Functional Test of Small-Delay Faults using SAT and Craig Interpolation

Matthias Sauer* Stefan Kupferschmid* Alexander Czutro* Ilia Polian† Sudhakar Reddy‡ Bernd Becker*

* Albert-Ludwigs-University Freiburg
Georges-Köhler-Allee 051
79110 Freiburg, Germany
{sauerm|skupfers|aczutro|becker}
@informatik.uni-freiburg.de

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

‡ University of Iowa
5324 Seamans Center
Iowa City, United States
reddy@engineering.uiowa.edu

***Abstract*—We present SATSEQ, a timing-aware ATPG system for small-delay faults in non-scan circuits. The tool identifies the longest paths suitable for functional fault propagation and generates the shortest possible sub-sequences per fault. Based on advanced model-checking techniques, SATSEQ provides detection of small-delay faults through the longest functional paths. All test sequences start at the circuit's initial state; therefore, overtesting is avoided. Moreover, potential invalidation of the fault detection is taken into account. Experimental results show high detection and better performance than scan testing in terms of test application time and overtesting-avoidance.**

I. Introduction

In the test of sequential circuits, deterministic sequential test pattern generation has long been considered too hard and hence has not been employed in most practical applications. Thus, despite the hardware overhead introduced by scan-based techniques, these have become the standard methodology.

However, scan-based techniques bear more drawbacks than just the hardware overhead introduced by the use of scan chains. For instance, scan-based testing in general is expensive in power consumption due to the large number of shift operations, which result in excessive circuit activity during the scan-in and scan-out phases. In fact, the application of scan patterns may consume up to thirty times as much power as the functional mode's peak power, which can lead to damage or to yield loss due to power droop that would not occur in functional mode [1]. Peak power and toggling rates can be reduced by using gated scan cells [2], thus precluding switching activity during scan shifting phases, by partitioning the scan chains [3], or by using dedicated low-power scan operations [4]; yet these techniques either lead to increased hardware costs or increase test application time.

Furthermore, power consumption is an issue not only during test application, but also in functional mode, as the test circuitry (e.g. scan flip-flops – these are bigger and draw more power than normal flip-flops) remains active after the testing has taken place. Additionally the non-functional scan enable signal needs to be routed to all scan flip-flops requiring considerable routing area. These issues are becoming critical in extra low power designs such as implantable medical devices.

The gain in importance of security concerns [5], [6], especially in the context of cryptochips [7], has identified the security threat posed by the scan infrastructure as another serious shortcoming of scan-based test. For instance, [8] shows that scan chains can be used as a side channel to recover keys from a hardware implementation of the Data Encryption Standard. Existing countermeasures like making the scan circuitry unusable after the production test [9], [10], as well as more sophisticated protection techniques like scan-chain scrambling [11] or permission-based security [12], [13], bring about additional hardware overhead that makes them impractical for large designs in which area is a concern. Furthermore, diagnosis may be compromised by these techniques. Some approaches rely on partial scan only [14], but the security threat is still present.

For all these reasons, several authors have worked on methods that do not require scan. [15] presents a modification of the D-algorithm [16] where the forward propagation is replaced by predictions based on the so-called driveability of the primary outputs. Similarly, [17] combines elements of PODEM [18] and FAN [19] with learning techniques that allow the application of the algorithm to several time frames. [20] extends PODEM by an initial estimation of the number of required time frames, and it considers the targeted fault's effects during the initialization phase.

Later approaches adopt the use of heuristic optimization algorithms, including genetic optimization [21], [22], in order to cope with the complexity of the sequential-ATPG problem. The test generation system MIX [23] combines deterministic, state-driven and genetic-optimization-based test generation in order to achieve higher fault coverages, while the ATPG method PROPTEST [24] uses a combination of static test sequence compaction and several sequence extension techniques. In [25], a symbolic fault simulator is successfully integrated into a genetic-algorithm environment, thus producing shorter test sequences than those generated by other similar approaches. In [26], so-called spectral techniques are used. First, a set of random test patterns is generated. Then, in an iterative process, unsuitable patterns are filtered out by simulation and new patterns are constructed by transformations applied to the remaining patterns based on a mathematical analysis of their

This work has been supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, *www.avacs.org*), and under grants GRK 1103, BE 1176-14/2 and PO 1220-1/2. We thank Xiaoxin Fan (University of Iowa) for fruitful discussions about sequential ATPG.

978-1-4673-1595-1/12/$31.00 ©2012 IEEE

characteristics. In [27], this approach is enhanced by the use of a selfish-gene algorithm, i.e. a variation of genetic optimization. Similar spectral techniques based on the wavelet transform are also employed in [28].

All these works have in common that they consider only stuck-at faults. However, timing-related fault models are particularly important for the test of cryptochips, as one important type of attack consists in reducing VDD to induce such faults.

In this work, we present the timing-aware ATPG framework SATSEQ (SAT-based SEQuential test generator) for small-delay faults in non-scan circuits. To the best of our knowledge, this is the first sequential ATPG for small-delay-faults; previous approaches considered only stuck-at faults. Furthermore, SATSEQ relies solely on deterministic algorithms for path initialization and propagation, without using randomness-based or other heuristic methods. The tool benefits from the tight integration between two approaches. The first one is a SAT-based approach for the search of longest sensitizable paths (PHAETON) [29]. It relies on powerful learning techniques that have recently been successfully employed not only in test generation for hard-to-detect faults in traditional fault models [30], [31], but also in complex test-generation scenarios, even including additional optimization constraints [32], [33], [34]. The second is an advanced model checker that relies on the theory of Craig interpolation [35], [36] in order to reduce the number of time steps that need to be considered to solve a model-checking instance (*Craig Interpolation Prover* – CIP) [37].

A first straightforward combination of these two techniques was used for reachability analysis of pre-defined sensitizable paths [38]. In contrast, SATSEQ identifies the longest paths suitable for functional fault *activation* and *propagation* to an observable output. Then, connecting test sequences guaranteed to be as short as possible are generated, such that the resulting concatenated test sequence sensitizes all found paths at least once. The test application is fully functional and hence can be combined with at-speed testing and prevents over-testing.

Furthermore, we define a categorization of the possible causes that invalidate the detection of a fault. Based on this categorization, we developed the concept of *immunity*. A test sequence is defined to be immune to a certain type of detection invalidation if that kind of invalidation does not occur when applying that sequence. This concept is different from the widely-used concept of robustness [39]. As an example, in combination with robustness, the generated test sequences are guaranteed to detect the fault independently of timing variations that may occur on off-paths in any of the modeled time frames.

Detailed experimental results on circuits from the ISCAS 89 and ITC 99 benchmark suites demonstrate the applicability of the flow. Additionally, the high fault efficiency and quality of the test sequence is demonstrated.

The remainder of the paper is structured as follows. An overview of the method is given in Section II. Craig-interpolant-based model checking is briefly explained in Section III. The details of the flow are given in Section IV. Experimental results are reported in Section V. Section VI concludes the paper.

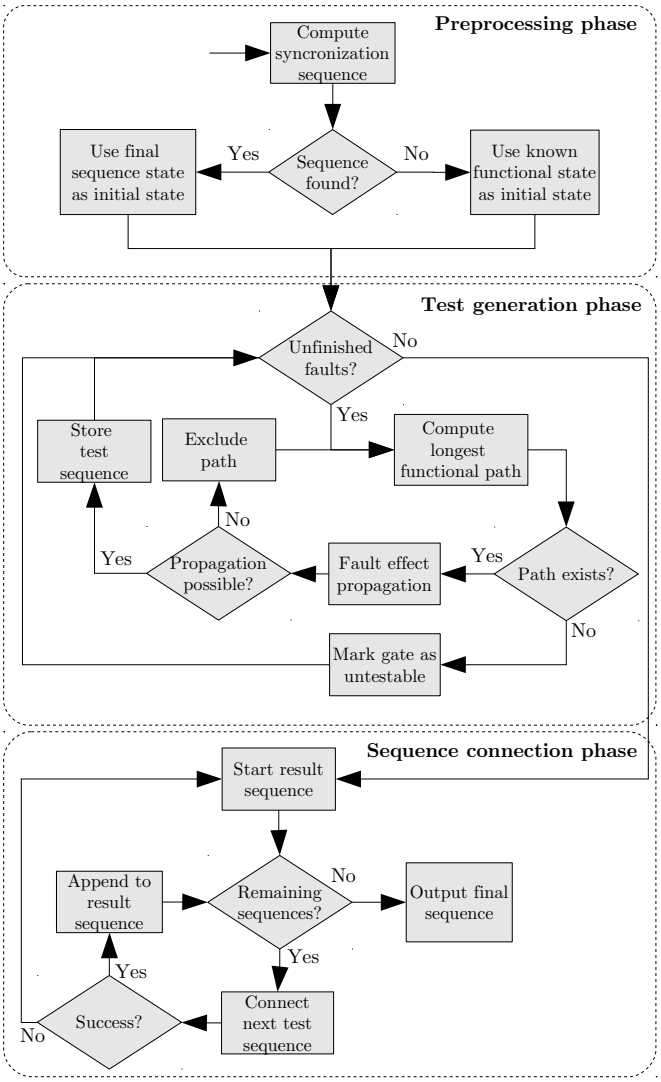

Fig. 1. General flow of "SATSEQ"

II. General flow of the method

This section provides an overview of the general flow (see Figure 1). Detailed information about the implementation of the various steps can be found in Section IV.

We consider small-delay faults in synchronous sequential digital circuits without scan. First, a synchronization sequence is calculated, i.e. a sequence that brings the circuit into a fully-specified state starting at the all-X-state. If such a synchronizing sequence can not be detected, the approach restarts to the all-0-state. All gates are assumed to have fixed pin-to-pin delays, i.e. the delays may differ for individual input-output-pairs of a gate and for rising and falling transitions. The small-delay faults are associated to the outputs of logic gates. The fault effect is assumed to be detected if a path of sufficient length (calculated as the sum of the appropriate delays of the on-path gates) is sensitized and ends at a primary output of the circuit. If such a path ends at a flip-flop, the fault effect is captured in the flip-flop and has to be propagated to a primary output in subsequent clock cycles.

978-1-4673-1595-1/12/$31.00 ©2012 IEEE

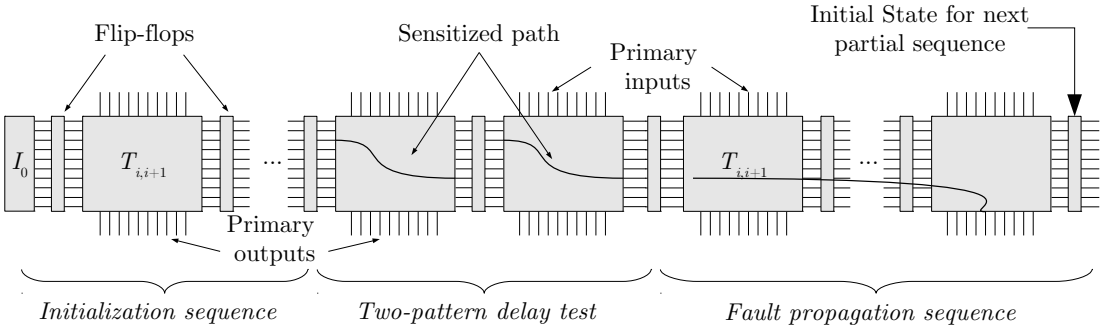

Fig. 2. Combined sequence for one fault

The test generation phase results in a test sequence that consists of several sub-sequences as displayed in Figure 2. Each sub-sequence targets a small-delay fault (but may detect further small-delay faults) and is composed of the initialization portion, the test-pair portion and the propagation portion. The initialization portion includes several primary-input vectors that justify required values in the flip-flops; the test pair sensitizes the longest path through the fault location; and the propagation portion propagates the faulty effect from the flip-flops to a primary output.

Our method generates shortest possible sub-sequence portions; if detection is possible at a primary output, no propagation portion is generated. Note that only functional states reachable from the synchronized or restart state are visited, and therefore no over-testing occurs. Also, SATSEQ is not limited to a single initialization state. If multiple initialization sequences exist, the method utilizes the initialization sequence that leads to the shortest sub-sequence. If no sub-sequence exists to detect the fault starting at the synchronized or restart state, then the fault cannot manifest itself during functional operation; hence, detecting it, e.g. using scan or other DFT mechanisms, would constitute over-testing.

We assume that only one gate in the circuit is affected by the small-delay fault. Due to the sequential nature of the test application, the presence of the fault in the circuit may invalidate the test due to one of the following three reasons:

F-invalidation—Our method explicitly sensitizes one path to one flip-flop. However, additional flip-flops may be sensitized to the fault and consequently capture the fault effect, which may invalidate the propagation portion of the sub-system and corrupt the starting state of the next sub-sequence.

I-invalidation—The fault may manifest itself during the initialization portion of the sub-sequence, resulting in corrupted values in the flip-flops during the test pair application.

P-invalidation—The fault may manifest itself during the propagation portion of the sub-sequence, thus preventing the propagation of the captured fault effect to the primary outputs.

Note that these invalidation mechanisms do not result in loss of detection in most practical cases. On the contrary, they introduce additional fault effects which amplify the fault under consideration and make it easier to observe. Nevertheless, a small possibility of test invalidation still exists. If the circuit is known not to contain static or gross-delay faults, then I- and P-invalidation could be ruled out by applying the initialization and propagation portions of each sub-sequence with sufficiently slow speed. We are also able to apply generation strategies to obtain sequences that detect the fault even in the presence of invalidation mechanisms. This is achieved by enforcing the X-value (unknown) on locations potentially affected by the invalidation. For example, one possibility to avoid F-invalidation is to enforce X on all off-path flip-flops sensitized to the fault; such sequences are called *F-immune*. Analogously, enforcing X on the fault locations during the initialization and the propagation portion, respectively, leads to sequences that tolerate I- and P-invalidation. We call such sequences *I-immune* and *P-immune*. We denote a sequence that is both F-immune and P-immune by *FP-immune*, and so on. Sequences which are immune with respect to many invalidation mechanisms are less prone to invalidation, but on the other hand some faults become untestable when the immunity is enforced using X-values.

The main objective of the third phase (sequence connection) in Fig. 1 is to combine the various sub-sequences in order to form a connected sequential test sequence. Upon the first application, the sequence will start with the initial state, i.e. either the final state of the initialization sequence or the all-0-state. Then, a connection sequence is generated that starts in the initial state and ends in the first state of the "nearest" sub-sequence. The last state of the newly connected sub-sequence is then used as initial state and so forth. If no sub-sequence can be found from that state, the synchronization sequence or restart are re-applied. After all faults and their sensitization paths are connected, the complete test sequence is returned. Note that the sequence connection step is intended for testing all target faults with as little restarts as possible. However, this phase can be skipped for applications where individual test sequences are preferred.

III. Craig-interpolant-based model checking

In order to ease the understanding of the algorithmic details in the next section, this section briefly reviews Bounded Model

978-1-4673-1595-1/12/$31.00 ©2012 IEEE

Checking (BMC) [40], Craig interpolants [35] and McMillan's work on SAT-based model checking [36].

Among other applications, BMC is employed to derive error traces in sequential circuits that are required to satisfy a certain property P. The structure of the circuit and the problem conditions are encoded as a propositional formula of the form

$$BMC_k = I_0 \wedge T_{0,1} \wedge \ldots \wedge T_{k-1,k} \wedge P_k \quad (1)$$

I_0 encodes the initial state of the circuit. The terms of the form $T_{i,i+1}$ represent the so-called transfer function or transition relation that defines one system step from time point i to time point $i+1$. The last predicate P_k stands for a desired property whose satisfiability after k steps has to be checked. If the property never holds independently of the value of k, BMC_k is unsatisfiable for all k, whereas BMC_k is satisfiable if there exists a k and a path in the transition system that starts at I_0 and, after k transition steps, reaches a state in which P_k holds.

A classical BMC approach solves a series of problem instances. The first one is $BMC_0 = I_0 \wedge P_0$ (cf. Equation 1). It is satisfiable if the property holds in the initial state. If the instance is not satisfiable, BMC checks whether taking one more step into consideration will satisfy the property, i.e. whether the formula $BMC_1 = I_0 \wedge T_{0,1} \wedge P_1$ is satisfiable. This is repeated until P_k holds for some k, or until a user-defined maximal bound is reached [40].

In order to prove that a certain state of a transition system cannot be reached independently of k, i.e. that the desired property never holds, the circuit could be unfolded until reaching its diameter. However, very large k-values may be necessary. Hence, there are several approaches to find a fixed point sooner, including k-induction [41] and BDD-based approaches [42]. One particularly efficient method based on the theory of *Craig interpolation* [36] is employed by the *CIP*-solver [37] used in this work.

Formally, Craig interpolants [35] are defined as follows:

Theorem 1 (Craig): Let A and B be two propositional formulas such that their conjunction is unsatisfiable. Then, a formula C with the following properties exists:

1) C contains only variables which occur in both A and B.
2) $A \rightarrow C$
3) $C \rightarrow \neg B$

C is called a *Craig interpolant* of A and B. □

Craig interpolants represent over-approximations of all reachable system states after a certain number of transition steps. These over-approximations are used in additional fixed-point checks to detect whether all reachable system states have already been checked or not. If a fixed point is found the procedure has proven that the given property P never holds.

Figure 3 illustrates the Model Checking procedure (MC procedure) which combines BMC and Craig interpolation. As in classical BMC, the first step consists in testing whether the property holds in the initial state I_0, i.e. whether BMC_0 is satisfiable. The next problem instance to be tested is $BMC_1 = I_0 \wedge T_{0,1} \wedge P_1$. If this instance is unsatisfiable, by Craig's theorem, there is a formula C_1 such that $I_0 \wedge T_{0,1} \rightarrow C_1$ and $C_1 \rightarrow \neg P_1$. Moreover, C_1 contains only variables that occur in both sub-formulas, i.e. it contains only variables that represent the flip-flop contents. Given that $I_0 \wedge T_{0,1} \rightarrow C_1$, all flip-flop assignments (states in the sequential circuit) reachable after one transition step starting at the initial state, are over-approximated by C_1. Then, a fixed-point check is performed by evaluating whether the Craig interpolant implies the initial state, i.e. whether the set of states that are reachable after one transition step are initial states themselves. If so, the algorithm terminates as no new states can be reached starting at the current step.

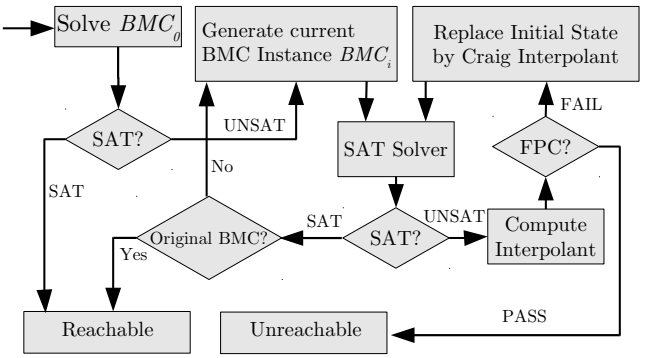

Fig. 3. BMC and Craig interpolation

If the fixed-point check fails, the next problem to solve is not BMC_2, but a variation of BMC_1 in which the initial state is replaced by the found Craig interpolant. If this instance is also not satisfiable, a new Craig interpolant is computed and used for a new iteration of the algorithm. In contrast, if the formula is satisfiable, verification is performed by solving the original BMC_i formula corresponding to the current unfolding depth, since the Craig interpolant is an over-approximation and may therefore contain non-reachable states. In case that the verification formula BMC_i is unsatisfiable, the overall algorithm restarts by computing a new Craig interpolant starting from BMC_i. See [36] for details.

IV. Generation of the MC-instance

In this section, we explain in detail the construction of the MC-instance $MC(I_0, T_{i,i+1}, P)$. It consists of three predicates, namely the initial state I_0, the transfer function $T_{i,i+1}$, and a target property P. Each predicate is given as a SAT-formula in conjunctive normal form (CNF) and encodes the different requirements for each individual step of the flow.

Figure 4 illustrates how these predicates are connected. The initial state I_0 defines the starting state of the sequence and corresponds to the initial logical values of the flip-flops.

Each application of the transfer function $T_{i,i+1}$ defines one arbitrary step of the sequential circuit from time point i to time point $i+1$. A Boolean formula encoding multiple time frames of a sequential circuit is obtained by connecting the transfer function multiple times, i.e. $T_{0,1} \wedge T_{1,2} \wedge \ldots \wedge T_{k,k+1}$. With each application of $T_{i,i+1}$ we extend the sequence by an additional time frame and therefore extend the search space.

The target property P describes the justification requirements that need to hold at the end of the sequence. Note that our

978-1-4673-1595-1/12/$31.00 ©2012 IEEE

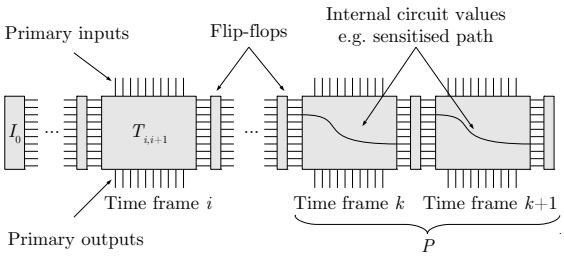

Fig. 4. MC formulation

method supports justification conditions for internal circuit lines over multiple time frames. Therefore, unlike in traditional BMC we can not only require certain target states, but also impose very specific requirements like the sensitization of a path with certain properties.

The complete MC-instance is passed to the CIP-solver that returns a Boolean satisfying assignment in case that a solution is found, or classifies the instance as unsatisfiable which means that no solution exists. A solution to such an MC-instance $MC(I_0, T_{i,i+1}, P)$ is a functional sequence of length k that starts at an initial state I_0 justifying the target property P. Note that, due to the iterative nature of the CIP-solver's algorithm, the returned sequence is guaranteed to have the shortest possible length. Also, the method is complete and guaranteed to find a solution if one exists, provided that the CIP-solver does not exceed the chosen time or memory restrictions. If the CIP-solver classifies the instances as unsatisfiable, no such sequence exists regardless of the number of time frames.

By defining I_0, $T_{i,i+1}$ and P specifically according to the requirements, we can derive a symbolic problem formulation for each step.

A. Computation of synchronization sequence

At the start of the method (propagation phase), a synchronization sequence is computed in order to obtain an initial functional state. To do so, we create an MC-instance $MC(I_0, T_{i,i+1}, P)$ using a 3-valued 01X-logic [43] that allows us to express unknown values.

As the state of the circuit is not known initially, we define I_0 as the all-X-state by adding clauses for each flip-flop that enforce the logic value 'X'. Analogously, at the end of an initialization sequence, the final circuit state needs to be well defined. Therefore, we define the zero-X-state, i.e. the state where each flip-flop is set to a known value, as target property.

When passed to the CIP-solver, the synchronization sequence, i.e. a functional sequence that changes the circuit's state to a known pattern starting from any initial state is returned.

B. Computation of longest reachable sensitizable paths

In the path generation step of the flow, the longest sensitizable path that can be reached from the start state of the flow is searched for every target gate. Just like in a usual combinational two-pattern delay test, the main task of such a path is to sensitize the fault location and to propagate the fault effect to an output. The actual delay of the sensitized path is one of the key properties influencing the quality of the small-delay test. Therefore, in order to achieve high small-delay quality of the test sequence, functional sensitization of each fault location through the longest sensitizable path is targeted.

In earlier work [32], we proposed a method to search for such a path in combinational circuits by defining a SAT-instance $S_G[\geq l]$. Such an instance $S_G[\geq l]$ is only satisfiable if and only if a sensitizable path through a target gate G with length greater or equal to l exists. This was done by encoding sensitization rules and path-length computations symbolically as a SAT-instance. Using a binary search over l, the longest combinational path can be found. In addition, further quality requirements (e.g. robustness conditions) are encoded into the instance. However, as the earlier method was intended for combinational circuits only, it potentially generated non-functional pattern pairs and relied on the use of scan to apply them.

In order to search for the longest functionally sensitizable path, we embed such a SAT-instance $S_G[\geq l]$ into the target property of the MC-instance $MC(I_0, T_{i,i+1}, P)$. This is done by generating a SAT-instance $S_G[\geq l]$ first, which encodes the path search in the combinational case using two time frames. In order to be compatible with P, a new SAT-instance $S'_G[\geq l]$ is generated by renaming each literal in $S_G[\geq l]$, such that its literal space does not overlap with the rest of the MC-instance. Therefore the complete SAT-instance $S'_G[\geq l]$ can be added as P. In order to connect $S'_G[\geq l]$ with the rest of the MC-instance, additional constraints are added to the target property. Using those additional constraints, we require each flip-flop to have identical values, both in P and in the initial state of $S'_G[\geq l]$. Therefore, each solution for the combined instance needs to satisfy both, the requirements of the MC-instance, i.e. the final state must be functionally reachable from the initial state, and the requirements of $S_G[\geq l]$, i.e. to sensitize a path through G of length $\geq l$. Additionally, since the initial state of $S_G[\geq l]$ is identical with the state in P, both parts are connectable with each other. As $S_G[\geq l]$ requires broadside tests, the combined sequence is functional. This representation allows to search for sensitizable paths that can be reached from the initial state while at the same time imposing quality requirements like length of the sensitized path or robust sensitization. Note that we do not impose any restrictions on the final output of the path and therefore also allow sensitized paths to end at a flip-flop.

C. Fault propagation

As we do not impose restrictions on the final output of a sensitization path, many of the sensitized paths actually end in a flip-flop, i.e. the fault effects are not directly observable. Hence, a propagation sequence needs to be computed that propagates the fault effect from the faulty output to an observable primary circuit output.

In order to map the fault propagation to the MC-instance, we encode the circuit by defining two logic values G and B for each circuit line. While G defines the logic value in the fault-free case, B defines the faulty value.

We developed several fault propagation modes, depending on the required type of invalidation immunity. In the basic case where no immunity is achieved, we assume a small delay

978-1-4673-1595-1/12/$31.00 ©2012 IEEE

effect that is large enough to cause a faulty value at the output of the longest sensitized path, but small enough to cause no fault effect at any other sensitized output. Therefore, we define the initial state to have identical values as in the final time frame of the previous path sensitization pattern for G and B for all flip-flops but the final flip-flop of the last sensitized path p. For that flip-flop, G and B are defined to be logic 1 and 0 respectively if p ends with a rising transition. In case of a fault effect, we assume the rising transition to be delayed and therefore B is still set to the logic values of the previous time frame. In case of p ending with a falling transition, G and B are defined to be logic 0 and 1, respectively.

The transfer function $T_{i,i+1}$ also defines G and B for each line, allowing to symbolically represent both values for each time frame. We force each primary input to have identical logic values for G and B, in order to ensure that the fault effect originates from the start of the propagation sequence.

The target property is defined to guarantee the fault propagation. This is formulated by requiring a difference of G and B on a primary output. As the only source of such a difference is the faulty flip-flop of the sensitized path p, the found sequence is guaranteed to propagate the fault effect to a primary output. Therefore, the combination of fault sensitization and propagation shown in Figure 2 represents a sequential test for one fault.

When immunity is required, we extend the encoding to 3-valued logic in order to represent unknown values. For F-immunity, we can not assume fault-free values at each flip-flops sensitized to the fault. Therefore, each such flip-flop is set to X in the initial state of the propagation sequence. This constraint enforces the returned propagation sequence to be independent of the actual logic value of that flip-flops and the fault propagation is F-immune.

For P-immunity, the individual logic values of the sensitizable path may be invalidated during fault propagation. Analogously to F-immunity, we set the output of each gate on the path to X. Hence, fault propagation does not depend on the propagation path and therefore a resulting sequence is P-immune.

D. Sequence connection

The main objective of the final sequence connection step is to connect each individual sequential test to a combined test sequence without requiring to restart the circuit after each test.

The problem of finding such a connection sequence has some similarities to an asymmetric traveling-salesman problem. Each test sequence can be mapped to a city whereas the number of time frames needed to connect one test sequence with another represents the distance. The number of time frames between two test sequences s_1 and s_2 can be mapped to an MC-instance by defining the final state of s_1 as initial state and the first state of s_2 as the target property. However, it is not practical to compute the distance between each test sequence as that would require $(n-1)^2$ instances where n is the number of test sequences. This information would be a prerequisite to compute the optimal solution of the problem.

To achieve good performance and low global sequence length at the same time, we solve the sequence connection problem using a greedy nearest-neighbor strategy. We start the final test sequence with the initial functional state. Using that start state as initial state, we define another MC-instance. As the target property, we require the sensitization of any yet unconnected test sequence. This is achieved by formulating a special target property that is satisfied if any test sequence p_i holds, i.e. the final state of connection sequence is identical to the initial state of p_i. Using that formulation, SATSEQ will return a sequence that sensitizes the nearest unconnected test sequence that can be reached from the initial state using as few time frames as possible. It is also possible to utilize initialization sequences for this step, if that results in a shorter connection sequence.

For the next run, the initial state is defined to be the final state of the last test sequence and again the nearest unconnected test sequence is connected. This is iterated until either all remaining sequences are connected, or no remaining sequences can be reached from the current state. The latter case can be true, if the state-space of the circuit has different connected components and therefore no single combined sequence exists. If such cases are identified, we allow the sequence to restart again from the very first functional state.

V. Experimental results

The flow described in the previous sections was applied to sequential ITC 99 and ISCAS 89 benchmark circuits. All experiments were executed on an AMD Opteron computer using one 2.6 GHz-core and up to 4 GB RAM. In all experiments, SATSEQ was set to classify an MC-instance as an abort after a timeout of 5 seconds. All run-times listed in this section are given in seconds. We used non-robust path sensitization and targeted the root node of each fanout-free-region for each circuit.

Columns 2, 3 and 4 of Table I report the number of small-delay faults in the circuit, the number of paths used for their detection, and the length of the resulting test sequence, respectively. Column 5 (Test application time – TAT) shows the duration (in clock cycles) of test application by scan. We calculate this number as $(\#FF + 2) \cdot \#P + \#FF$, where $\#FF$ is the number of flip-flops in the circuit and $\#P$ is the number of paths from Column 2. We assume that each path is tested by a test pair, which is scanned in followed by two functional cycles and scanned out in parallel with scanning in the next test pair. It can be seen that test sequences have reasonable lengths, much shorter than the duration of scan operations (even though we don't assume a slower scan clock). The number of required restarts and the total computation time are given in the remaining columns. The number of restarts is low in most cases, indicating that many sub-sequences can be successfully connected without new synchronization or reset. The run times are generally high but still reasonable, even though a much more complex test generation problem (small-delay fault testing through the longest sensitizable path) than in existing sequential stuck-at ATPGs is targeted. As usual for purely sequential ATPG, the number of considered time frames has a substantial influence on the test generation run times.

978-1-4673-1595-1/12/$31.00 ©2012 IEEE

TABLE I
METHOD APPLICATION TO ISCAS 89 AND ITC 99 BENCHMARK CIRCUITS

Circuit	Faults	Paths	Test-length	Scan-TAT	Restarts	Time
s00027	7	5	22	28	0	0.09
s00208	37	20	1129	208	0	32.70
s00298	54	28	358	462	0	18.62
s00344	57	35	248	610	0	499.18
s00349	58	35	248	610	0	480.64
s00382	76	36	1991	849	0	230.70
s00386	39	24	148	198	0	25.45
s00400	80	38	2183	895	0	214.55
s00420	75	20	499	376	4	308.14
s00444	92	39	2132	918	0	290.12
s00510	86	52	483	422	0	19.18
s00526	81	41	2286	964	0	341.59
s00641	98	50	306	1069	0	34.78
s00713	122	47	276	1006	0	37.76
s00820	63	50	473	355	0	154.25
s00832	63	50	473	355	0	155.80
s00838	151	20	564	712	4	691.73
s00953	210	115	1084	3594	0	55.51
s01196	187	113	628	2278	0	46.64
s01238	197	121	691	2438	0	59.52
s01423	259	139	3149	10638	1	5192.21
s01488	101	55	625	446	0	134.30
s01494	101	55	615	446	0	144.78
s05378	1000	432	4228	78371	0	7880.63
s09234	1263	398	4888	84985	190	22102.10
s13207	1986	316	3759	202878	302	31533.80
s15850	2199	839	10308	450238	694	117108.00
b01	24	16	141	117	0	0.41
b02	11	9	77	58	0	0.17
b03	72	35	401	1150	4	43.23
b04	244	153	1564	10470	105	166.66
b05	276	57	4525	2086	3	2130.78
b06	30	20	133	229	4	0.54
b07	201	41	3245	2140	12	1290.45
b08	69	38	1984	895	4	153.99
b09	81	46	1360	1408	20	15.40
b10	86	45	512	872	18	220.40
b11	203	76	3284	2539	1	968.33
b12	499	37	1325	4672	6	3488.32
b13	160	40	1090	2253	16	806.57
b14	2347	1114	17000	275403	104	43170.50
b15	2919	676	11894	305325	647	100482.00
b17	9086	522	5724	740566	516	191746.00

TABLE II
DETAILED RESULTS ON FAULT CLASSIFICATION AND OVERTESTING

Circuit	Faults	Detected		Non-Detected		Max avg. delay	
		F-imm.	No imm.	Untest.	Unknown	Prop.	Init.
s00027	7	7	0	0	0	0.00%	0.00%
s00208	37	34	1	2	0	0.00%	0.00%
s00298	54	44	5	5	0	6.92%	6.92%
s00344	57	47	7	3	0	10.53%	9.96%
s00349	58	47	7	4	0	10.53%	9.96%
s00382	76	40	21	13	2	10.13%	6.69%
s00386	39	36	3	0	0	2.76%	2.76%
s00400	80	44	23	11	2	6.42%	3.21%
s00420	75	34	1	6	34	51.97%	51.97%
s00444	92	48	28	14	2	5.14%	2.30%
s00510	86	57	23	6	0	4.74%	4.74%
s00526	81	39	21	16	5	9.26%	6.09%
s00641	98	82	2	14	0	32.54%	32.54%
s00713	122	106	0	16	0	32.53%	32.53%
s00820	63	39	24	0	0	5.71%	5.71%
s00832	63	39	24	0	0	5.71%	5.71%
s00838	151	33	2	14	102	76.26%	76.26%
s00953	210	196	11	3	0	1.64%	1.64%
s01196	187	187	0	0	0	3.71%	3.68%
s01238	197	195	2	0	0	3.69%	3.42%
s01423	259	155	83	17	4	31.77%	29.61%
s01488	101	68	31	2	0	1.14%	1.14%
s01494	101	68	31	2	0	1.14%	1.14%
s05378	1000	760	67	127	46	19.06%	16.83%
s09234	1263	451	188	208	344	54.13%	33.58%
s13207	1986	489	65	523	909	79.47%	57.80%
s15850	2199	980	286	332	491	44.48%	34.58%
b01	24	21	3	0	0	0.00%	0.00%
b02	11	9	2	0	0	3.23%	3.23%
b03	72	45	5	19	3	35.23%	35.23%
b04	244	236	7	1	0	3.83%	3.83%
b05	276	73	45	126	32	75.76%	69.32%
b06	30	28	2	0	0	1.26%	1.26%
b07	201	56	28	54	63	72.08%	57.64%
b08	69	47	9	9	4	8.70%	2.08%
b09	81	55	8	18	0	4.55%	4.55%
b10	86	59	14	7	6	15.13%	10.04%
b11	203	79	68	53	3	16.60%	12.88%
b12	499	70	14	34	381	82.70%	78.13%
b13	160	60	9	49	42	53.53%	34.40%
b14	2347	1917	104	308	18	5.06%	4.85%
b15	2919	744	611	366	1198	65.83%	62.41%
b17	9086	205	995	4892	2994	91.01%	88.51%

Detailed results of sequential ATPG are reported in Table II. Column 2 reiterates the total number of faults and columns 3 through 6 show the breakdown according to their detection status: detected by an F-invalidation-immune subsequence (Column 3); detected by a sub-sequence that is not F-invalidation-immune (Column 4); untestable by the sequential ATPG (Column 5); and aborted (Column 6). Average numbers for ISCAS 89 and ITC 99 benchmarks are shown in a separate row. Most faults are detected, and F-invalidation immunity can be ensured for the majority of the detected faults.

The final two columns quantify the impact of the sequential nature of the circuit on the quality of small-delay faults. The lengths of all sensitized paths are added to obtain the value L for the following three scenarios: L_{SEQ} (the sequential circuit where both initialization and propagation are performed); $L_{SEQNOPROP}$ (the sequential circuit without the need to propagate the fault effect); and L_{COMB} the combinational circuit where neither initialization nor propagation are done. A larger cumulative path length L indicates a larger coverage of small-delay faults. In Column 7 (Prop.), the propagation loss is given as $1 - L_{SEQ}/L_{COMB}$. This value quantifies the additional small-delay fault coverage which would be achieved if all flip-flops would be observable. The final column (Init.) shows the number $1 - L_{SEQNOPROP}/L_{COMB}$, which corresponds to the additional coverage achievable by testing the combinational core. This value quantifies the amount of overtesting which occurs when the circuit is tested using scan, as small-delay faults with a small size are detectable by scan testing but not detectable sequentially. It can be seen that this amount is significant and sequential ATPG can be useful in preventing overtesting in many cases.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a versatile technique for the generation of functional test patterns for small-delay faults in sequential circuits without the use of scan. The corresponding

978-1-4673-1595-1/12/$31.00 ©2012 IEEE

tool SATSEQ is based on model-checking and advanced SAT-technology. In contrast to other existing approaches, the presented flow is fully deterministic and guaranteed to produce the shortest possible sub-sequences of patterns. By means of extensive experiments, we demonstrated the applicability of the method and the high quality of the generated test sequence even under additional immunity requirements. In future we want to improve scaling and extend the method to yield instruction-based test sequences for microprocessors.

REFERENCES

[1] P. Girard, N. Nicolici, and X. Wen, eds., *Power-Aware Testing and Test Strategies for Low Power Devices*. Springer, 2010.

[2] S. Gerstendörfer and H.-J. Wunderlich, "Minimized Power Consumption for Scan-Based BIST," in *Int'l Test Conf.*, pp. 77–84, 1999.

[3] L. Whetsel, "Adapting scan architectures for low power operation," in *Int'l Test Conf.*, pp. 863–872, October 2000.

[4] D. Czysz, M. Kassab, X. Lin, G. Mrugalski, J. Rajski, and J. Tyszer, "Low-Power Scan Operation in Test Compression Environment," *IEEE Trans. on CAD*, vol. 28, pp. 1742–1755, November 2009.

[5] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper resistance mechanisms for secure embedded systems," in *Int'l Conf. on VLSI Design*, pp. 605–611, 2004.

[6] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a new dimension in embedded system design," in *IEEE Design Automation Conference*, pp. 753–760, July 2004.

[7] K. Hafner, H. C. Ritter, T. M. Schwair, S. Wallstab, M. Deppermann, J. Gessner, S. Koesters, W. Moeller, and G. Sandweg, "Design and test of an integrated cryptochip," *IEEE Design & Test of Computers*, vol. 8, pp. 6–17, December 1991.

[8] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard," in *Test Conference, 2004. Proceedings. ITC 2004. International*, pp. 339–344, 2004.

[9] L. Sourgen, "Security Locks for Integrated Circuits," 1993. US Patent 638459.

[10] D. Mueller, "Method of Protecting a Circuit Arrangement for Processing Data," 2002. US Patent 0087284.

[11] D. Hély, M.-L. Flottes, F. Bancel, B. Rouzeyre, N. Bérard, and M. Renovell, "Scan Design and Secure Chip," in *Int'l On-Line Testing Symp.*, pp. 219–224, July 2004.

[12] B. Yang, K. Wu, and R. Karri, "Secure Scan: a Design-for-Test Architecture for Crypto Chips," in *IEEE Design Automation Conference*, pp. 135–140, June 2005.

[13] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic, "Securing Designs against Scan-Based Side-Channel Attacks," *IEEE Trans. on Dependable and Secure Computing*, vol. 4, no. 4, pp. 325–336, 2007.

[14] M. Inoue, T. Yoneda, M. Hasegawa, and H. Fujiwara, "Partial Scan Approach for Secret Information Protection," in *European Test Symp.*, pp. 143–148, May 2009.

[15] W.-T. Cheng, "The BACK Algorithm for Sequential Test Generation," in *Int'l Conf. on Comp. Design*, pp. 66–69, 1988.

[16] J. P. Roth, W. G. Bouricius, and P. R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits," *IEEE Trans. on CAD*, vol. 16, no. 10, pp. 567–579, 1967.

[17] T. M. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," in *European Conf. on Design Automation*, pp. 214–218, 1991.

[18] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. on CAD*, vol. 30, pp. 215–222, 1981.

[19] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. on CAD*, vol. 32, pp. 1137–1144, 1983.

[20] T. P. Kelsey, K. K. Saluja, and S. Y. Lee, "An Efficient Algorithm for Sequential Circuit Test Generation," *IEEE Trans. on Computers*, vol. 42, pp. 1361–1371, November 1993.

[21] M. Boschini, X. Yu, F. Fummi, and E. Rudnick, "Combining Symbolic and Genetic Techniques for Efficient Sequential Circuit Test Generation," in *European Test Workshop*, pp. 105 –110, 2000.

[22] F. Corno, P. Prinetto, M. Rebaudengo, and M. Sonza Reorda, "GATTO: A Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits," *IEEE Trans. on CAD*, vol. 15, pp. 991 –1000, aug 1996.

[23] X. Lin, I. Pomeranz, and S. M. Reddy, "MIX: A Test Generation System for Synchronous Sequential Circuits," in *Int'l Conf. on VLSI Design*, pp. 456–463, 1998.

[24] R. Guo, S. M. Reddy, and I. Pomeranz, "PROPTEST: a property-based test generator for synchronous sequential circuits," *IEEE Trans. on CAD*, vol. 22, no. 8, pp. 1080–1091, 2003.

[25] M. Keim, N. Drechsler, R. Drechsler, and B. Becker, "Combining GAs and Symbolic Methods for High Quality Tests of Sequential Circuits," *Jour. Electronic Testing*, vol. 17, no. 1, pp. 37–51, 2001.

[26] A. Giani, S. Sheng, M. S. Hsiao, and V. D. Agrawal, "Efficient Spectral Techniques for Sequential ATPG," in *Conf. on Design, Automation and Test in Europe*, pp. 204–208, March 2001.

[27] J. Zhang, M. L. Bushnell, and V. D. Agrawal, "On Random Pattern Generation with the Selfish Gene Algorithm for Testing Digital Sequential Circuits," in *Int'l Test Conf.*, pp. 617–626, 2004.

[28] S. K. Devanathan and M. L. Bushnell, "Sequential spectral ATPG using the wavelet transform and compaction," in *Int'l Conf. on VLSI Design*, 2006.

[29] 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. Best Paper Award in the Test Category.

[30] R. Drechsler, S. Eggersglüß, G. Fey, A. Glowatz, F. Hapke, J. Schlöffel, and D. Tille, "On Acceleration of SAT-based ATPG for Industrial Designs," *IEEE Trans. on CAD*, vol. 27, no. 7, pp. 1329–1333, 2008.

[31] A. Czutro, I. Polian, M. Lewis, P. Engelke, S. M. Reddy, and B. Becker, "Thread-Parallel Integrated Test Pattern Generator Utilizing Satisfiability Analysis," *International Journal of Parallel Programming*, vol. 38, pp. 185–202, June 2010.

[32] 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.

[33] A. Czutro, M. Sauer, T. Schubert, I. Polian, and B. Becker, "SAT-ATPG Using Preferences for Improved Detection of Complex Defect Mechanisms," in *VLSI Test Symp.*, 2012.

[34] A. Czutro, M. Sauer, I. Polian, and B. Becker, "Multi-Conditional SAT-ATPG for Power-Droop Testing," in *European Test Symp.*, May 2012.

[35] W. Craig, "Linear Reasoning: A New Form of the Herbrand-Gentzen Theorem," *Journal of Symbolic Logic*, pp. 250–268, 1957.

[36] K. L. McMillan, "Interpolation and SAT-Based Model Checking," in *Int'l Conference Computer Aided Verification*, pp. 1–13, 2003.

[37] S. Kupferschmid, M. Lewis, T. Schubert, and B. Becker, "Incremental Preprocessing Methods for Use in BMC," *Formal Methods in System Design*, pp. 1–20, 2011. 10.1007/s10703-011-0122-4.

[38] M. Sauer, S. Kupferschmid, A. Czutro, S. M. Reddy, and B. Becker, "Analysis of Reachable Sensitisable Paths in Sequential Circuits with SAT and Craig Interpolation," in *Int'l Conf. on VLSI Design*, January 2012.

[39] K.-T. Cheng and H.-C. Chen, "Delay Testing for Non-Robust Untestable Circuits," in *Int'l Test Conf.*, pp. 954–961, 1993.

[40] E. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded Model Checking Using Satisfiability Solving," *Journal of Formal Methods in System Design*, pp. 93–98, 2001.

[41] M. Sheeran, S. Singh, and G. Stålmarck, "Checking Safety Properties Using Induction and a SAT-Solver," in *Int'l Conference on Formal Methods in Computer-Aided Design*, pp. 108–125, 2000.

[42] J. R. Burch, E. M. Clarke, D. E. Long, K. L. Mcmillan, and D. L. Dill, "Symbolic Model Checking for Sequential Circuit Verification," *IEEE Trans. on CAD*, vol. 13, pp. 401–424, 1994.

[43] M. Prasad, M. Hsiao, and J. Jain, "Can SAT be Used to Improve Sequential ATPG Methods?," in *Int'l Conf. on VLSI Design*, pp. 585 – 590, 2004.

978-1-4673-1595-1/12/$31.00

©2012 IEEE