Determining Application-specific Knowledge for Improving Robustness of Sequential Circuits

Sebastian Huhn*† Stefan Frehse† Robert Wille†‡ Rolf Drechsler*†

*Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
†Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
‡Institute for Integrated Circuits, Johannes Kepler University Linz, 4040 Linz, Austria
{huhn,drechsle}@informatik.uni-bremen.de stefan.frehse@dfki.de robert.wille@jku.at

***Abstract*—Due to their shrinking feature sizes as well as environmental influences such as high-energy radiation, electrical noise, particle strikes, etc., integrated circuits are getting more vulnerable to transient faults. Accordingly, how to make those circuits more robust has become an essential step in today's design flows. Methods increasing the robustness of circuits against these faults already exist for a long period of time but either introduce huge additional logic, change the timing behavior of the circuit, or are applicable for dedicated circuits such as microprocessors only.**

In this work, we propose an alternative method which overcomes these drawbacks by determining application-specific knowledge of the circuit, namely the relations of Flip Flops and when they assume the same value. By this, we exploit partial redundancies, which are inherent in most circuits anyway (even the optimized ones), to frequently compare circuit signals for their correctness – eventually leading to an increased robustness. Since determining the correspondingly needed information is a computationally hard task, formal methods such as Bounded Model Checking, SAT-based Automatic Test Pattern Generation, and Binary Decision Diagrams are utilized for this purpose.

The resulting methodology requires only a slight increase in additional hardware, does only influence the timing behavior of the circuit negligibly, and is automatically applicable to arbitrary circuits. Experimental evaluations confirm these benefits.

I. Introduction

Several breakthroughs in the field of the design, fabrication, and test of integrated circuits allowed for the implementation of highly complex *Integrated Circuits* (ICs). These ICs fulfill several mission- or even safety-critical tasks at once while following a highly complex functional behavior.

The enormous complexity and the intensive environmental interaction, particularly in case of long-term autonomous systems, forms a straining environment. Moreover, shrinking feature sizes as well as different environmental influences such as high-energetic radiation, electrical noise, particle strikes, etc. frequently cause unintended behavior of an IC, which can lead to disastrous consequences in the worst case. In particular, the sequential components of the IC, i.e., *Flip Flops* (FFs), are characteristically vulnerable to so-called transient faults. Such a fault appears in form of a toggled bit for a short period of time and is not logically, electrically, or temporarily masked, hence, the output signals of the system are invalidated. Due to this fact, these FFs have to be explicitly protected. To address this highly relevant field, powerful mechanisms have to be developed, which are capable to detect and react on occurring transient faults.

In this context, the *robustness* of a given circuit is an important metric, which can be derived from the number of non-robust FFs that are vulnerable to transient faults. In order to increase the robustness of a sequential circuit, the number of vulnerable (non-robust) FFs needs to be decreased. To this end, the corresponding FFs are *hardened* by extending the investigated circuit such that the respective values are recomputed and, in case of faults, faulty signals can be replaced. These recomputations are usually conducted either by additionally employing redundant hardware or redundant time. More precisely, existing methods which are currently applied in order to increase the robustness of a given sequential circuit can roughly be categorized into the following three schemes [1]:

- *Space-based approaches*, which embed additional logic blocks to generate certain redundancy in order to enhance their robustness. Approaches such as *Triple Modular Redundancy* (TMR) [2] or *Error-Correction Code* [3]–[5] constitute representatives of this scheme.
- *Timing-based approaches*, which influence the timing behavior of the considered circuit in order to guarantee correct output values at the FFs. Representative candidates of this scheme have been proposed in [6], [7].
- *Application-specific approaches*, which only consider dedicated parts of a circuit for which a robust solution is explicitly derived. Examples of this scheme include, e.g., dedicated fault-tolerance control flows for microprocessors [8]. Other application-specific approaches exploit invariants automatically to ensure correctness, e.g., hardware assertions [9]. Those assertions are used to uncover violations of the specific functional behavior during verification, but do not focus on any dedicated fault model considering transient faults and, hence, no compact realization is given a priori.

However, all these schemes come with significant shortcomings: Space-based approaches introduce huge additional logic into the circuit – often caused by naive multiplication of sequential elements and, hence, a redundancy which is not necessarily needed for the functional behavior. Timing-based approaches suffer from the fact that they potentially increase the latency and, thus, cause restrictions to the actual design of the circuit. Application-specific approaches are limited to dedicated parts of a circuit (e.g., the considered microprocessor) and, hence, are not applicable for sequential circuits in general.

In this work, we are aiming to address these shortcomings by proposing a solution, which does not simply recomputes FFs values of the original design, but instead determines application-specific knowledge of their behavior[1]. More precisely, the proposed methodology determines the relations of FFs and stores the conditions under which FFs assume the same logic value to partially hardens the FFs. This knowledge allows employing a dedicated logic block which compares all corresponding FF values and, hence, can detect if one of them inherits a fault. The proposed application-specific knowledge is derived out of typical signal values which really occurs in the intended functional operation. Common techniques as described above cover the entire state space exhaustively, i.e., those techniques cover even a high fraction of irrelevant state space.

By this, redundancies within the IC are exploited. Even if, in a typical IC design flow, different optimization techniques are applied to eliminate redundant components, often a significant amount of functional equivalence remains and can be exploited for the purposes considered here. As an example, consider a *Full Adder* (FA) with three inputs (a, b, c) and a *Half Adder* (HA) with two inputs (a, b). Both of these adders have two outputs representing the sum and the carry-out bit. Although both circuits are not completely functional equivalent (both indeed realize slightly different functions), their behavior is identical whenever the carry input c is set to '0' – both circuits are partially functional equivalent and generate the same value in certain states. Having the information on what signals assume the same value in what state can be used to compare those signals with each other and, by this, to strengthen the robustness of sequential circuits. Later, Section III provides a more detailed description of the proposed idea.

Overall, this yields substantial improvements compared to the other solutions reviewed above: In fact, exploiting the information on partial redundancies allows increasing robustness with only a moderate hardware overhead in terms of gate count compared to the space-based approaches. At the same time, the timing behavior is only affected negligibly (in contrast to the timing-based approaches). Finally, the resulting flow is automatically applicable to any sequential circuit and, hence, not limited to dedicated circuits as the existing applications-specific solutions.

Instead, the proposed methodology needs to determine the needed information, i.e., the relations of FFs and the conditions under which they assume the same logic value. Since this is a computationally hard task, we propose to address this with a dedicated orchestration of formal methods such as *Bounded Model Checking* (BMC, [11]), powerful solvers for the *Boolean Satisfiability Problem* (SAT problem, [12]), SAT-based *Automatic Test Pattern Generation* (ATPG) and compact data structures involving *Binary Decision Diagrams* (BDDs, [13]), which is capable of coping with this complexity. Finally, this approach is flexible in the sense that the designer can easily configure the trade-off between the hardware overhead and the desired enhancement in robustness.

Experimental results confirm the benefits of the proposed methodology. In fact, robustness of a circuit can be increased to approx. 95% in most of the cases, while the circuit size increases only by a factor of approx. 1.06 on average. Compared to established methods (e.g., TMR, where indeed 100% robustness is achieved, but at the expense of more than tripling the circuit size), this provides a suitable alternative and a trade-off between robustness and hardware overhead.

The proposed methodology is described in the remainder of this paper as follows: Section II provides a comprehensive description of the background which is needed to make this paper self-contained – including a review on sequential circuits, transient faults, and robustness. Afterward, the general idea and the general flow of the proposed approach are described in Section III. A detailed description of the main components, namely the Partition Enumeration and the State Collector, are presented in Section IV and V, respectively. The conducted experimental results are summarized and discussed in Section VII. Finally, conclusions are drawn in Section VIII.

II. Background

This section briefly reviews the relevant terminologies used in the remainder of this paper to assess the vulnerability of ICs. First, the concept of sequential circuits is formally introduced and the frequently used fault model is covered. Finally, an established metric to qualitatively assess the vulnerability of a circuit with respect to these faults is reviewed.

A. Sequential Circuits

A sequential circuit Φ is given as a commonly known gate level representation that consists of *Primary Inputs* (PIs), *Primary Outputs* (POs), combinational gates G, and *sequential elements* (SE) such as FFs, i.e., $\Phi = (\mathsf{IN}, \mathsf{OUT}, \mathsf{G}, \mathsf{SE})$. The sequential elements are assumed to be synchronous to (at least) one clock domain.[2] The FFs of a given sequential circuit can be grouped by a *hierarchical levelizing* procedure. Two FFs FF_i and FF_j are contained in the same group, if the number of FFs in both fan-in cones are the same on the shortest path towards the PIs.

Alternatively, a sequential circuit can also be represented by a *Finite State Machine* (FSM). An FSM is defined by a tuple $M = (I, S, T)$, where I describes the set of initial states, S represents the state space of the circuit, and T defines the *transition relation*. A *transition relation* $T(s, s')$ evaluates to true, if there is at least one transition from state s to state s'. The *set of reachable states* $S^* \subseteq S$ contains those states that are reachable from an initial state in an arbitrary number of steps.

[1] A preliminary version of this work has previously been presented in [10].

[2] In order to ease the following descriptions, we will assume a single clock domain. However, the proposed methodology can be extended to further clock domains as well.

B. Transient Faults

The shrinking feature size leads to an increased vulnerability of circuits against *Single Transient Faults* (STFs), which are typically caused by *Single Event Upsets* (SEUs), e.g., high-energetic radiation, electrical noise, particle strikes, or other environmental effects [14], [15]. Typically, the influence of a transient fault occurring at a FF is modeled as an unintended toggled output value. This influence can possibly cause an invalid and unintended behavior of the circuit Φ for a short period of time.

In order to increase the robustness of a circuit Φ, a *Fault Detection Mechanism* (FDM) can be applied that handles cases in which a single transient fault occurs at a FF, e.g., to realize precautions.

C. Assessing Robustness

To consider the vulnerability of sequential circuits against transient faults, a metric for robustness has been introduced, which measures the fault tolerance (i.e., the robustness) with respect to a fault model [16], [17]. More precisely:

Definition 1: Let $\Phi = (\mathsf{IN}, \mathsf{OUT}, \mathsf{G}, \mathsf{SE})$ be a sequential circuit. A FF is considered to be non-robust, if there is at least one reachable state and one transient fault such that the output behavior of Φ is inverted. Let N be the *set of non-robust FFs* with $N \subseteq \mathsf{SE}$. Then, the robustness of Φ can be determined as follows [18]:

$$\mathcal{R} = 1 - \frac{|N|}{|\mathsf{SE}|}$$

In order to determine the robustness of a given sequential circuit, the non-robust FFs N can be computed by either formal methods [19] or simulation-based techniques [20]–[23]. In general, the robustness can be determined using a simulation-based approach as follows:

1) Define a number r of simulation cycles to be considered while adjusting the state of the circuit which is finally used for fault injection. Beside this, define a number k of cycles to be simulated for fault propagation.
2) The PIs of a given sequential circuit Φ are stimulated up to $r - 1$ cycles using random values.
3) In cycle r, the state s_r is extracted from the simulation environment. A copy $\widehat{s}_r$ of s_r is modified so that an STF is injected at a randomly chosen FF $f \in \mathsf{SE}$ and, hence, the output value of f is toggled.
4) The circuit Φ is simulated twice for up to k cycles: One simulation starts from the healthy state s_r and one from faulty state $\widehat{s}_r$ containing the injected STF. During both simulation runs and for every clock cycle, the same PIs values are driven.
5) In cycle $r + k$, all POs of both simulation runs are compared. If at least one of the POs values differs, the f is non-robust unless the circuit contains an FDM reporting a fault.
6) This procedure is repeated from Step 3 until all FFs are covered by fault injection.

Due to the nature of the random simulation, the number of covered states depends on the chosen parameters for r and k.

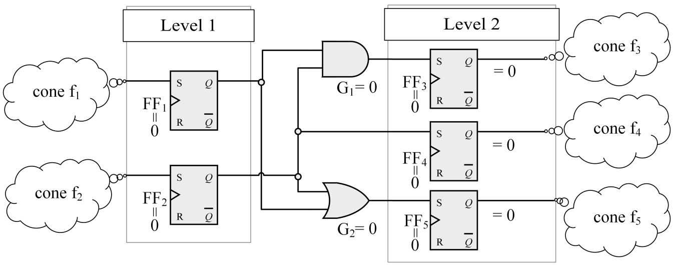

Fig. 1: A non-robust sequential circuit

III. Proposed Approach

This section introduces the proposed methodology in somewhat more detail. To this end, the general idea is motivated and illustrated. Afterward, we summarize the core components needed to realize the ideas. This provides the basis for the detailed description of the implementation of the proposed methodology which follows in Sections IV-VI.

A. General Idea

The general idea of the proposed methodology rests on the following observations:

- Today's circuits usually contain a huge number of FFs, which can store at least a single bit, i.e., '0' or '1'. If a single FF is affected by a transient fault, this bit is toggled. Existing approaches insert redundant logic into the design, e.g., to recompute the correct value, which causes a significant hardware overhead.
- At the same time, the value of an observed single FF is often equal to the value of many other FFs. Moreover, since the behavior of the circuit is known, it is possible to determine the relation between them, i.e., the states in which certain FFs assume the same value.
- Hence, instead of introducing redundancy for recomputations, we propose to simply compare the value of a FF to the values of other FFs from which it is known that, for the respectively considered state, they are supposed to generate the same value.

In order to realize this idea, a formalism is required that posts whether a partition of non-robust FFs assumes the same value for given reachable states. In the following, this is formally described in terms of an equivalence property.

Definition 2: Let $P_j \subseteq N$ be a partition of at least two non-robust FFs and $\widehat{S} \subseteq S^*$ be a set of reachable states. Then, an *Equivalence Property* (EP) is defined by

$$\mathsf{EP}(\widehat{S}, P_j) := \left\{ f_1, \ldots, f_l \in P_j \;\middle|\; \begin{array}{l} \text{all FFs } f_1, \ldots, f_l \text{ outputs the} \\ \text{same value under the same} \\ \text{state } s \in \widehat{S} \end{array} \right\}$$

and evaluates to true if all combinations of FFs $f_n, f_m \in P_j$ assume the same output value in all of these states $\widehat{S} \subset S^*$.

Example 1: Consider the circuit shown in Fig. 1 which is composed of five FFs distributed in two hierarchical circuit levels 1 and 2. If both FF_1 and FF_2 (level 1) are set to '0', then FF_3, FF_4, and FF_5 (level 2) are assumed to have the same output value '0' after a single clock cycle. This scenario is represented by an $\mathsf{EP}(\widehat{S}, P_j)$ with the partition $P_j = \{FF_3, FF_4, FF_5\}$ and the state $s_j \in \widehat{S}$ being defined by $FF_1 = 0$ and $FF_2 = 0$, i.e., $\mathsf{EP}(\widehat{S}, P_j) = 1$ holds.

Using the Equivalence Property and the general idea sketched above, robustness of sequential circuits is enhanced as follows:

1) Determine the set $N \subseteq \mathsf{SE}$ of non-robust FFs of the given sequential circuit. The assessment of robustness as reviewed in Section II-C can be utilized.
2) Consider all non-robust FFs N and determine the level-wise subsets $N_i \cup N_{i+1} \cup \cdots \cup N_L = N$ with $1 \leq i \leq L$ according to their hierarchical circuit levels (L being the total number of hierarchical levels in a rank-ordered circuit [24, p. 45]). Furthermore, assume that each FF has exactly one hierarchical level: $N_i \cap N_j = \emptyset \; \forall i \neq j$. The above described clustering allows to execute all FFs' comparisons within one single time frame. This is crucial to reduce the complexity of the calculation itself and, especially, the costs of the robustness improvement. Consequently, there is no need to hold specific FF values over different time frames, e.g., by introducing further, potentially vulnerable, state elements while massively increasing the computational effort as well as hardware scale. Thus, the level-wise sets of non-robust N_i are exclusively used in the remainder.
3) For each level $1 \leq i \leq L$ and for all subsets of non-robust FFs $N_i \subseteq N$, determine suitable partitions $P_j \in \mathcal{P}(N_i)$ and a set of reachable states $\widehat{S} \subseteq S^*$ such that all FFs in P_j are supposed to generate the same value, i.e., determine P_js and corresponding $\widehat{S}$s for which $\mathsf{EP}(\widehat{S}, P_j) = 1$ holds.
4) Using the knowledge from the obtained EPs, synthesize a *Fault Detection Mechanism* (FDM). To this end, realize the following logic blocks:

- *Activator* $\mathcal{A}$: Generates a signal $\mathcal{A}$ (supposed to trigger the FDM) stating whether ($\mathcal{A} = 1$) or not ($\mathcal{A} = 0$) the FFs in P_j are supposed to generate the same value under the current state $s \in \widehat{S}$. This signal is directly calculated by the current state (single time frame) of FFs within the fan-in cone. More precisely, it is not required to consider previous values, which is solely enabled by the hierarchical sort as described above in Step 2).
- *Comparator* $\mathcal{C}$: Generates a signal $\mathcal{C}$ stating whether ($\mathcal{C} = 1$) or not ($\mathcal{C} = 0$) all FFs in a partition P_j to be hardened actually assume the same output value.
- *Detector*: Generates a fault signal $\mathcal{F}$ reporting the detection of a fault. A fault is detected, if not all FFs in a partition P_j assume the same output value (i.e., $\mathcal{C} = 0$), although they are supposed to do that for the current state (i.e., $\mathcal{A} = 1$), i.e., $\mathcal{F} = \neg\mathcal{C} \wedge \mathcal{A}$.

This proposed FDM detects transient faults occurring in FFs of the considered circuit. If a fault is detected, an introduced fault signal $\mathcal{F}$ is driven. This enables the realization of precautions against faulty behavior at the POs, e.g., by resetting the circuit or masking the affected POs. Overall, this leads to an enhanced robustness. The ratio of the enhancement can thereby be controlled, e.g., by adjusting the number knowledge collected through the EPs.

B. Realization

The *bottleneck* of the proposed methodology is the determination of – as much as possible – application-specific knowledge in terms of EPs. Ensuring the completeness would require that all possible partitions $P_j \in \mathcal{P}(N_i)$ of all non-robust FFs in the same hierarchical circuit level are considered. Obviously, this lead to an exponential complexity, which is not feasible for practical applications. Moreover, most of the partitions P_j are likely to be not suitable for an EP anyway, since no state s_j may exists for them so that all assume the same value.

In order to realize this proposed approach effectively, a mechanism is introduced, which aims to determine *good* partitions. Particularly, a SAT-based ATPG model is adopted to compute the criteria of quality for an investigated partition. In addition to that, we heavily exploit formal methods such as *Bounded Model Checking* (BMC, [11]), powerful solvers for the *Boolean Satisfiability Problem* (SAT problem, [12]), and compact data structures involving *Binary Decision Diagrams* (BDDs, [13]).

Eventually, this leads to a methodology composing:

1) A *Partition Enumerator* selects suitable partitions $P_j \in \mathcal{P}(N_i)$ which have not been considered before.
2) A *State Collector* determines the states $\widehat{S}$ under which all FFs in the selected partition P_j assume the same value and, hence, determines all $\mathsf{EP}(\widehat{S}, P_j)$ evaluating to true.
3) An *FDM Synthesizer* takes the obtained knowledge, realizes the FDM, and, eventually, embeds the resulting logic into the original circuit.

In the remainder, each step is described in more detail and illustrated by means of the circuit considered in Fig. 1.

IV. Partition Enumerator

The partition enumerator is supposed to select partitions of FFs which are likely to generate the same value in certain states. To this end, all non-robust FFs (given in N) are initially distinguished according to their hierarchical circuit level. The FFs of a circuit level N_L are then divided into sets of partitions, which are separately considered. However, since it is not practically feasible to enumerate all possible partitions, a method is needed which returns a proper set of partitions yielding *good* robustness results.

For this purpose, two different methods are proposed in the following: The first method is a random-based computation $\mathcal{P}_{\mathsf{RAND}}$ and the second one $\mathcal{P}_{\mathsf{SAT}}$ exploits SAT techniques. However, both techniques differ strongly concerning their complexity and quality. Generating a random set is considerably less time-consuming than solving a SAT-instance. In contrast, the quality of the enumerated sets is much better following the SAT-based approach, since the respectively guided search considers the functional behavior of the circuit.

Algorithm 1: Partition Enumeration procedure

```
   Data: set of non-robust FFs: $N_i$, partition generator: $\mathcal{P}_{\mathsf{GEN}}$
1  Container $\mathcal{E} = \emptyset$    /* Data container for EPs */
2  Context $ctx$              /* Overall context */
3  while True do
4      Let $P_j = \mathcal{P}_{\mathsf{GEN}}(ctx, N_i)$
             /* PGEN calls PRAND _or_ PSAT */
5      if $P_j = \emptyset$ then
6          return $\mathcal{E}$
7      $\widehat{S} = \mathsf{StateCollector}(P_j)$
8      if $\widehat{S} \neq \emptyset$ then
9          $\mathcal{E} = \mathcal{E} \cup \mathsf{EP}(\widehat{S}, P_j)$
10         $ctx.\text{accept}(P_j)$
11     else
12         $ctx.\text{decline}(P_j)$
```

For both methods $\mathcal{P}_{\mathsf{RAND}}$ as well as $\mathcal{P}_{\mathsf{SAT}}$, a common algorithmic framework as shown in Algorithm 1 is used: The non-robust FFs N_i and a partition generator function $\mathcal{P}_{\mathsf{GEN}}$ – sharing the algorithmic base of both methods– are provided as input. The generator function additionally gets a context object representing the actual state and providing two functions: accept marks a partition to be accepted and decline marks that the State Collector could not identify any state satisfying EP, which is strictly required as stated in Definition 2.

The approach evaluates new partitions as long as $\mathcal{P}_{\mathsf{GEN}}$ generates non-empty ones by invoking either the random-based $\mathcal{P}_{\mathsf{RAND}}$ or the newly proposed SAT-based $\mathcal{P}_{\mathsf{SAT}}$ technique based on designer's choice. In contrast to the $\mathcal{P}_{\mathsf{RAND}}$ random technique, $\mathcal{P}_{\mathsf{SAT}}$ invokes the proposed metric to determine the quality (with respect to the stated criteria) of an arbitrary partition. This allows evaluating the quality of the individual partition to, especially, select the best – again with respect to the stated criteria – one. In each iteration, the generated partition is passed to the State Collection (Line 7) that computes states $\widehat{S}$ in which all FFs in the currently considered partition P_j assume the same value (described in detail in Section V). If at least one state $s \in \widehat{S}$ exists (Lines 8 to 10), an EP is created using the states $\widehat{S}$ determined by the State Collector as well as the currently considered partition P_j. Then, the resulting EP is stored within a global data container $\mathcal{E}$ (used later by the FDM Synthesizer) and the partition P_j is accepted by calling accept. In contrast, if no state exists, i.e., $\widehat{S} = \emptyset$ the partition is declined by calling decline.

A. Randomized Search

The first method $\mathcal{P}_{\text{RAND}}$ to obtain partitions selects some FFs of a given set randomly. Generating those partitions is very fast but does not consider any logical context of the FFs. However, this naive approach of randomly selecting partitions achieves already strong improvements in terms of robustness while keeping the hardware overhead low.

Technically, a set of non-robust FFs and a pre-defined upper bound p_s are provided as an input. The resulting size of the generated partition is less than the pre-defined bound. The algorithm randomly picks an uncovered FF and adds it to the partition. Finally, the partition is checked whether there are some states to satisfy EP. If there are such states, the partition is taken for hardening. Otherwise, another partition is generated. Eventually, the algorithm terminates since already checked partitions are stored in order to avoid loops.

B. SAT-based Search

Due to the huge number of possible partitions, enumerating all of them is practically not feasible. Thus, a further method is proposed, which conducts a *SAT-based* search. This addresses the crucial task of determining a *good* subset of promising partitions. The basic idea of the SAT-based search $\mathcal{P}_{\text{SAT}}$ is that FFs of a set P forms a *good* partition if (a) an assignment of the fan-in cone exists such that an occurring transient fault in one $FF \in P$ is propagated and visible towards at least one primary output and (b) all FFs of P assume the same value. Counting the total number of those test patterns (cf. Definition 3) can be assumed as a metric as formally given in Definition 4 for the qualitative evaluation. Thus, it is possible to rate different sets of FFs (partitions), i.e., the most promising candidate can be derived out of this evaluation.

More precisely:

Definition 3: Given a set of non-robust FFs $P \subseteq N_L$ of a level L, a test pattern is a set of assignments of primary inputs such that a transient fault of any flip flop of P is observable at the primary outputs. All test patterns are denoted as $\mathsf{TP}(P)$.

Definition 4: Given a set of test patterns of FFs $\mathsf{TP}(P)$. The *test pattern metric* (denoted $M(P)$ for a given partition P) is defined as

$$M(P) = \frac{|\mathsf{TP}(P)|}{2^{|X|}},$$

where X defines the number of primary inputs as well as FFs in the fan-in cone of partition P.

Observation 1: Given two sets of FFs $P_1 \subseteq N_i$ and $P_2 \subseteq N_i$ of a circuit level i and let $M(P_1)$ and $M(P_2)$ be two ratings of sets of FFs P_1 and P_2, respectively. Then, without loss of generality, $m_1 > m_2$ states that P_1 is more suited to satisfy an EP than P_2. This is due to the relatively high percentage of states, which are considerable for an EP.

In the following part, the underlying algorithmic approach is described in more detail, which takes advantages of the Observation 1 to, eventually, implement the proposed SAT-based approach.

1) Background: The proposed method is inspired by *Automatic Test Pattern Generation* (ATPG), particularly, the formal model typically used by SAT-based ATPG [25]–[27]. Generally, ATPG identifies a test set, more precisely, a set of input stimuli (test patterns) for a considered circuit, which is capable to detect a certain fault. Here, a fault is an assumed faulty behavior of a circuit's component due to physical defects. Different well-known structural [28] as well as formal SAT-based [25]–[27] approaches exist, which both determine suitable test sets for arbitrary sequential circuits.

Generally, the SAT problem is about determining a satisfying solution for a given Boolean function (or proving that no such solution exists). These functions can be represented in *Conjunctive Normal Form* (CNF). A CNF Φ is a conjunction of clauses, whereby, a clause ω is a disjunction of literals and a literal represents a Boolean variable ν in its positive x or negative form $\bar{x}$. The Boolean function $\Phi : \{0,1\}^n \rightarrow \{0,1\}$ is classified as *satisfiable* (sat) if an assignment of all variables exists such that $\Phi = 1$ holds. Otherwise, it is classified as *unsatisfiable* (unsat) [29].

These SAT-based ATPG approaches [25]–[27] allow determining test pattern for even hard-to-detect faults in complex designs. This is due to the fact that powerful SAT-solvers are applied for determining the actual test pattern, hence, a CNF has to be generated, processed (by a SAT-solver) and, especially, used to extract the test pattern. At first, the investigated circuit is completely converted into a CNF representation as it is, i.e., in a fault-free fashion.[3] Beside this, a fault (with respect to the considered fault model) is introduced into a copy of this circuit. More precisely, this faulty circuit is reduced to the essential components, which are affected by the introduced fault, and translated into a CNF representing the faulty circuit. Finally, a miter circuit is expanded to the corresponding primary output pairs of both, the fault-free and faulty circuit. The proposed SAT-based approach adopts the well-known SAT-based ATPG scheme and extends the underlying SAT-instance by the problem specific requirements, which are discussed in the following. Especially, this leads to the generation of a formal representation, which allows invoking powerful solving engines and, finally, to determine certain criteria of quality.

2) Formal Model: Given a set P of FFs that needs to be rated. The basic principle of the SAT-based approach is shown in Figure 2 for partition P, more precisely, the adopted model of the SAT-based ATPG [27], [32] is shown. This model is used to generate the SAT-instance and mainly consists of the following: At first, the fault-free sub-circuit, which behaves like the original circuit. All FFs $FF_1, ..., FF_3$ of partition P are included, whose outputs are assumed to be equal. Additionally, the fan-in cone of all these FFs are included in the sub-circuit as well, i.e., towards the previous hierarchical FF level or towards the primary inputs, respectively. Secondly, the faulty sub-circuit holds a single $\widehat{FF}_3 \in \mathcal{P}$ modeling a transient fault. Hereby, the sub-circuit contains the fan-in and fan-out cone from the fault-free as well as faulty sub-circuit are driven by the same *Test Pattern* (TP). It is assumed that this transient fault – while following the fault injection of [18] – tampers at least one signal in the cone of *Test Response* (TR), which is directly driven by $\widehat{FF}_3$. Thus, a Boolean difference is enforced between the corresponding signals TR and $\widehat{TR}$. This model is translated into SAT-problem resulting in a SAT-instance. If the SAT-instance is satisfiable, there is assignment of the primary inputs - forming the test pattern. All test patterns up to an upper bound are computed. Finally, a rating is computed by calculating $M(P)$.

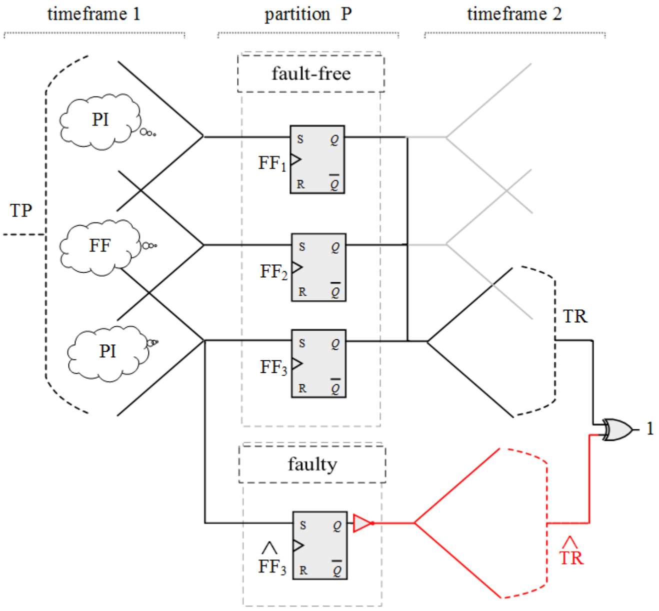

Fig. 2: SAT-based circuit model

Algorithm 2: State Collecting procedure

```
Data: enumerated partition: $P_j$, max. number of states: $u$
Data: unrolling depth: $l$
$\widehat{S} = \emptyset$                          /* stored as BDD */
for $k = 1$ to $l$ do
    $F = \mathsf{SFind}(P_j, k)$
    repeat
        if $|\widehat{S}| > u$ then return $\widehat{S}$
        $\widehat{S} = \widehat{S} \cup s_{i+1}$          /* collects state */
        $F = F \wedge \neg s_{i+1}$        /* blocks solution */
    until $\mathsf{SAT}(F)$
return $\widehat{S}$
```

Despite the complexity of solving a SAT-problem reasonable run times to solve the instance are achieved by applying very common techniques, e.g., incremental satisfiability, cone-of-influence reduction.

3) Algorithm: Given a set of non-robust FFs N_L of a circuit level L. The SAT-based search first randomly builds a pre-defined number of subsets $P_1, \ldots, P_n$ of N_L. For each partition the test pattern metric $M(P)$ is obtained. Finally, the partition with the highest rating is considered as best partition and returned as result. To improve the efficiency of the realization, common techniques from SAT-based ATPG are applied, for instance, *incremental satisfiablity* [32], [33] as well as cone-of-influence reduction [34] are both applied.

V. State Collector

Once promising partitions have been determined by the Partition Enumerator, the main task of the State Collector is to

[3] Note that, in order to ease the description, we will not cover the details of the CNF conversion. This has already sufficiently been covered by previous work e.g. in [30], [31].

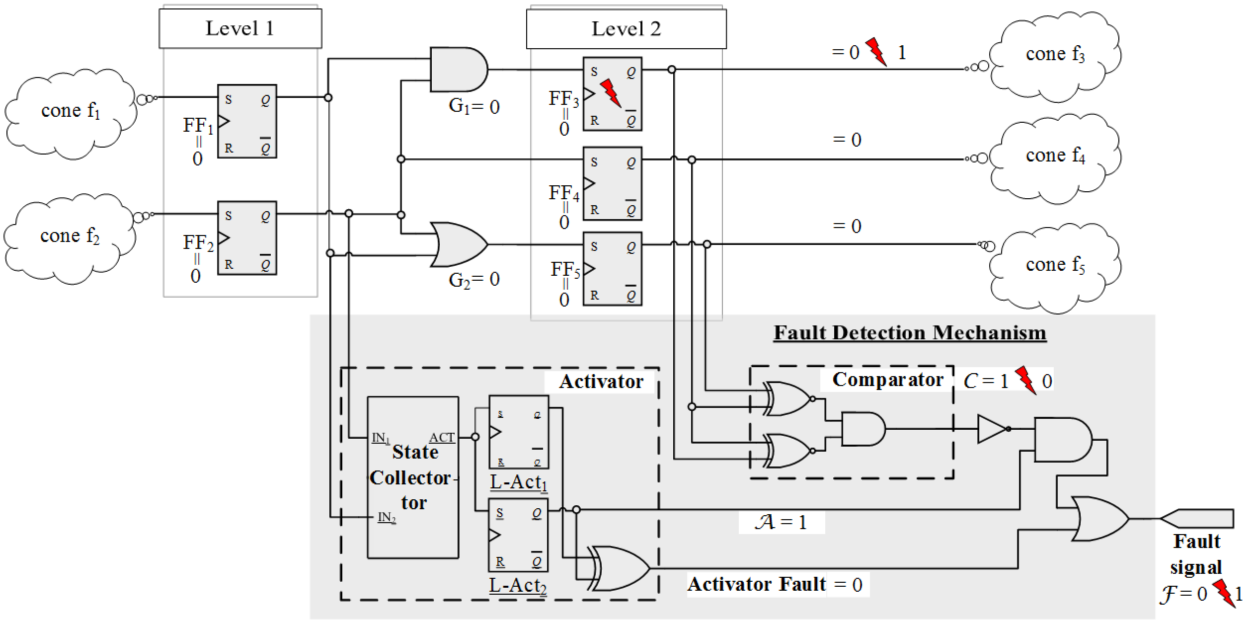

Fig. 3: Applying the proposed methodology to the circuit from Fig. 1

determine reachable states $\widehat{S}$ such that the EP holds for a partition P_j obtained by the Partition Enumerator. These states are collected by utilizing *Bounded Model Checking* (BMC). This technique is initially designed for functional verification of digital circuits proving or disproving a temporal property [35], [36]. A bounded number of time frames of a sequential circuit is checked against a specification often in terms of an *LTL*-property [11]. However, practically BMC is used for identifying functional failures rather than proving the absence of those. Efficient SAT-solver are used to solve a BMC instance leading to a practically efficient technique. Given a *Finite State Machine* (FSM) $M = (I, T, S)$ with (1) I being a predicate of the initial state, (2) T being the transition relation, and (3) S being the state space as well as given P as a temporal property, the BMC problem is formulated as follows:

$$\mathsf{BMC} = I(s_0) \wedge \bigwedge_{0 \leq i \leq l} T(s_i, s_{i+1}) \wedge P(s_l)$$

The variable l defines the number of time frames to be unrolled. To cover all reachable states, l needs to be set to the sequential depth of the FSM – often not practically feasible. Hence, l is usually chosen with a proper value by the designer. Therefore, a trade-off between run time and accuracy is employed. To solve the generated BMC instance by a SAT-solver [12], the instance has to be unrolled starting from $l = 0$ up to the designer-defined threshold. Translating a BMC problem into a satisfiability instance is done by unrolling the circuit up to a certain time frame applying the technique of [37]. The resulting formula can be translated into a CNF in polynomial time with respect to the number of variables and operators using the encoding proposed in [30][4].

Once the instance becomes satisfiable, the property is disproved and a counterexample can be extracted from the SAT-instance. Otherwise, the property holds with respect to l. As initially mentioned, this BMC technique is utilized by the State Collector. In particular, solutions for the BMC problem [11] are used to determine the specific states, which describes the actual translation of the LTL properties into a satisfiability problem. In general, BMC is about determining a path of states $s_0 \ldots s_l$ from the initial state s_0 so that, eventually, the terminal state s_l violates or satisfies a certain property.

For our purposes, we revise this BMC formulation in order to determine a path of states so that, eventually, the EP holds for the currently considered partition P_j. More precisely,

$$\mathsf{SFind}(P_j, l) = I(s_0) \wedge \bigwedge_{0 \leq i < l} T(s_i, s_{i+1}) \wedge \mathcal{P}$$

is employed where $\mathcal{P}$ is a logical formula modeling $\mathsf{EP}(s_l, P_j)$. The formula SFind is satisfiable, if there is at least one path $s_0 \ldots s_l$ such that all FFs in the currently considered partition P_j assume the same output value at state s_l. The number l of transitions to be considered, i.e., the unrolling depth of the circuit, can be iteratively increased until either a state s_l has been determined or the maximal unrolling depth (defined by the designer) is reached. The selection of the maximal unrolling depth l while keeping run times reasonable is a challenging task. To cover the entire search space, l needs to be aligned with the completeness threshold as stated in Algorithm 2. However, from a practical point of view, this

[4] Note that, in order to ease the description, we will not cover the details of the CNF conversion. This has already sufficiently been covered by previous work, e.g., in [30], [31].

is often unfeasible and, moreover, (mostly) not required at all. For many practical instances, getting a satisfactory result is possible without reaching the completeness threshold.

Please note that there are many techniques available, which are approximating the reachable state space of digital circuits and, hence, those techniques can significantly improve the run time [38]–[40]. However, they are not considered in this work to keep the presentation clear.

If no path can be determined, the partition P_j has been found unsuitable as no state could have been determined in which all FFs in P_j assume the same output value. In addition to that, another parameter $u > 0$ is added which prevents the State Collector from determining too many states. Considering too many states increases significantly the complexity of the FDM while hardly improving the achieved robustness anymore.

Overall, this leads to the State Collecting method as summarized in Algorithm 2. The algorithms receive the partition P_j from the Partition Enumerator, the maximum number u of states to be generated, as well as the unrolling depth l for the underlying BMC problem. The collected states $\widehat{S}$ are compactly represented by means of BDDs [13] and initialized by the empty set in Line 1.

As long as the maximum number u of states to be determined is not reached (Line 5), further states are computed. This is done by formulating the BMC problem (Line 3) for the currently considered unrolling depth k $(0 < k < l)$. Afterward, the resulting formulation (denoted by F) is solved by a SAT-solver (Line 8). As long as a satisfying solution is determined, the corresponding states are added to $\widehat{S}$ (Line 6) and, afterward, blocked in the BMC formulation F so that new states can be determined (Line 7).

Example 2: Given the exemplary partition $P_j = \{FF_3, FF_4, FF_5\}$ as provided by the Partition Enumerator, states shall be determined so that all FFs in P_j assume the same output value. Assuming that both cones f_1 and f_2 in the circuit from Fig. 1 do not contain any FFs, a State Collector instance according to SFind is formulated. Solving this instance yields a satisfying solution where all FFs $\{FF_3, FF_4, FF_5\}$ have the same output value '0'. From that, it is shown that $\mathsf{EP}(\widehat{S}, P_j)$ holds under the state $s \in \widehat{S}$ defined by $FF_1 = 0$ and $FF_2 = 0$. Consequently, the partition P_j is valid. In the Partition Enumerator (Algorithm 1), this EP is later stored in the container $\mathcal{E}$.

VI. FDM Synthesizer

The methods from above yield a data container $\mathcal{E}$ including all application-specific knowledge which has been obtained in terms of $\mathsf{EP}s$, i.e., all valid partitions P_j and corresponding states $\widehat{S}$ which satisfy the equivalence property. This knowledge is now utilized in order to synthesize an FDM. More precisely, for each determined $\mathsf{EP}(\widehat{S}, P_j) \in \mathcal{E}$, a fault signal $\mathcal{F}$ is to be generated which is set to '1' whenever the circuit is in a state $s \in \widehat{S}$ (checked by the activator) and, at the same time, the FFs in the partition P_j do not assume the same value (checked by the comparator). In the following, details on the realization of this FDM are provided.

1) Activator $\mathcal{A}$: For a given $\mathsf{EP}(\widehat{S}, P_j)$, a signal $\mathcal{A}$ has to be created which is set to '1' iff the circuit is in a state $s \in \widehat{S}$. As mentioned before in Section V, all currently relevant states $\widehat{S}$ are stored in terms of a BDD. Hence, corresponding logic triggering the signal $\mathcal{A}$ can easily be derived from the BDD by replacing all BDD nodes with a corresponding MUX gate (as, e.g., shown in [41]).

Besides that, the timing of $\mathcal{A}$ has to be properly adjusted. Transient faults are assumed to occur in the transition between two consecutive states. This is why the states $s \in \widehat{S}$ are collected for state s_{l-1} (assuming their effects manifest in state s_l). Consequently, the check for states has to be conducted one state before the values of all FFs in P_j are to be compared, i.e., the activator signal $\mathcal{A}$ has to be generated one state before the comparison is conducted. This requires signal $\mathcal{A}$ to be buffered for one cycle, which is accomplished by introducing an additional FF L-Act$_1$.

However, since L-Act$_1$ is vulnerable to transient faults, robustness is not guaranteed anymore. Hence, a second FF L-Act$_2$ which also receives the value of signal $\mathcal{A}$ is introduced. After one cycle, the output values of both FFs L-Act$_1$ and L-Act$_2$ are checked for equivalence. If the two values are not equal, a fault is reported (by setting the fault signal $\mathcal{F}$ to '1').

Example 3: Consider again the running example with the circuit from Fig. 1 and the determined $\mathsf{EP}(\widehat{S}, P_j)$. Fig. 3 shows the resulting circuit created by the FDM scheme proposed in this work. The bottom left corner of Fig. 3 sketches the resulting *Activator* logic. More precisely, using $\widehat{S}$ represented as BDD, a MUX circuit is created which generates the signal $\mathcal{A}$ (sketched by the block *State Collector*). Then, the resulting signal is passed to the two FFs L-Act$_1$ and L-Act$_2$. To protect both newly inserted FFs against possible single transient faults, an XOR gate is introduced. In case of a deviation between both FFs L-Act$_1$ as well as L-Act$_2$, the *Activator Fault* signal is triggered, which, especially, overdrives the fault signal $\mathcal{F}$ to indicate an occurred fault.

2) Comparator $\mathcal{C}$: For a given $\mathsf{EP}(\widehat{S}, P_j)$, a signal $\mathcal{C}$ has to be created which is set to '1' iff the FFs in a partition P_j assume the same values. This can easily be realized by connecting the corresponding FF outputs by XNOR gates and comparing their result. Since a further fan-out is introduced to the specific FF, the timing is slightly affected, which can easily be addressed by state-of-the-art techniques of the power optimization and, hence, is negligible.

The example illustrates the resulting logic.

Example 4: Consider again the running example with the circuit from Fig. 1 and the determined $\mathsf{EP}(\widehat{S}, P_j)$. The bottom middle part of Fig. 3 sketches the resulting *Comparator* logic. Here, the outputs of all FFs $\{FF_3, FF_4, FF_5\} \in P_j$ are compared by XNOR gates. Afterward, the outputs of these XNOR gates are passed to an AND gate. If all FFs assume the same value, this AND gate evaluates to '1'.

3) Generating the Fault Signal $\mathcal{F}$: Finally, the signals $\mathcal{A}$ and $\mathcal{C}$ are assembled into a single FDM that generates the fault signal $\mathcal{F}$. Recall that $\mathcal{F}$ is set to '1' iff a fault has been detected. For a given $\mathsf{EP}(\widehat{S}, P_j)$, this is the case if the circuit just left a state $\widehat{S}$ (stored in the both FFs $L{-}Act_1$ and $L{-}Act_2$, which drive the signal $\mathcal{A}$)[5] and all FFs in $\mathcal{P}_|$ are not equal. This is described by $\mathcal{F} = \neg\mathcal{C} \wedge \mathcal{A}$ and, hence, an occurred transient fault can be realized easily within the later application.

Example 5: Consider again the running example and the resulting circuit shown in Fig. 3. As can be seen in the bottom right corner, the signal $\mathcal{C}$ is first inverted and, afterward, ANDed with the $\mathcal{A}$ signal. The resulting value is additionally ORed with the fault value from the robustness check of the *Activator* – eventually resulting in the desired signal $\mathcal{F}$. Consider now the entire circuit and, e.g., a transient fault in FF_3 (denoted by the red strike symbol). This causes the FFs $\{FF_3, FF_4, FF_5\} \in P_j$ to not assume the same value anymore in states $\widehat{S}$ where this is supposed to happen, i.e., the EP$(\widehat{S}, P_j)$ fails. This case is propagated through the FDM (see annotations in Fig. 3) which, eventually, sets the fault signal $\mathcal{F}$ to '1', and, by this, detects the fault.

Logic as described above is, of course, added for all EP $\in \mathcal{E}$. Overall, this leads to a circuit which has a slightly increased number of gates but substantially improved robustness. This has been confirmed by experimental evaluations whose results are summarized next.

VII. Experimental Results

This section describes the experiments, which have been conducted to evaluate the proposed approach. At first, the experimental setup is presented and, secondly, the results are shown and discussed.

A. Setup

The proposed methodology has been implemented in C++. To determine the non-robust FFs of the circuit, a simulation-based robustness checker has been implemented which transforms the given circuits (provided in *Verilog* and parsed by *Verific*) into a compiled simulation model (to this end, *LLVM* [42] *IR code* is generated by the simulation environment). In order to conduct the respective BMC instance (cf. Section V) and the SAT-based partitioning approach (cf. Section IV-B), *MiniSAT* [12] on top of *metaSMT* [43], together with the *X-value abstraction* as described in [44] has been utilized. The BDD package *CUDD* has been used to generate the MUX circuits.

The resulting flow has been evaluated using *ITC'99* benchmark circuits. The robustness of a circuit has been introduced in Section II-C.

In order to determine the set of all non-robust FFs (cf. Section II-C), the parameters $l = 500$ and $r = 5$ for simulation have been found suitable for these circuits according to the (minimal) latency analysis of [45]. The *Partition Enumerator* (cf. Section IV) considered different maximal partition sizes holding maximal $p_s \in \{8, 16\}$ FFs.

Finally, the *State Collector* (cf. Section V) always assumed an unrolling depth of $l = 10$ and a bounded number $u = 1024$ of states to be collected per partition P_j[6]. The selection of the unrolling depth l is aligned with the maximal latency, which is determined for the simulation-based approach in work [45] and acts as an upper bound for the applied formal BMC model. The developed flow seamlessly integrates both, the random as well as the SAT-based technique. This allows the designer to easily select a trade-off between the overall robustness improvement with respect to the introduced hardware overhead.

TABLE I: Run time for different $p_s \in \{8, 16\}$

circ.	**#gates**	**#FFs**	$\mathcal{P}_{\mathsf{RAND}}$ **run time [m]**		$\mathcal{P}_{\mathsf{SAT}}$ run time [m]	
			$p_s = 8$	$p_s = 16$	$p_s = 8$	$p_s = 16$
b05	608	66	< 0.10	< 0.10	< 0.10	0.15
b06	66	9	< 0.10	< 0.10	< 0.10	< 0.10
b07	382	51	0.18	0.18	2.96	44.03
b08	168	21	< 0.10	< 0.10	< 0.10	< 0.10
b09	131	28	< 0.10	< 0.10	2.88	11.25
b10	172	17	< 0.10	< 0.10	0.26	17.18
b11	366	30	< 0.10	< 0.10	1.50	0.28
b12	1000	121	1.26	1.15	0.71	0.911
b13	309	53	< 0.10	0.11	21.98	791.67

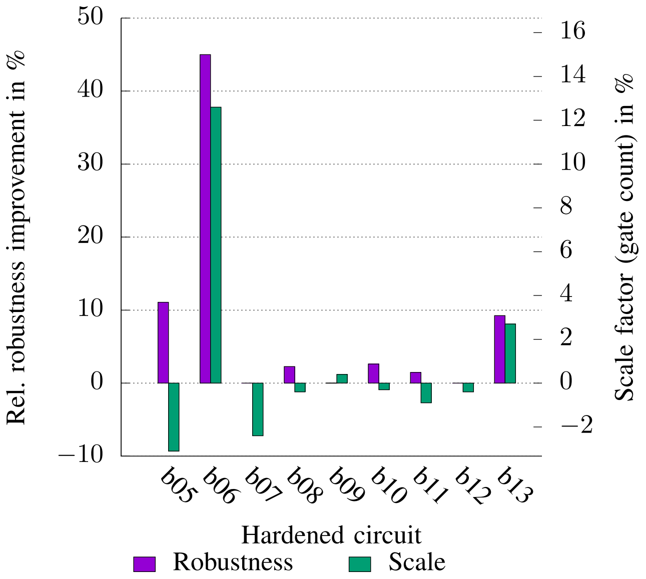

Fig. 4: Comparison between random-based and SAT-based approach

All evaluations have been conducted on an *Intel Xeon E5-2640v4 2.4 GHz* processor with 256GB system memory. The *time-out* (TO) is set to 96h and for the *memory-out* (MO) is set to 64GB.

B. Evaluation

The obtained results are summarized as follows:

- Table I provides details on the considered benchmark circuits, i.e., its respective name, number of gates, and number of FFs, as well as the run time (in CPU minutes)

[5] Note that this FF stage introduces a required delay of one clock cycle.

[6] Note that this bound u was not exceeded.

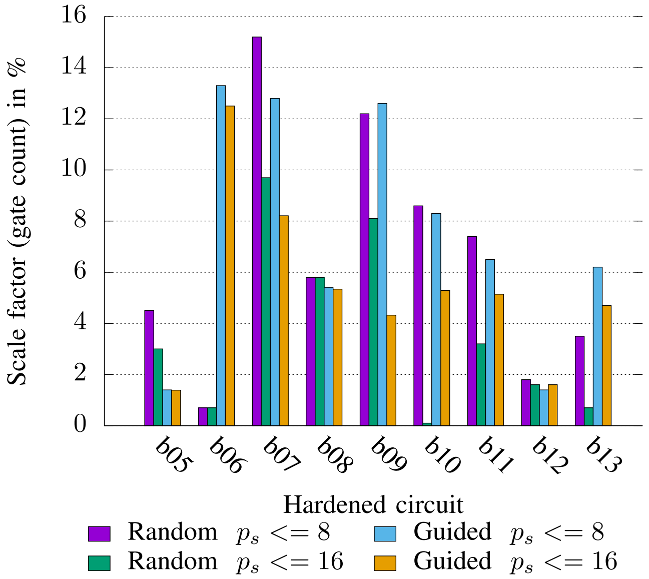

Fig. 5: Hardware overhead for random and guided technique with $p_s \leq \{8, 16\}$

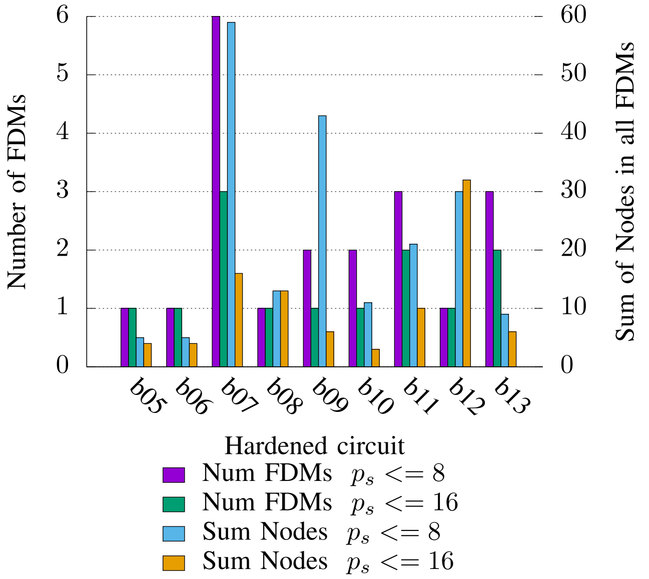

Fig. 6: Number and size of introduced FDMs (by SAT-based technique)

required by both proposed methodologies when maximal partition sizes of $p_s \in \{8, 16\}$ are applied.

- Fig. 4 presents a direct comparison between the random-based and SAT-based approaches: The right X-axis shows the difference $\mathcal{H}_{SG} - \mathcal{H}_{RB}$ of the random-based scale factor $\mathcal{H}_{RB}$ and of the SAT-based scale factor $\mathcal{H}_{SG}$. Thus, the lower the *scale* bar chart is, the lower the introduced hardware overhead is while applying the SAT-based approach instead of the random-based one and vice versa. The resulting robustness improvement of the enhanced designs (by utilizing the proposed approaches) $\mathcal{R}_{RB}$ and $\mathcal{R}_{AG}$ are compared with respect to the design's initial robustness[7] $\mathcal{R}_{init}$ is determined as follows: $\Delta\mathcal{R}_{RB} = \mathcal{R}_{RB} - \mathcal{R}_{init}$ and $\Delta\mathcal{R}_{SG} = \mathcal{R}_{SG} - \mathcal{R}_{init}$, respectively. The left X-axis provides the difference of both robustness improvements: $\Delta\mathcal{R}_{SG} - \Delta\mathcal{R}_{RB}$. Thus, the higher the *robustness* bar chart is, the higher the robustness improvement is, which is achieved while applying the SAT-based approach instead of the random-based one and vice versa.

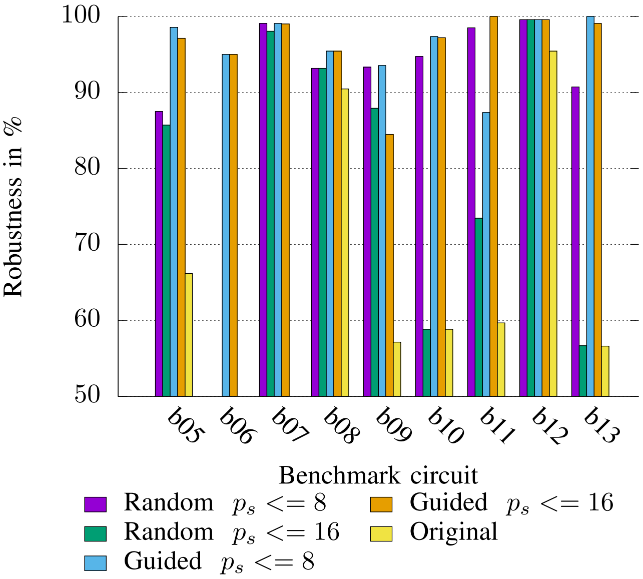

Fig. 7: Robustness Improvement for enhanced circuits

- Fig. 5 shows the hardware overhead (in terms of a gate count in percentage) caused by applying the proposed methodologies, i.e., one random-based as well as SAT-based approach, for the considered maximal partition sizes.[8]
- Fig. 6 presents on the left X-axis the overall number of introduced FDMs into the circuit by applying the SAT-based approach for both maximal partition sizes. Furthermore, the right X-axis shows the overall nodes of *all* state collectors. Note that each FDM holds exactly one individual state collector. Besides this, note that the state collector is represented as a BDD, which can be synthesized by a one-to-one mapping between nodes and MUX-gates as stated in [41].
- Fig. 7 shows the robustness of the original circuit as

[7] Note that the initial robustness is due to the circuit's structure and, more precisely, due to the implicitly given redundancies as observed in [46].

[8] If no bar is shown, a hardware overhead of 0% or close to 0% is measured.

well as the robustness after applying both proposed methodologies (again for different maximal partition sizes $p_s \in \{8, 16\}$).

First, Figure 4 presents a direct comparison between the random-based and the SAT-based approach. The SAT-based approach allows reducing the introduced hardware-overhead in two third of the conducted experiments while achieving at least the same (or even a slightly higher) robustness enhancement. In case of benchmark circuits *b06* (*b13*), the newly proposed SAT-based approach clearly shows its advantages against the random-based one: The robustness is even more increased by approx. 55% (10%).

Besides that, the results nicely show the effect of different partition sizes p_s as stated in Figure 5: In almost all cases a larger p_s leads to a smaller hardware overhead. This is because larger partitions cover more FFs and, hence, require the consideration of a smaller total number of partitions leading to less FDM logic. This is both valid for the random-based as well as the SAT-based approach.

Particularly, this observation can also be validated for the SAT-based approach by considering Figure 6: The number of introduced FDMs is lower equal when invoking the SAT-based technique with $p_s \leq 8$ instead of $p_s \leq 16$. For instance, we see 3 (2) FDMs being introduced to the circuit b11 with p_s equals to 8 (16). This leads to overall 21 (10) nodes in the generated state collectors.

At the same time, this reduces the required run time since less BMC checks have to be conducted. Both methodologies are slightly able to improve the robustness for b08 only. Moreover, the hardware overhead is even the same.

However, more important is the overall performance. In this regard, the proposed methodology provides a suitable alternative to previously proposed solutions such as discussed in Section I. Although space-based approaches such as TMR [2] can guarantee 100% robustness, they usually require more than thrice the amount of hardware (i.e., yielding a scaling factor of > 3.0). In contrast, the solution proposed in this work is capable of always improving the robustness to more than 90% (in some cases even close to 100%), while only 5.8% more hardware is required for this in average. As already discussed before, the proposed solution also outperforms timing-based and application-specific approaches, since timing is hardly affected at all in the proposed solution and the methodology can be applied to arbitrary sequential circuits. By this, a suitable trade-off between enhancing the robustness and keeping the hardware overhead small is achieved.

Moreover, the determination of *good* partition is a computational hard task: While considering structural information as done by the SAT-based approach, the run time is increased. The run times are manageable, even though common ATPG techniques, e.g., modeling J-frontiers and D-chain-based propagation, have not been applied.

The proposed SAT-based approach tackles this challenge by exploiting powerful formal model, which is inspired by SAT-based ATPG techniques, leading to:

1) An enhanced robustness in most of the cases compared to the random-based approach and has never been reduced,
2) the scale factor is in almost all cases reduced compared to the random-based approach or the robustness [cf. 1)] has been significantly enhanced.

Additionally, for the SAT-based approach a trade-off between the effort and the benefit can be adjusted on designer's choice.

VIII. Conclusions & Future Work

In this work, we proposed an approach for improving the robustness of sequential circuits. The main idea is to avoid an hardware overhead, which often introduces unnecessary redundancy. The proposed approach exploits application-specific knowledge about the FFs in each reachable state. To this end, a methodology is introduced, which gains the corresponding knowledge and, afterward, utilizes them for a fault detection mechanism. To cope with the underlying complexity, a dedicated orchestration of formal techniques is employed. This results in a hardening method which requires only a slight increase in additional hardware, does only influence the timing behavior negligibly by introducing just one further fan-out to the FFs, and is automatically applicable to arbitrary circuits. Experimental evaluations confirmed these benefits: Robustness can be increased to approx. 84% (97%), while the circuit size increases only by a factor of approx. 1.07 (1.07) on average while applying the random-based (SAT-based) approach. Future work will focus on developing a technique for not only detecting the respective faults but also correcting them with the proposed methodology. Furthermore, a significant performance improvement can be achieved by introducing a K-induction for the state-collector, which would allow representing the collected stats in an exact and non-symbolic manner.

Future work will focus on applying the proposed technique to large (industrial-sized) circuits. One method towards this is the consideration of a compositional approach. This technique is often applied to reduce the search-space when employing formal methods. The compositional approach follows the idea to pick parts of the circuits, e.g., provided by module-bounds of a hardware description language. Then, the technique proposed here can be applied to harden these parts. Since manageable parts are hardened separately, the search space is significantly reduced compared to the entire circuit. However, the number FFs fulfilling the EP condition will potentially grow since more states will lead to the EP condition than actually possible considering the entire circuit. But the FDM integrated into the entire circuit will not cause false-positive since only reachable states occur in operation.

IX. Acknowledgment

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 276397488 – SFB 1232 in subproject P01 'Predictive function'. We would like to thank Verific Design Automation Inc. for providing the SystemVerilog front-end used for the implementation of our technique.

References

[1] F. L. Kastensmidt, L. Carro, and R. Reis, *Fault-Tolerance Techniques for SRAM-Based FPGAs*. Springer, 2006.

[2] C. E. Stroud and A. E. Barbour, "Design for testability and test generation for static redundancy system level fault-tolerant circuits," in *Int'l Test Conference*, 1989, pp. 812–818.

[3] Z. Luo, "ECC, an extended calculus of constructions," in *Int'l Symposium on Logic in Computer Science*, 1989, pp. 386–395.

[4] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[5] A. Sanchez-Macian, P. Reviriego, and J. A. Maestro, "Hamming SEC-DAED and extended hamming SEC-DED-TAED codes through selective shortening and bit placement," *IEEE Trans. Device and Materials Reliability*, vol. 14, no. 1, pp. 574–576, 2014.

[6] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," in *Microarchitecture*, 2003, pp. 7–18.

[7] D. Blaauw, S. Kalaiselvan, K. Lai, W. H. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull, "Razor II: in situ error detection and correction for PVT and SER tolerance," in *IEEE Int'l Conference on Solid-Sate Circuits*, 2008, pp. 400–622.

[8] N. Farazmand, M. Fazeli, and S. G. Miremadi, "FEDC: control flow error detection and correction for embedded systems without program interruption," in *Int'l Conf. on Availability, Reliability and Security*, 2008, pp. 33–38.

[9] S. Hertz, D. Sheridan, and S. Vasudevan, "Mining hardware assertions with guidance from static analysis," *IEEE Trans. on CAD of Integrated Circuits & Systems*, vol. 32, no. 6, pp. 952–965, 2013.

[10] S. Huhn, S. Frehse, R. Wille, and R. Drechsler, "Enhancing robustness of sequential circuits using application-specific knowledge and formal methods," in *ASP Design Automation Conference*, 2017, pp. 182–187.

[11] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 1999, ch. Symbolic Model Checking without BDDs, pp. 193–207.

[12] N. Eén and N. Sörensson, *Int'l Conf. on Theory and Applications of Satisfiability Testing*. Springer, 2004, ch. An Extensible SAT-solver, pp. 502–518.

[13] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. on CAD of Integrated Circuits & Systems*, vol. C-35, no. 8, pp. 677–691, 1986.

[14] T. Heijmen and A. Nieuwland, "Soft-error rate testing of deep-submicron integrated circuits," in *IEEE European Test Symposium*, 2006, pp. 247–252.

[15] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[16] U. Krautz, M. Pflanz, C. Jacobi, H. W. Tast, K. Weber, and H. T. Vierhaus, "Evaluating coverage of error detection logic for soft errors using formal methods," in *Design, Automation and Test in Europe*, vol. 1, 2006, pp. 1–6.

[17] L. Doyen, T. A. Henzinger, A. Legay, and D. Nickovic, "Robustness of sequential circuits," in *Int'l Conference on Application of Concurrency to System Design*, 2010, pp. 77–84.

[18] G. Fey, A. Sülflow, S. Frehse, and R. Drechsler, "Effective robustness analysis using bounded model checking techniques," *IEEE Trans. on CAD of Integrated Circuits & Systems*, vol. 30, no. 8, pp. 1239–1252, 2011.

[19] G. Fey and R. Drechsler, "A basis for formal robustness checking," in *Int'l Symposium on Quality Electronic Design*, 2008, pp. 784–789.

[20] Shi-Yu-Huang, K.-T. Cheng, K.-C. Chen, and J. Y. J. Lu, "Fault-simulation based design error diagnosis for sequential circuits," in *Design Automation Conference*, 1998, pp. 632–637.

[21] N. Miskov-Zivanov and D. Marculescu, "Multiple transient faults in combinational and sequential circuits: A systematic approach," *IEEE Trans. on CAD of Integrated Circuits & Systems*, vol. 29, no. 10, pp. 1614–1627, 2010.

[22] M. Bozzano, A. Cimatti, and F. Tapparo, *Symbolic Fault Tree Analysis for Reactive Systems*. Springer, 2007, ch. ATVA, pp. 162–176.

[23] D. Nayak and D. M. H. Walker, "Simulation-based design error diagnosis and correction in combinational digital circuits," in *IEEE VLSI Test Symposium*, 1999, pp. 70–78.

[24] A. Miczo, *Digital Logic Testing and Simulation*. Wiley, 2003.

[25] T. Larrabee, "Test pattern generation using boolean satisfiability," *IEEE Trans. on CAD of Integrated Circuits & Systems*, vol. 11, no. 1, pp. 4–15, 1992.

[26] R. Drechsler, S. Eggersglüß, G. Fey, and D. Tille, *Test Pattern Generation Using Boolean Proof Engines*, 1st ed. Springer, 2009.

[27] S. Eggersglüß, R. Wille, and R. Drechsler, "Improved SAT-based ATPG: More constraints, better compaction," in *Int'l Conference on CAD*, 2013, pp. 85–90.

[28] J. P. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM Journal of Research and Development*, vol. 10, no. 4, pp. 278–291, 1966.

[29] A. Biere, M. Heule, H. Maaren, and T. Walsh, *Handbook of Satisfiability*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009, vol. 185.

[30] G. Tseitin, *On the Complexity of Derivation in Propositional Calculus*. Springer, 1983, vol. 8, pp. 466–483.

[31] P. Manolios and D. Vroon, "Efficient circuit to cnf conversion," *Theory and Applications of Satisfiability Testing*, pp. 4–9, 2007.

[32] D. Tille, S. Eggersgluss, and R. Drechsler, "Incremental solving techniques for SAT-based ATPG," *IEEE Trans. on CAD of Integrated Circuits & Systems*, vol. 29, no. 7, pp. 1125–1130, 2010.

[33] S. Disch and C. Scholl, "Combinational equivalence checking using incremental SAT solving, output ordering, and resets," in *ASP Design Automation Conference*, 2007, pp. 938–943.

[34] C. Loiacono, M. Palena, P. Pasini, D. Patti, S. Quer, S. Ricossa, D. Vendraminetto, and J. Baumgartner, "Fast cone-of-influence computation and estimation in problems with multiple properties," in *Design, Automation and Test in Europe*, 2013, pp. 803–806.

[35] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *Int'l Conference on Software Engineering*, 1999, pp. 411–420.

[36] A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu, "Symbolic model checking using sat procedures instead of BDDs," in *Design Automation Conference*, 1999, pp. 317–320.

[37] H. Konuk and T. Larrabee, "Explorations of sequential atpg using boolean satisfiability," in *IEEE VLSI Test Symposium*, 1993, pp. 85–90.

[38] O. Grumberg, T. Heyman, N. Ifergan, and A. Schuster, "Achieving speedups in distributed symbolic reachability analysis through asynchronous computation," in *Correct Hardware Design and Verification Methods*. Springer, 2005, pp. 129–145.

[39] K. L. McMillan, "Interpolation and SAT-based model checking," in *Computer Aided Verification*. Springer, 2003, pp. 1–13.

[40] K. Ravi and F. Somenzi, "High-density reachability analysis," in *Int'l Conference on CAD*, 1995, pp. 154–158.

[41] R. Drechsler, J. Shi, and G. Fey, "Synthesis of fully testable circuits from BDDs," *IEEE Trans. on CAD of Integrated Circuits & Systems*, vol. 23, no. 3, pp. 440–443, 2004.

[42] C. Lattner and V. Adve, "LLVM: a compilation framework for lifelong program analysis transformation," in *Int'l Symp. on Code Generation and Optimization*, 2004, pp. 75–86.

[43] H. Riener, F. Haedicke, S. Frehse, M. Soeken, D. Große, R. Drechsler, and G. Fey, "metaSMT: focus on your application and not on solver integration," *Software Tools for Technology Transfer*, pp. 1–17, 2016.

[44] O. Grumberg, *3-Valued Abstraction for (Bounded) Model Checking*. Springer, 2009, pp. 21–21.

[45] A. Finder, A. Sülflow, and G. Fey, “Latency analysis for sequential circuits,” in *IEEE European Test Symposium*, 2011, pp. 129–134.

[46] D. Tille and R. Drechsler, “A fast untestability proof for sat-based ATPG,” in *IEEE Symposium on Design and Diagnosis of Electronic Circuits and Systems*, 2009, pp. 38–43.

Sebastian Huhn is currently pursuing his Ph.D. degree at the Group of Computer Engineering, University of Bremen, Germany. Sebastian Huhn received his bachelor degree in 2012 and his master degree in 2014 both in computer engineering from the University of Bremen. Besides this, he is a researcher at the German Research Center for Artificial Intelligence (DFKI). His research interests include test interfaces, formal methods, in particular, formal solving techniques, and pattern retargeting as well as reliability analysis or enhancement of circuits. He has been in the Program Committee of the International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS) in 2018.

Stefan Frehse Stefan Frehse received the Diploma degree in computer science from the University of Bremen, Germany, in 2009. After his research exchange with IBM Haifa in Israel, he received his Dr.-Ing. degrees from the University of Bremen focusing on formal verification of fault-tolerant systems. Since then he is working as an external researcher at the German Research Center for Artificial Intelligence (DFKI). His research interests are formal verification and robust distributed systems.

Robert Wille is a Full Professor at the Johannes Kepler University Linz. He received the Diploma and Dr.-Ing. degrees in computer science from the University of Bremen, Germany, in 2006 and 2009, respectively. He was with the Group of Computer Architecture at the University of Bremen and with the German Research Center for Artificial Intelligence (DFKI). Additionally, he worked as Lecturer at the University of Applied Science in Bremen, Germany, and as a visiting professor at the University of Potsdam, Germany, and the Technical University Dresden, Germany. His research interests are in the design of circuits and systems for both conventional and emerging technologies with a focus in the domain of synthesis and verification. Since 2007, he published more than 200 journal and conference papers in this area and was repeatedly awarded (e.g. with a Best Paper Award at ICCAD in 2013 and FDL in 2010). Besides that, he served Guest Editor for the ACM’s JETC, Springer’s LNCS, and the MVLSC. Additionally, he was PC Chair at ISMVL, FDL, and others as well as a frequent PC member and track chair for conferences such as ASP-DAC, DAC, DATE, ICCAD, ISMVL, and more.

Rolf Drechsler (F’15) received the Diploma and Dr.Phil.Nat. degrees in computer science from J. W. Goethe University Frankfurt am Main, Frankfurt am Main, Germany, in 1992 and 1995, respectively. He was with the Institute of Computer Science, Albert-Ludwigs University, Freiburg im Breisgau, Germany, from 1995 to 2000, and with the Corporate Technology Department, Siemens AG, Munich, Germany, from 2000 to 2001. Since 2001, he has been with the University of Bremen, Bremen, Germany, where he is currently a Full Professor and the Head of the Group for Computer Architecture, Institute of Computer Science. In 2011, he became the Director of the Cyber-Physical Systems Group, German Research Center for Artificial Intelligence, Bremen. His current research interests include the development and design of data structures and algorithms with a focus on circuit and system design. Prof. Drechsler was a recipient of the best paper awards at the Haifa Verification Conference in 2006, the Forum on Specification & Design Languages in 2007 and 2010, the IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems in 2010, and the IEEE/ACM International Conference on Computer-Aided Design in 2013. He was a member of Program Committees of numerous conferences including Design Automation Conference (DAC), International Conference on Computer-Aided Design, Design, Automation and Test in Europe (DATE), Asia and South Pacific Design Automation Conference, Forum on specification and Design Languages, MEMOCODE, and Formal Methods in Computer-Aided Design, the Symposiums Chair ISMVL 1999 and 2014, and the Topic Chair for “Formal Verification” DATE 2004, DATE 2005, DAC 2010, as well as DAC 2011. He is a Co-Founder of the Graduate School of Embedded Systems and the Coordinator of the Graduate School “System Design” funded within the German Excellence Initiative.