resulting in test sets that are smaller but much harder to compress. Test relaxation methods that introduce Xes at positions not essential for detection [15], [16] are needed to combine compaction with test compression.

We present a small delay fault oriented dynamic compaction method that compacts long sensitizable paths. The method generates a SAT-instance that encodes the circuit functionality and sensitization rules for a group of target paths. By maximizing the number of paths that are sensitized by a single test pattern pair, highly compacted test sets are generated. In addition, efficient SAT-based test pattern relaxation is used to generate high-quality test cubes. Extensive experimental results on academical and industrial benchmark circuits demonstrate the effectiveness of the approach.

The remainder of the paper is structured as follows. Section II explains the preliminary work. An overview of the method is given in Section III. The details of our approach are explained in Section IV. Experimental results are reported in Section V and Section VI concludes the paper.

II. Preliminaries

A. SAT

Given a propositional formula ϕ, an *assignment* A is a function $A : \mathcal{V} \rightarrow \{0,1\}$, where $\mathcal{V}$ is the set of Boolean variables, which occur in ϕ. The *SAT problem* is looking for an assignment A for ϕ, such that ϕ is satisfied. If such an assignment exists we say A is a *model* for ϕ. Usually, the formula ϕ is given in conjunctive normal form (*CNF*), which is a conjunction of disjunctions of literals. A *literal*

978-3-9815370-0-0/DATE13/©2013 EDAA

Table I
Sensitization conditions for AND/NAND gates

Type	If on-path transition is	Off-path inputs are set to	Additional conditions
strong non-robust	$0 \rightarrow 1$	U1	–
	$1 \rightarrow 0$	H1	–
restricted functional	$0 \rightarrow 1$	U1	transition at gate output
	$1 \rightarrow 0$	XX	

is a variable v or its negation $\neg v$. A disjunction of literals is also called a *clause*, which is often written as a set of literals. A circuit can be transformed into a CNF by using *Tseitin-encoding* [17], which uses separate variables for each circuit line. The resulting CNF size is linear in the circuit size.

A clause c_e is *empty*, if all literals in c_e are assigned to 0. We say a clause c *implies* a literal l, if only $l \in c$ is not assigned and all other literals $l' \in c$ are assigned to 0. As resulting *implication* l is assigned to 1.

Consider a directed acyclic graph $G = (V, E)$, where the vertices V indicate assignments of literals, and edges E indicate implications. Given a clause $c = \{l_1, \ldots, l_n\}$, where l_1 is an implication (and consequently $l_2, \ldots, l_n$ are assigned to 0), G contains for every vertex l_i , $i \in \{2, \ldots, n\}$ an edge to the vertex l_1. This graph is called *implication graph* and is used to generate a *reason* in case an empty clause is produced by a SAT-solver. A reason is a set of variables $\mathcal{V}_R \in \mathcal{V}$, whose assignment is responsible for the empty clause. In general, there are many reasons for one empty clause.

Modern SAT-solvers are based on the DPLL [18] algorithm. They basically execute a loop consisting of (1) making a *decision*, (2) *propagating* the decision (processing all implications forced by the current decision), and (3) if a conflict occurred, *resolving* the conflict by determining the *reason* for the empty clause. After unassigning all *wrong* variables the search process continues until a model has been found or the instance has been proven unsatisfiable (unresolvable conflict).

B. Path Sensitization

A path p is defined by a sequence of gates $g_1, \ldots, g_k$, such that the input of g_j is driven by the output g_{j-1} for all $1 < j \leq k$. Intuitively, a path p is *sensitized* by a test pair (v_1, v_2) if a transition at its input g_1 propagates to its output g_k, thus exposing delays along the path. Path-oriented ATPGs [19], [20] generate test pairs for a delay fault on gate g that sensitize a number of longest paths through g. If the additional delay introduced by a fault is greater than the slack of one of these paths, the corresponding test pair is likely to detect the fault. Hence, the excitation of longer paths leads generally to an improved small delay fault detection quality.

A path p is formally defined to be sensitized by a test pair (v_1, v_2) if it launches a transition at g_1 and justifies certain *sensitization conditions* on the off-path inputs of all gates of p. The sensitization conditions considered in this work are shown in Table I for AND/NAND gates and explained in detail in [21], [22]. H1 stands for a signal that stabilizes to logic 1 in both cycles after application of v_1 and v_2. U1 stands for a signal that eventually stabilizes at logic 1 in the second cycle without imposing conditions on the first cycle. Finally, XX stands for a signal on which no conditions are imposed at all. The requirements for OR/NOR-gates are defined analogously. Due to their more stringent requirements on the side inputs, strong non-robust path sensitization reduces the probability of a test invalidation by e. g., glitches and offer hence a higher test quality than restricted functionally sensitized paths.

III. Overview of the method

The main objective of our proposed dynamic compaction and relaxation method is to sensitize all given target paths P^T with as few patterns as possible and minimize the number of specified primary inputs in addition. Figure 1 gives an overview of the algorithm.

In the *path generation* phase, we use the in-house SAT-based path generation tool PHAETON. PHAETON reads a sequential circuit given as gate-level net list along the timing specification (given as SDF-file) and the fault list. For each gate g in the fault list, the longest sensitizable path through g is generated.

In a *preprocessing* phase we simplify the generated list of longest sensitizable paths. A given path may be the longest path for several of the gates. As different instances of the same path can be compacted trivially without increasing the fault coverage, we remove duplications, and just keep the *unique paths*. Furthermore, we recognized in our experiments that paths with a high probability to *conflict* with other paths (c.f. IV-A), are also harder to compact and should therefore be considered early in the compaction process. Therefore we determine the number of conflicting paths for each path (c.f. IV-B). Hereupon the paths are *sorted* by this number, resulting in an ordered list of *target paths* P^T.

In the *dynamic compaction* phase we utilize modern SAT-solver techniques to generate test patterns that maximize the number of target paths sensitized at the same time.

First, we perform a *path selection*, where a list of *required paths* $P^R \subseteq P^T$, which we want to sensitize, and a list of *candidate paths* $P^C \subseteq P^T$, which should be sensitized in addition to the required paths P^R, are extracted. The number of candidates is limited by a user-defined value $c_{\max}$, which serves as a trade-off between quality and performance. Initially, we pick the first uncompacted path in the ordered path list P^T as the required path in P^R, and choose the next $c_{\max}$ paths in P^T as candidate paths P^C.

In the *path compaction* phase (c.f. IV-C), a SAT-instance is generated, which is satisfied iff 1) all paths in P^R can be sensitized at once and 2) k paths out of P^C can be sensitized in addition. As a greedy heuristic, we maximize the number of candidate paths k using incremental SAT-solving. In the manner of a binary search, we solve the instance with the same set of paths but with different values for k until the maximum value for k has been found. If an instance is satisfiable, a test pattern A_{TP} can be extracted from the model returned by the SAT-solver. This pattern sensitizes all required paths and k candidate paths.

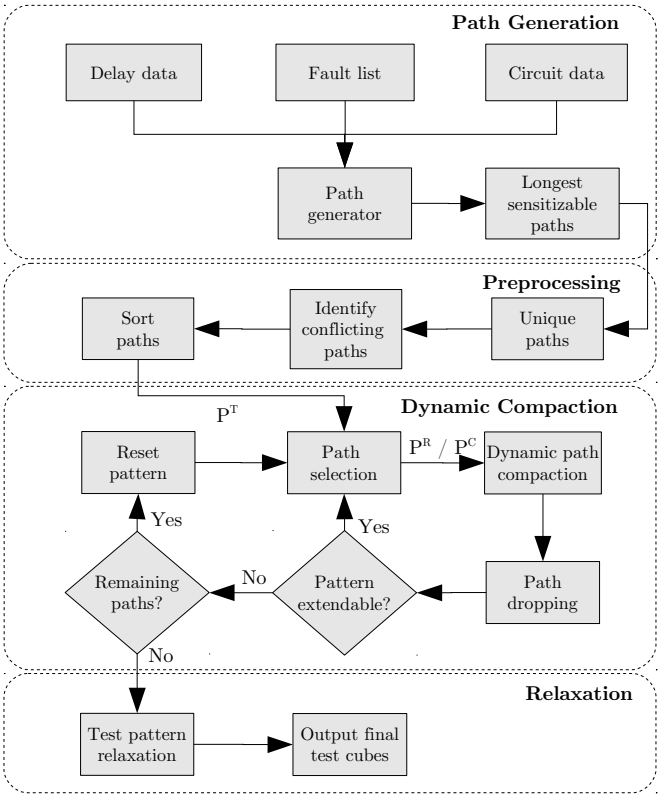

Figure 1. Dynamic compaction and relaxation flow

In addition, a forward-looking simulation-based *path dropping* is performed to identify paths that are not in P^C but still sensitized by the returned test pattern by chance.

If there are remaining target paths P^T that have not been tried to merge into the current test pattern, the current pattern may be *extendable*. Therefore, we extend the list of required paths by the newly sensitized candidate paths $P^R = P^R \bigcup P^C$ and select new candidate paths P^C from P^T. As before, we create the corresponding SAT-instance and maximize the number of newly compacted paths. This step is repeated until each path in P^T is conflicting with at least one path in P^R, i. e. P^R can not be extended anymore. Note that the test compaction is dynamic and hence the pattern may change in every step of this procedure.

If the pattern is not extendable anymore, the compacted paths are removed from P^T and the current test pattern is marked as representative for these compacted paths. If P^T is not empty, the pattern is *reset* for the following steps and we start over with the path selection, choosing a new required path and c_{new} candidate paths from P^T. This loop repeats until all initial target paths are covered by at least one test pattern.

In order to increase the compatibility of the generated test patterns with on-chip compaction techniques like LSFR-based test generation in a BIST environment, we apply *test pattern relaxation* (c.f. IV-D) for each compacted test pattern. That way, we obtain test cubes that are guaranteed to sensitize all required paths. Hence, the method is highly flexible and can be easily combined with refilling techniques to optimize secondary objectives.

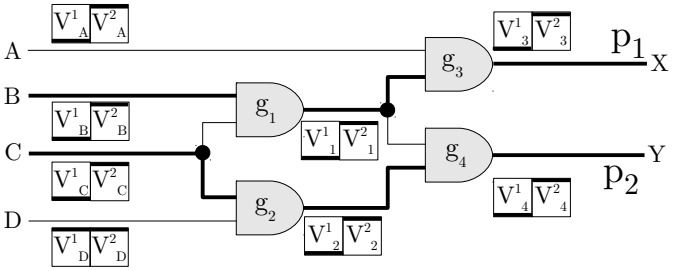

Figure 2. Path sensitization example

IV. Details of the method

A. Encoding of path sensitization

Figure 2 shows an example circuit instance consisting of the inputs $A, \ldots, D$, AND-gates $g_1, \ldots, g_4$ and the outputs X, Y.

Path p_1, given by a rising transition along the gates B, g_1, g_3, X and path p_2 that is defined by a rising transition at C, g_2, g_4, Y are indicated by thick lines. For each line in the example, the logic value upon application of the first and the second test pattern is given by thick lines inside the boxes. Although both paths are not directly connected, they are partially influenced by the same signals, e. g., the output of gate g_1. Despite this fact, it is possible to sensitize both paths at the same time with the given test pattern.

For each path p we identify the set of *sensitization logic constraints* $L^p = l_1^p, \ldots, l_n^p$ that need to hold in order to guarantee sensitization of p under the selected sensitization conditions, i. e. strong-non-robust or restricted functional sensitization. Based on this logic constraints, a decision variable s^p indicates whether a path p is sensitized as defined as follows:

$$s^p \Leftrightarrow l_1^p \wedge \ldots \wedge l_n^p \tag{1}$$

A list of paths $P = p_1, \ldots, p_n$ is *compatible* with each other, if a test pattern pair exists, such that each path's logic constraint can be met at the same time, i. e.

$$s^P = \left(s^{p_1} \wedge \ldots \wedge s^{p_n}\right) \tag{2}$$

does hold. Otherwise, the paths in P are *conflicting*.

B. Implication-based path-conflict analysis

The implication-based path-conflict analysis identifies conflicting paths very efficiently employing the infrastructure of the used SAT-solver.

At first, a general SAT-instance $\xi(E, s^P)$ is generated, that encodes the logic of the circuit for two timeframes (E). In addition, for each path $p_i \in P$ we encode s^{p_i} according to Equation 1. Based on this general instance, we pass the SAT-solver a path s^{p_j} variable as an assumption. The solver can identify the implications of requiring s^{p_j} to hold and therefore can identify which of the s^{p_i} variables are implied to be 0. Hence, we can identify the paths that are conflicting with p_j directly by looking at implied variables in the SAT-solver.

In addition to the direct implications on the logic constraints, the solver can identify more complex implications even on related lines. For example, if a gate input has a controlling input constraint, the output of the gate can be determined.

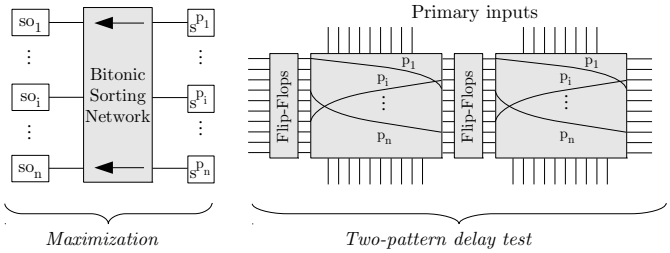

Figure 3. Illustration of SAT-instance generation

Likewise, if a gate output has a non-controlling output constraint, each input must be set accordingly.

Although our implication-based path-conflict analysis may miss some conflicting paths that require a more detailed analysis, the method is very fast and efficient. Hence, it is perfectly suited for a runtime efficient preprocessing step to identify initial conflicts.

C. Path compaction

The main path compaction step is based on the generation of the SAT-instance $\phi(E, P^R, P^C, k)$ where E is the circuit encoding of the two timeframes. $P^R = p_1^R, \ldots, p_m^R$ is a list of paths that are required to be sensitized by the resulting test pattern and $P^C = p_1^C, \ldots, p_n^C$ gives a list of path candidates out of which at least k paths should be sensitized in addition to the required paths.

Figure 3 shows the details of the instance generation. For each path in the list of required paths P^R and candidate paths P^C, the decision variables $s^{p_i^R}$ and $s^{p_i^C}$ are encoded as defined in Equation 1. An addition we are forcing the required paths to be sensitizable by encoding Equation 2 for P^R and force s^{P^R} to hold.

In order to maximize the number of paths that are sensitized at the same time, the number of $s^{p_i^C}$ variables set to 1 needs to be maximized. This is achieved by the encoding a bitonic sorting network [23] directly into the SAT-instance. The bitonic sorting algorithm is comparison based and has a complexity of $O(n \cdot \log\ n)$. Note that using such an unary sorting network based encoding is much more efficient than a binary representation. In addition, the bitonic sorting network is especially suited for implementation in a SAT-instance, as the sorting-flow is relatively independent on the sorted data. By sorting the variables s^{P^C} an ordered list of the decision variables $so_1^{P^C}, \ldots, so_n^{P^C}$ is defined that has the form $1 \ldots 10 \ldots 0$. Hence, if $so_i^{P^C}$ is set to 1, there are at least i paths sensitized simultaneously. Note that the ordered decision variables $so_i^{P^C}$ represent the number of sensitized paths, but do not correspond to the sensitization of a specific path p_i.

By requiring a certain $so_i^{P^C}$-variable to be set to 1, the SAT-solver is forced to return a solution that will sensitize at least i paths. If no such assignment exists, the instance is classified as unsatisfiable. By the means of a binary search over the number of paths, the maximal number k is computed such that $\phi(E, P^R, P^C, k)$ is satisfied but $\phi(E, P^R, P^C, k{+}1)$ is not. Hence, such a k represents the maximal number of paths that can be sensitizable simultaneously together with the required paths. Based on the returned model of the SAT-solver, the test pattern A_{TP} is obtained by extracting the logic values of the inputs. Furthermore, the sensitized paths are identified as their $s^{p_i^C}$-variables are set to 1.

As described in section IV-B modern SAT-solvers are tuned to detect implications very efficiently. Therefore many of the lines are set to defined values, big parts of the circuit are decided, and consequently the level of complexity is reduced.

D. Test pattern relaxation

The goal of *test pattern relaxation* is to minimize the specified primary inputs for a test pattern. Our relaxation is based on a static SAT-based method *lifting* [24].

For each obtained test pattern A_{TP} from the main compaction phase a new SAT-instance $\psi(E, P^R, A_{TP}) = (E \wedge A_{TP}) \Rightarrow P^R$ is built, where E is again the encoding of two timeframes, and P^R are the required paths (*requirement*), which are detected by the previous compaction step to be sensitizable with the test pattern A_{TP}. Therefore we can ensure that $\psi(E, P^R, A_{TP})$ is trivially satisfiable.

In order to find a minimal test cube, we have to identify a minimal subset $A_{TC} \subseteq A_{TP}$, such that $\psi(E, P^R, A_{TC})$ is still satisfiable. Intuitively we are looking for a sufficient set of assignments, which satisfies the requirement.

Due to SAT-solver specific properties, the requirement P^R is substituted by its negation, i. e. $\psi(E, \neg P^R, A_{TP}) = (E \wedge A_{TP}) \Rightarrow \neg P^R$. Based on this substitution, we have to identify a minimal subset $A_{TC} \subseteq A_{TP}$ such that $\psi(E, \neg P^R, A_{TC})$ is *not* satisfied. For more details of this modification the reader is referred to [24].

Based on this problem statement, there are several lifting variants, which differs in terms of test cube quality and runtimes. For our experiments we use a variant of *implication-graph*-lifting, which is based on the implication graph, produced by a SAT-solver. By definition $\psi(E, \neg P^R, A_{TP})$ is unsatisfiable, since the test pattern satisfies the requirement, therefore the pattern violates the negated requirement. A SAT-solver will produce an empty clause and a reason by traversing the implication graph backwards. The solver is able to produce several reasons from which we pick a reason consisting only input variables. Then we can ensure that all input variables, which are not part of this reason, are not responsible for the empty clause, i. e. are not part of the received test cube A_{TC}.

Other lifting techniques are more effective, but also more expensive. Since we have many solver calls (one for each test pattern), we choose this less expensive relaxation variant. In an earlier work [25], we compared our method to other relaxation variants and observed that the quality loss is very reasonable. The implication-graph-lifting profoundly depends on the initial input pattern, the order in which the root assignments are added to the SAT-instance and the implication graph built by the SAT-solver. The initial pattern is given by the compaction and the generated implication graph depends on the used SAT-solver. For the order of assignment we investigated several heuristics, depending on fanouts and the close environment of an input.

Table II
COMPACTION OF LONGEST STRONG NON-ROBUST PATHS

	Number of patterns					Time [s]	
Circuit	Init	Uniq.	Comp.	FD	X%	Comp.	Relax.
s05378	5090	2122	247	1521	65.07	2.88	0.57
s09234	8887	2110	317	1347	73.34	3.53	1.20
s13207	13190	2850	611	1843	93.88	4.34	3.42
s15850	15292	3566	608	2298	86.50	5.72	4.13
s35932	29026	9753	40	6436	55.96	35.88	3.56
s38417	37465	12121	398	8768	76.09	55.36	10.36
s38584	28074	11451	235	7118	80.94	30.95	4.51
b14	7987	5477	2194	4513	55.15	35.45	27.93
b15	10492	6706	2462	4779	81.73	95.74	41.23
b17	32842	21346	2956	15417	83.68	944.64	147.48
b20	18553	12714	4437	11014	56.81	1104.30	100.73
b21	18457	12538	4811	10776	53.52	1036.72	109.98
b22	27023	19186	5398	16514	60.23	3946.82	174.93
p35k-s	74963	48082	6956	41194	73.10	22457.40	826.22
p45k-s	73231	29437	3655	25131	87.78	2873.19	170.09
p78k-s	147550	82850	778	73103	38.72	23506.90	321.78

Table III
APPLICATION IN TRANSITION FAULT MODE

	Number of patterns			Test quality			Time [s]	
Circuit	Init	Uniq.	Comp.	FC%	PL%	X%	Comp.	Relax.
cs05378	5450	1527	90	98.06	96.98	76.10	0.54	0.18
cs09234	11018	2297	145	98.43	97.00	75.99	1.78	0.53
cs13207	15766	3490	595	99.14	97.05	96.18	2.29	2.55
cs15850	19234	3912	297	98.41	95.11	90.06	2.55	1.46
cs35932	29570	9700	26	92.03	93.71	54.85	34.81	2.56
cs38417	44166	9832	88	99.57	95.76	81.39	11.28	2.66
cs38584	35572	13387	304	92.38	94.44	93.04	19.58	3.52
b14c	10690	4632	800	99.96	96.53	67.64	14.15	4.94
b15c	14008	6441	2073	99.74	94.50	91.45	43.46	18.33
b17c	45378	22151	1896	99.70	94.99	90.68	393.03	64.34
b20c	23900	10116	707	99.94	96.73	63.88	87.67	14.36
b21c	24260	10295	693	99.97	96.74	56.67	94.47	15.08
b22c	34638	14680	720	99.94	97.00	64.25	273.34	25.23
p35k	87122	25216	664	99.49	95.13	76.19	695.62	82.81
p45k	79542	23149	2072	99.96	93.24	98.27	295.98	98.82
p78k	148486	47645	237	100.00	95.81	45.28	3437.34	97.81
p81k	176460	60870	351	99.42	95.31	46.08	1208.61	90.19
p89k	166772	50782	444	99.90	93.66	87.41	963.68	71.14

V. EXPERIMENTAL RESULTS

The method is applied to the sequential versions of ISCAS 89 and ITC 99 benchmark circuits and industrial circuits provided by NXP in two experimental setups. All measurements were performed on an Intel Xeon computer running at 3.3 GHz with the runtimes listed in seconds.

As SAT-solving back-end we chose a single-threaded version of the in-house SAT-solver *antom* [26] which supports efficient incremental SAT-solving with and without assumptions. We adjusted several parameters of the SAT-solver so as to make it fit our problem, which is mostly characterized by a very large number of rather easy-to-solve SAT-instances.

In the first experiment, we generated the longest rising and falling non-robust path through each gate using our in-house timing-aware ATPG-tool *PHAETON* [27], [28] with strong non-robust path sensitization. Hence, the generated paths are highly suitable for a small-delay fault oriented test for both, slow-to-rise and slow-to-fall faults. We applied our presented compaction and relaxation method to these paths and list the results in Table II. The maximal number of paths selected for the maximization step $c_{\max}$ was set to 50, which tends to be the optimum value in terms of quality and efficiency in our experiments. The first column gives the circuit name. The next column lists the number of initial paths which corresponds to the uncompacted number of test patterns. Column 3 lists the number of unique paths. The number of test patterns after the compaction are listed in the "Comp." column. For comparison, we implemented a forward-looking fault dropping based compaction which also subsumes as static compaction. That results are given in the column "FD". The "X%" column shows the number of X inputs after the application of the relaxation. The last two columns list the runtimes needed for the application of the compaction step and the relaxation step respectively.

As can be seen, many of the longest paths are shared resulting in a much lower number of unique paths. The compaction method succeeds in reducing the numbers of needed test patterns significantly for each of the circuits. On average, the number of needed test patterns is reduced by more than 93% and more than 87% compared to the number of initial and unique paths respectively. Our method outperforms fault dropping by 85%. Despite this strong compaction, our relaxation step is still able to generate highly unspecified test cubes. On average, over 70% of the inputs can be relaxed. The application of a static compaction step after the relaxation does not improve the compaction which can be seen as evidence for the high quality of the generated test cubes. In addition, the runtimes needed for compaction and relaxation were rather low, esp. when compared to the time needed to generate the initial set of paths. Hence, our proposed method is efficient and effective at the same time.

In a second experiment, we targeted the generation of a compact test set for *transition delay faults* through long sensitizable paths.

In contrast to the previous experiment, we skipped the computation of the longest path through a gate, if the path-generator already found a path through that gate while targeting a different gate. Hence, the number of unique paths is reduced. The test set covers the complete circuit using rather long paths but not necessarily the longest paths. In order to achieve high fault coverages obtainable by an enhanced full scan design, we applied this mode to the combinational cores of the reported benchmark circuits. Furthermore, we generated rising and falling restricted-functionally-sensitizable paths for this experiment. Such paths have weaker sensitization-conditions compared to strong-non-robust paths but still guarantee the detection of a transition delay fault.

The results are given in Table III. The columns containing the number of patterns are listed like in the previous experiment. The columns 5 to 7 list measurements of the test quality. The transition delay fault coverage is given in the "FC%" column. The column "PL%" gives the average percentage of the sensitized path lengths compared to the maximal sensitizable path length in percent. The number of unspecified inputs is shown in the column "X%". The

runtimes needed for compaction and relaxation are given in the last two columns.

The application of the transition delay mode leads to an even larger reduction in the number of test patterns as the initial number of unique paths is reduced compared to the previous experiment. On average, the number of needed test patterns in this mode is reduced by more than 98% and more than 95% compared to the number of initial and unique paths respectively. However, the obtained test set still achieves a comprehensive transition delay fault coverage (more than 98% on average) as each gate is guaranteed to be sensitized by a rising and falling transition. The average length of the found path is 95% of the provably longest sensitizable path through the same gate and therefore only a limited amount of small-delay-fault coverage is lost. Compared to the sensitization of the longest paths, this mode achieves an additional pattern count reduction of about 30%.

A fair comparison with other path-compaction results is difficult as the exact experimental setup is not comparable. Identical timing specifications are needed to generate the same longest paths. In addition, restrictions like considerations of low-cost tester capabilities in [9] lead to limitations that are easily quantifiable. Compared to the compacted test-set sizes for transition delay faults reported in [29] our method achieves a better compaction while at the same time offering an improved small-delay fault coverage. In addition, our method yield better pattern counts than the stuck-at-fault test sizes reported in [3] for some circuits like cs38417.

Hence, our proposed compaction and relaxation flow offers a high delay fault quality with a pattern count comparable or better than the numbers reported in previous works. In addition, it is suitable for large industrial circuits.

VI. Conclusions

We presented a novel SAT-based dynamic compaction method able to handle well defined sensitized paths in sequential and combinational circuits. The method identifies necessary assignments for path sensitization and encodes them as a SAT-instance. An efficient implementation of a bitonic sorting network is used to find test patterns maximizing the number of simultaneously sensitized paths. Detailed experimental results demonstrate the applicability and effectivity of the method on academic and industrial benchmarks. Compared to fault dropping the number of patterns was significantly reduced by over 85% on average. Additionally our relaxation methods results in over 70% of X-inputs on average.

In future, we want to extend the method to include secondary objectives, e. g., the energy usage of the test patterns.

Acknowledgement

Parts of this work were supported by the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG) under grants BE 1176-15/2, PO 1220-2/2, GRK 1103 and SFB/TR 14 AVACS.

We thank J. Schlöffel (Mentor Graphics Hamburg, formerly NXP) for supplying industrial benchmarks.

References

[1] R. Sankaralingam, R. Oruganti and N. Touba, "Static Compaction Techniques to Control Scan Vector Power Dissipation," in *VLSI Test Symposium, 2000. Proceedings. 18th IEEE*, pp. 35 –40, 2000.

[2] I. Pomeranz and S. Reddy, "On Static Compaction of Test Sequences for Synchronous Sequential Circuits," in *Proceedings of the 33rd annual Design Automation Conference*, DAC '96, pp. 215–220, 1996.

[3] I. Pomeranz, L. Reddy and S. Reddy, "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 12, pp. 1040 –1049, jul 1993.

[4] E. Rudnick and J. Patel, "Efficient Techniques for Dynamic Test Sequence Compaction," *Computers, IEEE Transactions on*, vol. 48, no. 3, pp. 323 –330, 1999.

[5] I. Pomeranz and S. Reddy, "Forward-Looking Fault Simulation for Improved Static Compaction," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 20, no. 10, pp. 1262 –1265, 2001.

[6] A. Czutro, I. Polian, P. Engelke, S. Reddy and B. Becker, "Dynamic Compaction in SAT-Based ATPG," in *Asian Test Symposium, 2009. ATS '09.*, pp. 187 –190, 2009.

[7] S. Eggersgluss, R. Krenz-Baath, A. Glowatz, F. Hapke and R. Drechsler, "A New SAT-based ATPG for Generating Highly Compacted Test Sets," in *Design and Diagnostics of Electronic Circuits Systems (DDECS), IEEE International Symposium on*, pp. 230 –235, 2012.

[8] A. Biere, M.J.H. Heule, H. van Maaren and T. Walsh, eds., *Handbook of Satisfiability*, vol. 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, February 2009.

[9] Z. Wang and D. Walker, "Dynamic Compaction for High Quality Delay Test," in *VTS 2008*, pp. 243 –248, 2008.

[10] J. Saxena and D. Pradhan, "A Method to Derive Compact Test Sets for Path Delay Faults in Combinational Circuits," in *Test Conference, 1993. Proceedings., International*, pp. 724 –733, 1993.

[11] M. Fukunaga, S. Kajihara, X. Wen, T. Maeda, S. Hamada and Y. Sato, "A Dynamic Test Compaction Procedure for High-quality Path Delay Testing," in *Design Automation, 2006. Asia and South Pacific, Conference on*, p. 6 pp., 2006.

[12] B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs," pp. 237–242, 1991.

[13] F. Hsu, K. Butler and J. Patel, "A Case Study on the Implementation of the Illinois Scan Architecture," in *Test Conference, 2001. Proceedings. International*, pp. 538 –547, 2001.

[14] J. Rajski, J. Tyszer, M. Kassab and N. Mukherjee, "Embedded Deterministic Test," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 23, no. 5, pp. 776 – 792, 2004.

[15] P.F. Flores, H.C. Neto and J.a.P. Marques-Silva, "An exact solution to the minimum size test pattern problem," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 6, no. 4, pp. 629–644, 2001.

[16] I. Pomeranz, "Computing Two-Pattern Test Cubes for Transition Path Delay Faults," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –11, 2012.

[17] G.S. Tseitin, "On the Complexity of Derivations in Propositional Calculus," in *Studies in Constructive Mathematics and Mathematical Logics* (A. Slisenko, ed.), 1968.

[18] M. Davis, G. Logemann and D. Loveland, "A Machine Program for Theorem Proving," *Communications of the ACM*, vol. 5, pp. 394–397, 1962.

[19] W. Qiu, J. Wang, D. Walker, D. Reddy, X. Lu, Z. Li, W. Shi and H. Balachandran, "K longest paths per gate (KLPG) test generation for scan-based sequential circuits," in *Test Conference, 2004. Proceedings. ITC 2004. International*, pp. 223 – 231, 2004.

[20] Z. He, T. Lv, H. Li and X. Li, "An Efficient Algorithm for Finding a Universal Set of Testable Long Paths," in *Test Symposium (ATS), 2010 19th IEEE Asian*, pp. 319 –324, 2010.

[21] N.K. Jha and S.K. Gupta, *Testing of Digital Systems*. Cambridge University Press, 2003.

[22] S. Reddy, *Models in Hardware Testing*, ch. 3. Springer, 2010.

[23] K.E. Batcher, "Sorting networks and their applications," in *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, AFIPS '68 (Spring), pp. 307–314, 1968.

[24] K. Ravi and F. Somenzi, "Minimal assignments for bounded model checking," in *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 2988, pp. 31–45, Springer, 2004.

[25] M. Sauer, S. Reimer, I. Polian, T. Schubert and B. Becker, "Provably Optimal Test Cube Generation using Quantified Boolean Formula Solving," in *Design Automation, 2013. Asia and South Pacific, Conference on*, 2013.

[26] T. Schubert, M. Lewis and B. Becker, "antom — Solver Description," in *SAT Race*, 2010.

[27] M. Sauer, A. Czutro, T. Schubert, S. Hillebrecht, I. Polian and B. Becker, "SAT-based Analysis of Sensitisable Paths," in *IEEE Design and Diagnostics of Electronic Circuits and Systems*, pp. 93–98, April 2011.

[28] M. Sauer, J. Jiang, A. Czutro, I. Polian and B. Becker, "Efficient SAT-Based Search for Longest Sensitisable Paths," in *Asian Test Symp.*, November 2011.

[29] I. Hamzaoglu and J. Patel, "Compact Two-Pattern Test Set Generation for Combinational and Full Scan Circuits," in *Test Conference, 1998. Proceedings., International*, pp. 944 –953, 1998.