Analysis of Reachable Sensitisable Paths in Sequential Circuits with SAT and Craig Interpolation

Matthias Sauer* Stefan Kupferschmid* Alexander Czutro* 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 Iowa
  5324 Seamans Center
  Iowa City, IA 52242, United States
  reddy@engineering.uiowa.edu

Abstract—Test pattern generation for sequential circuits benefits from scanning strategies as these allow the justification of arbitrary circuit states. However, some of these states may be unreachable during normal operation. This results in non-functional operation which may lead to abnormal circuit behaviour and result in over-testing.

In this work, we present a versatile approach that combines a highly adaptable SAT-based path-enumeration algorithm with a model-checking solver for invariant properties that relies on the theory of Craig interpolants to prove the unreachability of circuit states. The method enumerates a set of longest sensitisable paths and yields test sequences of minimal length able to sensitise the found paths starting from a given circuit state. We present detailed experimental results on the reachability of sensitisable paths in ITC 99 circuits.

I. INTRODUCTION

Scan-based manufacturing tests are universally used to screen out defective VLSI devices. Scan allows to justify arbitrary circuit states that may be unreachable and thus do not occur during functional operation of the circuits under test. Non-functional operation during test is known to not only cause abnormal switching activity, which causes abnormal power dissipation and supply-voltage droops, but also to sensitise non-functional paths [1]. These effects may lead to yield loss as good devices could fail test [2].

Several works have proposed methods for the generation and application of tests that introduce lower switching activity during test [3]. One way to avoid non-functional operation during scan-based test is to scan in only states that are reachable from the reset state or from a state reached after circuit synchronisation [4]. A simulation-based test generation procedure to compute such tests for detection of transition-delay faults was investigated in [5] and a sequential ATPG-based procedure was proposed in [6]. Other previous work on reachability in sequential circuits is often based on probabilistic methods. [7] and [8] combine random simulation with a BDD-based approach to compute the complete set of reachable states. However, these methods scale poorly as the possible search space grows exponentially with the number of flip-flops.

Furthermore, it is not possible to directly obtain the necessary assignments required to get into a reachable state. In [9] a genetic algorithm is used for sequential ATPG. However, due to the randomness of evolutionary approaches, such methods are heuristic and hence neither complete nor optimal.

Tests for delay faults are used to address several goals of manufacturing test of VLSI devices. These include determining the maximum frequency of operation using tests for several selected critical paths, and detection of small-delay defects by testing for gate-delay faults through longest paths [10].

In this work, we consider the generation of tests for delay faults using a versatile approach based on recent advances in SAT-based test generation. Thanks to sophisticated learning strategies that have been incorporated into modern SAT-solvers in the last few years, the classification of hard-to-detect and identification of redundant faults has recently been proved to be managed by SAT-based tools in a more efficient way than classical structural ATPG [11], [12], [13]. Also, the straightforward formalism of SAT-based problem formulation permits an easy combination of SAT-based approaches with other optimisation approaches [14], [15].

However, previous related SAT-based approaches for the test of sequential delay faults [16], [17] can not provide an unreachability proof as the search is executed until an upper bound is reached.

We present an approach that takes all these factors into consideration. Given a small delay fault in a sequential circuit with scanning capabilities, we derive a two-pattern test using the SAT-based path-enumeration tool PHAETON [18]. Then, the approach attempts to determine whether the computed assignment to the flip-flops represents a reachable state.

This is done by formulating the problem as a model-checking problem (MC) which is passed to the CIP-solver (Craig Interpolation Prover) [19]. CIP has been originally designed for formal verification problems and matches the performance of the best pure Bounded-Model-Checking(BMC)-solvers, while providing competitive performance when proving safety properties, i.e. proving that a certain (unsafe) state is not reachable. In this work, CIP is used for the first time in the context of test.

Furthermore, given a set of circuit paths, the approach can determine the functional sensitisability of every path starting at reachable states. It can also determine if the paths are robustly...
or non-robustly testable starting at reachable states. In addition, it is complete and therefore generates tests when they exist. For the results we report, we consider functional sensitisation. If a path is not functionally sensitisable, then it is known as a false path independently of circuit delays [20]. Also, the method is able to provide optimal test sequences. Here, by test sequence we mean a sequence of assignments for the primary circuit inputs such that the sequence constitutes a test for a target path with respect to a given initial state. An optimal test sequence is a test sequence of minimal length.

The remainder of the paper is structured as follows. A brief overview of SAT-based path enumeration and Craig-interpolant-based model checking is provided in Section II. The combination of both tools is introduced in Section III. Experimental results are reported in Section IV. Section V concludes the paper.

II. PRELIMINARIES

In this section, a brief overview of SAT-based path enumeration (PHAETON) and Craig-interpolant-based model checking (CIP) is provided.

A. PHAETON

The inputs of PHAETON are a circuit (gate-level net list) and a list of gates to be analysed. For every gate in the list, the method searches for the longest sensitisable paths that pass through that gate.

Only complete paths are considered, i.e. paths that start at an input and end at an output. A path is defined as functionally sensitisable if there is a test pattern pair that produces a rising or a falling transition at all gate outputs along the path. The length of a path is defined as the sum of the delays assigned to its gates according to the employed delay model. The longest sensitisable path is therefore the path with the maximum delay.

The supported delay model assigns every gate a fixed integer delay that depends on which gate input is the on-path input (pin-to-pin), and whether on the transition at the gate’s output is a rising or a falling one. Since the tool works with integer delays only, the real-valued delays used in the experimental setup are mapped to integer delays between 1 and a user-defined constant $\alpha$ called the delay resolution. In order to minimise the global path error, a sophisticated rounding strategy is employed.

A given target gate $G$ is processed in two stages. In the first stage, the length $L_G$ of the longest sensitisable path through the gate is measured by means of iterative SAT-solving using a binary search over a search space that starts with the length of the shortest structural path and ends with the length of the longest structural path. Several learning strategies are employed to narrow down the search space.

After the application of the first stage to all target gates, the second stage can be applied in two different modes either to all gates or to a selection of gates depending on the application. This stage can either enumerate all sensitisable paths through the target gate $G$ that have a length between $L_G - r$ and $L_G$, for a user-specified parameter $r$ known as the length range; or it can enumerate the $K$ longest sensitisable paths through $G$ for a user-specified target number $K$ of paths.

The enumeration of all paths of a fixed length $l$ is done by formulating a SAT-instance $S_{G}[= l]$ that is satisfiable if and only if a sensitisable path through $G$ of length $l$ exists. If such a path is identified, i.e. if a Boolean solution is found, clauses are added to the SAT-instance such that it remains satisfiable only if a found sensitisable path of length $l$ differs from the first one. This is repeated until the SAT-instance becomes unsatisfiable, which means that there are no more sensitisable paths of that length. Please refer to [18] for details.

Furthermore, the tool is able to consider additional conditions imposed on the path enumeration algorithm, such as whether the sensitisation has to be robust or non-robust, and also conditions regarding the scanning technique (broadside, skewed-load, full enhanced scan, etc).

B. BMC and Craig interpolation

This section briefly introduces Bounded-Model-Checking (BMC) [21], Craig interpolants [22] and McMillan’s work on SAT-based model checking [14].

Among other applications, BMC is employed to derive error traces in sequential circuits that are required to satisfy a certain property $P$ but in fact do not satisfy this property. The circuit’s structure and the problem conditions are encoded as a propositional formula of the form

$$BMC_k = I_0 \land T_{0,1} \land \ldots \land T_{k-1,k} \land P_k$$

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

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 is unfolded until reaching its diameter. However, very large $k$-values may be reached. Hence, there are several approaches that attempt to find a fixed point sooner. Among others including $k$-induction [23] and BDD-based approaches [24], there are methods based on the theory of Craig interpolation [14]. This approach is used by the CIP-solver that we employ in this work.

A classical BMC approach searches for the smallest $k$-value for which the desired property holds by attempting to solve a series of problem instances. The first one is $BMC_0 = I_0 \land P_0$ (cf. Equation 1). It is satisfiable if the property holds in the initial state. If the instance is not satisfiable, BMC tests whether taking one more step into consideration will satisfy the property, i.e. whether the formula $BMC_1 = I_0 \land T_{0,1} \land P_1$ is satisfiable. This is repeated until $P_k$ holds for some $k$, or until a user-defined maximal bound is reached [21].

The drawback of this classical BMC approach is that it is not applicable for larger circuits, since it is not viable to attempt to solve as many instances as the circuit’s diameter, which can become very large even for medium-sized circuits.
In order to cope with this issue, McMillan introduced in [14] a modified MC-approach that extends a classical BMC-procedure by using Craig interpolants. The Craig interpolant represents an over-approximation of all reachable states after a certain number of transition steps. This over-approximation is recomputed by an iterative algorithm until a fixed point is reached.

Formally, Craig interpolants [22] 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 → C
3) C → ¬B

C is called a _Craig interpolant_ of A and B.

Figure 1 illustrates the MC-algorithm 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 $BMC_1 = I_0 \land T_{0,1} \land P_1$. If this instance is unsatisfiable, by Craig’s theorem, there is a formula $C_1$ such that $I_0 \land T_{0,1} \rightarrow C_1$ 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 $I_0 \land T_{0,1} \rightarrow C_1$, 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 so-called fixed-point check (FPC) is performed. It consists in checking 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. If FPC 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 $BMC_1$ 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 $BMC_1$ is unsatisfiable, the overall algorithm restarts by computing a new Craig interpolant parting from $BMC_1$. See [14] for details.

**III. REACHABILITY OF SENSITISABLE PATHS**

In this section, we introduce the overall flow that combines the SAT-based path-enumeration and the model-checking tool in order to calculate sets of reachable sensitisable paths. The overall data flow is illustrated in Figure 2.

The inputs of the problem are a sequential circuit (gate-level netlist along with timing data needed by PHAETON) and an assignment of Boolean variables to the flip-flops in the circuit, representing an initial circuit state $I$ (flip-flop contents after a system reset or after the application of a synchronising sequence). The aim of the algorithm is to find a set of longest reachable sensitisable paths, as well as test sequences that sensitise those paths starting at the given initial state. A sensitisable path is defined as _reachable_ if a test sequence of length $k$ exists for some $k \in \mathbb{N}$ such that the application of that test sequence to the circuit in state $I$ sensitises the path after $k$ time frames. If a test sequence of length $k$ is found for a path $p$, we call $k$ $p$’s _depth._

First, PHAETON is used to extract a set of longest sensitisable paths. Then, for each found path $p$, a MC-instance $MC(I,T_{i,i+1}, P)$ is generated and passed to the CIP-solver. If the MC-instance is determined to be unsatisfiable, $p$ is proved as unreachable. On the other hand, if the MC-instance is satisfiable, $p$ is reachable and there exists an appropriate test sequence that can be extracted from the Boolean solution computed by the CIP-solver.

An MC-instance consists of three predicates namely the initial state $I$, the transfer function $T_{i,i+1}$, and a certain property $P$. Figure 3 illustrates how these predicates are connected in order to check whether a functional sequence starting from the
initial state $I_0$ justifying the target property $P$ performing $k$ transition steps exits or not. We call $MC(I, T_{i+1}, P)$ satisfiable if there exists a functional sequence for some value $k$.

One arbitrary step of the sequential circuit from time point $i$ to time point $i+1$ corresponds to one application of the transfer function $T_{i+1}$. The transfer function is given by the SAT-representation of the circuit’s combinational logic that can be obtained by performing a so-called Tseitin-transformation [25]. A Boolean formula encoding multiple time frames of a sequential circuit is obtained by connecting the transfer function multiple times, i.e. $T_{0,1} \wedge T_{1,2} \wedge \ldots T_{k,k+1}$.

The initial state $I_0$ defines the starting state of the sequence and corresponds to the initial logical values of the flip-flops. Analogously, the target property $P$ imposes the justification requirements that need to hold at the end of the sequence. Note that our method supports justification conditions for internal circuit lines over multiple time frames.

The complete MC-instance is passed to the CIP-solver that returns a Boolean solution, or classifies the instance as unsatisfiable which means that no solution exists

Let $p$ be one of the sensitisable paths found by PHAETON. Since PHAETON is a SAT-based tool, the values assigned to the side inputs of the gates along $p$ can be extracted from the Boolean solution at no additional expense. These values comprise the set of necessary assignments needed to guarantee the sensitisation of the path. Note that this set of necessary assignments automatically respects additional conditions imposed on the path-enumeration algorithm.

The algorithm can be applied in three different modes that differ in the definition of the target property $P$. In the first mode, the desired property $P_k$ passed to the CIP-solver is composed of the necessary assignments extracted from the found path’s side inputs as explained above. This mode is called Path-Targeting Mode. In the second mode, the desired property $P_k$ is composed only of the values that have to be assigned to the flip-flops in the last-but-one time frame. That means, in this mode, the CIP-solver attempts to justify the first test pattern of the test pair found by PHAETON. This mode is called Pattern-Targeting Mode. The last mode, called Second-Pattern-Targeting Mode, is a combination of the other modes. While Path-Targeting is used for the first time frame, we use pattern-targeting for the second one. Therefore, the solver is free to change the first pattern to a functional one while keeping the second one unchanged.

If the MC-instance is determined to be unsatisfiable, $p$ is proved to be unreachable. On the other hand, if the MC-instance is satisfiable, $p$ is reachable and there exists an appropriate test sequence that can be extracted from the Boolean solution computed by the CIP-solver.

Note that, due to the iterative nature of the CIP-solver’s algorithm, the found sequence is guaranteed to have the shortest possible length. Also, the method is guaranteed to find a solution if one exists, provided that the CIP-solver is allocated unlimited solving time and memory.

We also developed a technique to speed up the algorithm by shortening the number of iterations required by the CIP-solver. At the beginning of the overall flow, only one initial circuit state $I$ is given. However, whenever the CIP-solver identifies a reachable path of depth $d$ greater than or equal to a user defined limit $d_{\text{min}}$, the last reached state can be regarded as an additional pseudo-initial state. Subsequent applications of the CIP-solver can target one of the learnt pseudo-initial states in addition to the original initial state $I$. Since the triangle inequality holds, an upper bound for the optimal depth of new reachable paths can be computed. As the distance between $I$ and all learnt pseudo-initial states is known (it equals the depth of the paths for which they were identified), the upper bound is equal to the sum of the distance between $I$ and the targeted pseudo-initial state $I'$ and the length of the test sequence that starts at $I'$.

IV. EXPERIMENTAL RESULTS

The flow described in the previous section was applied to sequential ITC 99 benchmark circuits. All measurements were performed on an AMD Opteron computer using one 2.6 GHz-core and up to 4 GB RAM. In all experiments, the CIP-solver was set to classify an MC-instance as an abort after a time out of 10 seconds. All run-times listed in this section are given in seconds. As input data for all experiments, we extracted 500 globally longest sensitisable paths for each circuit using PHAETON [18]. Additionally, each path is sensitisable using broadside tests [26]. For circuits with less than 500 paths, we applied the flow to all sensitisable paths in the circuit.

We assume all circuits to be resetable and therefore use the all-zero-state (all flip-flops set to 0) as the initial state in all experiments. Please note that our flow itself can handle any set of initial states.

The results for the application of the path-targeting mode are shown in Table I. The table distinguishes between four sets of paths: sensitisable paths found by PHAETON (columns labelled “Total”), sensitisable paths found to be reachable according to the CIP-solver, sensitisable paths found to be unreachable, and sensitisable paths not classified by the CIP-solver within the timeout of 10 seconds. For each of the possible classifications, the table contains four data columns. Column “Count” shows the number of paths that fall into this classification. The run-time needed is given in the next column. The last two columns contain the maximal and the average iteration depth, i.e. the number of time frames needed to reach the desired state starting at the all-zero-state.

The results show that many MC-instances that are proven to be unsatisfiable, i.e. the target property is unreachable, have very low iteration depths. This is explained by the integration of Craig interpolants that accelerate the process of finding a fixed-point. Consequently, in contrast to previous purely fixed-point-based approaches, our flow is applicable for the proof of unreachability as well as for the proof of reachability.

Furthermore, the results are optimal in terms of the generated sequence lengths. Hence, the tool flow finds not only the correct classifications without the use of heuristics, but even computes the shortest possible assignments to the primary inputs.

Figure 4 provides a visualisation of the percentage of reachable, unreachable and aborted instances in the form of stacked-bar charts. The thick black line (secondary y-axis) shows the average iteration depth that was reached.

For many circuits, a high proportion or even all of the sensitisable paths are also sensitisable starting at reachable
Therefore, we did not employ the addition of multiple start states, while for some circuits none or a very small number of paths is sensitisable starting at reachable states.

In nearly all cases the needed iteration depth is very low. Consequently, in contrast to previous purely fixed-point-based approaches, our flow is applicable for the proof of unreachability as well as for the proof of reachability.

As can be seen, targeting the second pattern leads to higher numbers of reachable paths as compared to targeting the first test pattern. This supports the claim that the second test pattern of broadside testing is more functional than the first.

The second observation is that targeting the path’s necessary assignments requires higher run-times than targeting the first or targeting the second test pattern. Targeting a test pattern restricts the search space, since it does not allow the solver to choose other assignments to the primary inputs. Therefore, this mode provides a check whether a given test pattern is reachable or not. However, targeting the path’s side inputs is equivalent to performing a sequential test pattern generation. The CIP-solver performs a search for paths that are known to be sensitisable without restriction to the given test pattern. In many cases, this mode corroborates the reachability of paths that the pattern-targeting mode would otherwise classify as unreachable, e.g. because the given test pair uses illegal states. Hence, the reachability numbers of the path-targeting mode are higher. Extreme cases are e.g. circuits b21 and b22, where the pattern-targeting modes classify all 500 paths as unreachable although test sequences exist. These test sequences are found by the path-targeting mode.

The higher percentage of reachable instances in path targeting mode demonstrates that explicitly targeting functional justification requirements during test generation is more effective than generating the test pair first and then performing the functional justification.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented a versatile technique to verify the reachability of sensitisable paths. The technique is complete and can yield the shortest existing test sequence. Furthermore, the method can provide a formal proof that no justification is possible regardless of the number of allowed time frames. Consequently, in contrast to previous purely fixed-point-based approaches, our flow is applicable for the proof of unreachability as well as for the proof of reachability.
Our experimental results demonstrate the versatility of the approach by showing its application for the computation of a path’s reachability. Future application may include the computation of initialisation and synchronisation sequences.

One further extension of the method is the extraction of conflict information learned by the CIP-solver to increase the efficiency of the method.

REFERENCES
