Complete and Effective Robustness Checking by Means of Interpolation

Stefan Frehse[1] Görschwin Fey[1,3] Eli Arbel[2] Karen Yorav[2] Rolf Drechsler[1,4]

[1]Institute of Computer Science
University of Bremen, Germany
sfrehse@informatik.uni-bremen.de

[2]IBM Research Labs
Haifa, Israel
{arbel,yorav}@il.ibm.com

[3]Institute of Space Systems
German Aerospace Center
Bremen, Germany
goerschwin.fey@dlr.de

[4]Cyber-Phsyical Systems
DFKI-GmbH
Bremen, Germany
rolf.drechsler@dfki.de

Abstract

Technology scaling continues to downscale feature sizes. As a side-effect this has some serious drawbacks, in particular increasing vulnerability of circuits against transient faults caused, e.g., by radiation. Even under malfunctions of internal components the circuit must behave as specified. Several techniques have been proposed to overcome this problem. However, the implementation of those techniques in the design might be buggy and needs to be verified.

This paper provides an effective algorithm using formal reasoning to completely analyze the fault tolerance of a circuit, under all input sequences and all transient faults. The algorithm based on interpolation identifies components in which transient faults are observable. Experiments show that the newly introduced complete approach analyzes ITC'99 and IBM circuits, effectively.

I. Introduction

Technology scaling decreases power consumption and increases the integration density as well as computational throughput. As a side-effect technology scaling comes inherently with serious problems. In particular vulnerability against transient faults increases, caused by radiation or process variation. Transient faults are modeled as the negation of a signal at logic level, i.e, switching from 0 to 1 or 1 to 0 for a short period of time. However, the circuit must behave as specified even under those internal faults. Various techniques to catch and handle those faults are available. These techniques operate on different levels of the design flow. At design level, techniques such as *Triple-Modular-Redundancy* (TMR) or Hamming-Code are applied for detection and correction of internal faults. Furthermore, application specific techniques are available [1]. However, the implementation of these techniques in the design may be faulty itself and needs to be verified to ensure correctness or to show vulnerable parts of the circuit [2].

To ensure correctness of the implementation *complete robustness checking* must be performed: 1) under all possible input sequences and 2) under all transient faults it has to be proven that 3) all outputs sequences adhere to the specification.

Formal approaches have been proposed, which analyze the impact of transient faults on the functional output of the circuit. Previous complete approaches [3], [4], [5], [6] analyzing sequential circuits relying on *Binary Decision Diagrams* are restricted to small circuits due to the state explosion problem. The approach of [7] computes the vulnerability against a user-defined specification in terms of a property based on model checking techniques. Approaches based on *Boolean Satisfiability* (SAT) are either restricted to self-checking circuits (e.g. [8]) or consider only short time intervals [9].

Bounded Model Checking (BMC) based on SAT has been significantly improved by interpolation [10], which is further enhanced in, e.g., [11], [12], [13], [14], [15]. The approach proves or disproves a safety property for finite systems by a fixed-point computation and is successfully applied in complex hardware verification in industry.

Here, we propose two approaches for robustness checking: First, SAT-based robustness checking is improved by utilizing interpolation similarly to interpolation in BMC. Second, we go one step beyond by over-approximating the entire set of reachable states also based on interpolation. This over-approximation can be utilized in BMC for a series of safety properties in general. In robustness checking this is required to identify components that may cause *Silent Data Corruption* (SDC) upon failure - called *unbounded dangerous components* in the following. Knowing such

This work was supported in part by the European Union (Project DIAMOND, FP7-2009-IST-4-248613).

components is mandatory as any second fault may corrupt the system behavior. Previous SAT-based approaches for robustness checking cannot identify unbounded dangerous components. Experimental results on hard benchmarks show the effectiveness in comparison to a previous approach.

The paper is structured as follows: Section II covers the preliminaries in particular the fault model and the computational model. In Section III, the approach adapting interpolation-based model checking for robustness checking is presented. Section IV describes the complete approach including the over-approximation of reachable states and the fixed-point iteration based on interpolation. Section V provides the evaluation of both approaches and Section VI concludes.

II. Preliminaries

A. Circuits and Transition Systems

A synchronous circuit $C = (V, E, L)$ is a directed graph with vertices (components) V, edges $E \subseteq V \times V$, and a labeling function $L : V \to \{\text{IN}, \text{FF}, \text{AND}, \text{NOT}\}$, which maps each vertex to a function. The vertices labeled with FF are state elements. A *state* of the circuit is represented by the values of the FF nodes: Each FF $v_{\text{FF}} \in V$ is mapped to a Boolean value, i.e., $v_{\text{FF}} \mapsto \mathbb{B}$. From a circuit C a transition system $M = (I, S, T)$ is derived. The set $I \subseteq S$ describes the *initial* states. The state space of a circuit with m FFs is given by $S = \mathbb{B}^m$. The transition relation $T(s, s')$ is true, if there is a transition from present state s to a next state s'. The set $\mathsf{img}(Q) = \{s' \in S \mid \exists s \in Q \wedge T(s, s')\}$ contains all successor states reachable in one step from the states in $Q \subseteq S$. The operator img is called an exact image operator. Let $\mathsf{img}(Q)^0 = Q$ and $\mathsf{img}^{i+1}(Q) = \mathsf{img}(\mathsf{img}^i(Q))$, all states reachable from I are given by: $S^* = \bigcup_{i \geq 0} \mathsf{img}^i(I)$. The over-approximation operator $\hat{\mathsf{img}}$ has the following properties: $\mathsf{img}(Q) \subseteq \hat{\mathsf{img}}(Q)$ for $Q \subseteq S$ and hence it holds $S^* \subseteq \hat{S}$ with $\hat{S} = \bigcup_{i \geq 0} \hat{\mathsf{img}}^i(I)$.

In this paper a set of states S is often described by a Boolean predicate δ. We say *the set* δ is an abbreviation for $S = \{s | \delta(s) = 1\}$. Both symbols are used interchangeably.

A formula is called *concrete* if the formula does not contain any approximation operators. Otherwise, the formula is called *abstract*.

B. Boolean Satisfiability & Interpolation

Given a Boolean function, the Boolean satisfiability problem (SAT-problem) is to decide whether there exists an assignment such that the function evaluates to true. If there exists such an assignment, the formula is called *satisfiable* otherwise *unsatisfiable*. Often a *Conjunctive Normal Form* (CNF) is given as input to a SAT solver. A CNF is a conjunction of clauses, where a clause is a disjunction of literals. A literal is a variable x or its negation $\neg x$. A CNF formula F is a set of clauses $F = \{c_1, \ldots, c_k\}$, whereas a clause is a set of literals $c_i = \{l_1, \ldots, l_m\}$. The set of variables of a formula F is denoted as $\mathsf{Var}(F)$.

Given a pair of propositional formulas (A, B) such that $A \wedge B$ is unsatisfiable, there exists an interpolant σ of (A, B) with the following properties based on *Craig's Interpolation Theorem* [16]: 1) σ is a propositional formula over the subset of common variables of A and B, i.e., $\mathsf{Var}(\sigma) \subseteq \mathsf{Var}(A) \cap \mathsf{Var}(B)$, 2) A implies σ, and 3) $B \wedge \sigma$ is unsatisfiable. Intuitively, this means, σ abstracts some facts of A while σ contradicts B. Given a resolution proof of an unsatisfiable CNF an interpolant is computed by, e.g., [10], [17], [18], [19] in linear time with respect to the size of the proof. Note, the size of the proof may be exponentially larger than $|A \cup B|$. Let $\mathsf{itp}(A, B)$ be a function that computes an interpolant of the unsatisfiable pair (A, B).

C. Robustness Checking

The goal of robustness checking is to identify parts of a circuit that may cause unwanted behavior under occurrences of transient faults or to prove that any fault of a component is detected. We consider gates in this work to simplify the presentation. By considering more complex modules as components, multiple faults can be modeled as well [9]. Our approach can easily be lifted to component level, too.

A transient fault is modeled as a non-deterministic bit flip of an output of a node for one time frame. Furthermore, suppose the circuit is equipped with a fault signal f, which reports detected faults of the system by setting f to one. If the system does not have a fault signal, f is equal to zero is assumed. Components of the circuit are classified based on the behavior of the system when a fault is injected, i.e., a component $v \in C$ belongs to one of three classes: **non-robust**: at least one fault at node v is observable after any number of time frames at the primary outputs and no fault is reported, i.e., $f = 0$, **dangerous**: any fault at v only modifies the state after any number of time frames but is not observable on the primary outputs and $f = 0$, or **robust**: otherwise, i.e., all faults are either masked and not observable at the primary outputs or they are reported by the fault signal. Let $\mathbb{T}$ be the set of robust components, $\mathbb{S}$ the set of non-robust components and $\mathbb{D}$ the set of dangerous components: $V = \mathbb{T} \cup \mathbb{S} \cup \mathbb{D}$.

Given a circuit under verification $C = (V, E, L)$, the circuit $C_D(V', E', L')$ with $D \subseteq V$ is constructed by inserting a multiplexer at the component's output for each component $v \in D$ and the primary inputs of C and C' are stimulated by the same values. Each new multiplexer has a selector variable denoted by a_v for a component $v \in D$. The *data 1*-input of the multiplexer

is a fresh primary input and if the selector variable a_v is activated, an arbitrary value is chosen for this input. Otherwise, the normal computation of the component is performed, based on the *data 0*-input connected to the output of the respective component. This construction allows for a non-deterministic bit flip if one selector variable a_v is activated. The derived transition system of C_D is given by $M_D = (I, S', T')$ where the transition relation T' is extended as follows: $T_D = T' \wedge \left(\sum_{v \in D} a_v = 1\right)$, where $\sum$ adds Boolean variables, i.e., exactly one fault injection is performed. Formulas for determining non-robust components are constructed as follows:

$$\mathsf{init}(l) = I(x_0) \wedge \bigwedge_{i=1}^{l} T(x_{i-1}, x_i) \tag{1}$$

$$\mathsf{inj}(l, D) = T(x_l, x_{l+1}) \wedge T_D(x_l, \dot{x}_{l+1}) \tag{2}$$

$$\begin{aligned}\mathsf{prop}_{\mathrm{NR}}(l, k) = & \bigwedge_{i=l+1}^{l+k} (T(x_i, x_{i+1}) \wedge T(\dot{x}_i, \dot{x}_{i+1})) \\ & \wedge P(x_{l+k+1}, \dot{x}_{l+k+1})\end{aligned} \tag{3}$$

where $P(x_{l+k+1}, \dot{x}_{l+k+1})$ forces the primary outputs to be different in the last time frame. The fault signal is forced to be zero to consider scenarios where malfunctions are not detected. However, this is not explicitly written in the formula, to keep the formulas simple. Formula (1) is satisfiable if and only if there is at least one path of length l from the initial state x_0 to a state x_l. Formula (2) computes a normal transition from x_l to x_{l+1} based on the transition relation T and a transition from x_l to $\dot{x}_{l+1}$ based on T_D, i.e., a single fault is **injected** at a component $v \in D$. Finally, in Formula (3), the correct state x_{l+1} and the faulty state $\dot{x}_{l+1}$ are **propagated** over k time frames and the primary outputs are checked for equivalence at the last time frame expressed by P. Conjoining these three formulas the entire formula for determining non-robust components is obtained:

$$\phi(l, k, D) = \mathsf{init}(l) \wedge \mathsf{inj}(l, D) \wedge \mathsf{prop}_{\mathrm{NR}}(l, k) \tag{4}$$

Lemma 1. *Given a non-empty set of components to classify $D \subseteq V$. If the formula $\phi(l, k, D)$ is satisfiable for an l and k, with $a_v = 1$ for a component $v \in D$, then the component v is non-robust.*

Similar to Formula 4 dangerous components are computed using the following formula that compares the states in the last time frame. Here, only the last part is different from Forumla 4 which is given by:

$$\begin{aligned}\mathsf{prop}_{\mathrm{D}}(l, k) = & \bigwedge_{i=l+1}^{l+k} (T(x_i, x_{i+1}) \wedge T(\dot{x}_i, \dot{x}_{i+1})) \\ & \wedge (x_{l+k+1} \neq \dot{x}_{l+k+1})\end{aligned} \tag{5}$$

and the entire formula for classifying dangerous components is given by:

$$\psi(l, k, D) = \mathsf{init}(l) \wedge \mathsf{inj}(l, D) \wedge \mathsf{prop}_{\mathrm{D}}(l, k)$$

The function $\mathsf{sol}(\phi(l, k, D))$ computes all non-robust components of D and the function $\mathsf{sol}(\psi(l, k, D))$ computes all dangerous components of D with respect to the values of l and k. For each combination of l and k non-robust components are determined and afterwards the dangerous components are determined.

In order to compute the complete set of non-robust components of the circuit under verification each combination of $l \in \{0, \ldots, l'\}$ and $k \in \{0, \ldots, k'\}$ has to be checked, where l' and k' are sufficiently large completeness thresholds, i.e.,

$$\mathbb{S} = \bigcup_{\substack{l \in \{0,\ldots,l'\} \\ k \in \{0,\ldots,k'\}}} \mathsf{sol}(\phi(l, k, V)).$$

Once all combinations of l and k have been checked, all non-robust components are determined. While checking all combinations of l and k dangerous components are computed by comparing state bits instead of primary outputs. SDC may occur, i.e., components remain classified as dangerous even when the thresholds for l and k are reached. That means, under all input sequences, any possible faults is not observable at any time on the primary outputs, but at least one fault corrupts the state. This behavior is classified as **unbounded dangerous** and components showing this behavior are in the set $\mathbb{U} \subseteq V$. Given all non-robust components and unbounded dangerous components, the robust components are given by $\mathbb{T} = V \setminus (\mathbb{S} \cup \mathbb{U})$. However, the completeness thresholds might be very large, which makes it hard or practically impossible to check all combinations. The threshold for l is given by the diameter of the transition system $M(I, S, T)$, i.e., $l' = \mathrm{dia}(M)$, because all reachable states can be reached within this number time frames. A completeness threshold for k is given by $k' = 2^{2m}$ with m FFs nodes, because all states of the product machine of the normal transition system and the transition system with fault injection have to be discovered.

III. BMC with Interpolation for Robustness Checking

Checking every combination of l and k is infeasible in practice. However, a complete answer can often be given before reaching l' by applying interpolation-based model checking [10] for robustness checking. Interpolation is exploited in order to a find termination criterion before unrolling the transition relation l' times by computing a fixed-point.

A safety-property holds on a circuit if it is proven that the property holds in every reachable state of the circuit's automaton. BMC has been introduced to check the property on states by iteratively unrolling the transition relation. All reachable states are checked when the transition relation is unfolded l' times. This may become very expensive for larger circuits. Interpolation-based model checking [10] often terminates before reaching l'. Interpolants abstract some facts which are irrelevant to prove the property and therefore speeds up the convergence to a fixed-point. In order to prove a property only relevant states are considered.

In robustness checking a series of safety properties for all $k \in \{0, \cdots, k'\}$ has to be checked. For each new property a model checking instance is solved determining the set of non-robust components under relevant reachable states. Mapping robustness checking to interpolation-based model checking, avoids considering all values of $l \in \{0, \ldots, l'\}$. That means, for each k an earlier termination may be reached.

In the following, interpolation-based model checking [10] is adapted for robustness checking. Given $\phi(l, k, D)$ as defined in Equation 4, the property to check is composed of injection, propagation, and forcing the primary outputs to be different: $\mathsf{inj}(l, D) \wedge \mathsf{prop}_{\mathrm{NR}}(l, k)$. Starting with $l = 0$, consider a non-empty set of components to classify $D \subseteq V$ and the formula pair (A, B) with:

$$A := I(x_0) \wedge T(x_0, x_1) \tag{6}$$

$$B := \bigwedge_{i=2}^{l} T(x_{i-1}, x_i) \wedge \overbrace{\mathsf{inj}(l, D) \wedge \mathsf{prop}_{\mathrm{NR}}(l, k)}^{\text{property of length } k+1} \tag{7}$$

Suppose all non-robust components have been determined with respect to the values of l and k and only the remaining components not shown to be non-robust (at least one, i.e.,$|D| \geq 1$) are extended by a multiplexer to inject a fault. In this case, $A \wedge B$ is unsatisfiable. An interpolant σ with $\sigma = \mathsf{itp}(A, B)$ is computed based on the resolution proof of the SAT solver. The interpolant σ is defined over the state variables expressed by x_1, i.e, the common variables of A and B with $\mathsf{Var}(\sigma) \subseteq \mathsf{Var}(A) \cap \mathsf{Var}(B)$.

The part B may be unsatisfiable itself in circuits containing checker circuitry: The fault signals is constrained to zero but any injection of a fault forces the fault signal to one. In this case, determining the interpolant is skipped because all reachable states are covered.

Otherwise, the interpolant is added to A such that $A = (I(x_0) \vee \sigma(x_0)) \wedge T(x_0, x_1)$ where the variable x_1 of σ is replaced by x_0 and the procedure restarts. A fixed-point is reached when the disjunction of all previously computed interpolants implies the new interpolant. Once the instance becomes satisfiable on such an abstract formula, a potentially spuriously classified non-robust component has been determined due to the over-approximation by the interpolant. Then, the procedure restarts by increasing the value of l, which either allows for classifying non-robust components on the concrete formula or makes the interpolants weaker by strengthening B. A fixed-point will be eventually found and the classification is complete with respect to the current value of k [10]. After reaching a fixed-point, k is increased by one and the interpolation-based model checking procedure restarts, discarding all interpolants.

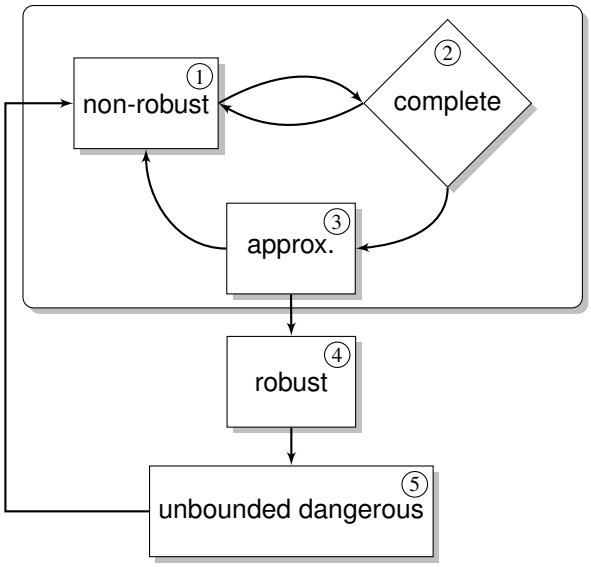
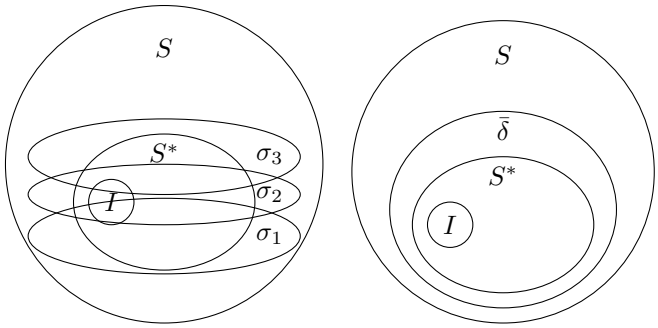


Fig. 1: Flow of new approach

IV. Complete Classification

The approach from the previous Section III may find earlier termination criterion for each k. However, calling this approach for all values of k still remains infeasible. A new approach presented in this section may terminate earlier than reaching l' and k'.

A rough overview of the new approach is depicted in Figure 1: Step ① classifies non-robust components. The subsequent Step ② checks whether all relevant reachable states are considered for a fixed k. These steps have been described in the previous Section III and are embedded in this flow. Any classification of non-robust components is based on reachable states. If there are more states to be covered, the transition relation is further unrolled and the flow goes back to Step ①. Otherwise, if all relevant states are considered an over-approximation of the entire set of reachable states based on a newly introduced construction of interpolants is performed in Step ③ and is described in Section IV-A. After having a new over-approximation, non-robust and dangerous components are over-approximated. This classification is exploited in order to compute robust components. All dangerous components are considered for a further analysis in Step ④, because these components are potentially unbounded dangerous. For example, this



(a) Interpolation by McMillan: (σ) (b) Adequate Approximation: (δ)

Fig. 2: Approximations by Interpolation

occurs in TMR circuits: a fault is corrected by the majority voter but the affected module remains in an erroneous state. In order to prove that these components are unbounded dangerous the proof procedure is called in Step ⑤ and is presented in Section IV-C.

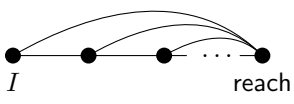
A. Over-approximation of Reachable States

The termination before reaching k' requires that at least all reachable states are modeled for fault injection. Since computing the exact set of reachable states is hard, an over-approximation is introduced in the following based on interpolation. The difference of the interpolants from Section III ([10]) and those here is illustrated in Figure 2. In both figures S is the complete state space and S^* is the set of states reachable from the initial state as introduced in Section II. The left figure shows interpolants σ_1, σ_2, and σ_3 after reaching a fixed-point in three steps. The union of all interpolants yields an over-approximation of the exact set of reachable states. In order to compute a fixed-point using interpolation-based model checking at least one step of interpolation is required, often a few steps, in order to reach a fixed-point. The right figure shows an interpolant $\bar{\delta}$ from the new approach. The computation of this interpolant requires exactly one step of interpolation with additional model checking steps.

Both 1) the union of all interpolants from McMillan's approach and 2) the interpolant introduced in this work over-approximate the exact set of reachable states but the quality, i.e., the number of included non-reachable states may differ. Due to space limitation a detailed comparison about differences cannot not be discussed.

The computation of the over-approximation is introduced in the following. A new partition of (A, B) and a check whether the computed interpolant is an adequate approximation are introduced.

To compute the over-approximations, interpolants are derived from the formula $\mathsf{reach}(l)$ which models all states reachable in l steps from the initial state illustrated in the figure on the right hand side.



$$\mathsf{reach}(l) = I(x_0) \wedge \bigwedge_{i=1}^{l} (I(x_i) \vee T(x_{i-1}, x_i)) \quad (8)$$

All reachable states are modeled on x_l, if $l \geq l'$. The entire formula to determine non-robust components becomes:

$$\phi_{\mathsf{reach}}(l, k, D) = \mathsf{reach}(l) \wedge \mathsf{inj}(l, D) \wedge \mathsf{prop}_{\mathrm{NR}}(l, k) \quad (9)$$

The difference between $\phi(l, k, D)$ and $\phi_{\mathsf{reach}}(l, k, D)$ is that the state x_l for injection might be any state along any path of length l from the initial state rather than only states reachable in l steps. Based on this formulation an over-approximation of the exact set of reachable states is derived.

Definition 1. *Given a transition system $M = (I, S, T)$ and a predicate δ defined over the state variables of M. Then δ is an* **adequate approximation** *if the set δ contains only non-reachable states.*

Lemma 2. *Given an adequate approximation δ, then for all $s \in S^*$, $\bar{\delta}(s)$ is true. That means $\bar{\delta}$ is an over-approximation of the reachable states.*

In order to compute adequate approximations consider the following pair (A', B') of formulas for given l and k

$$\phi_{\mathsf{reach}}(l, k, D) = \overbrace{\mathsf{reach}(l)}^{B'} \wedge \overbrace{\mathsf{inj}(l, D) \wedge \mathsf{prop}_{\mathrm{NR}}(l, k)}^{A'}. \quad (10)$$

The interpolant $\delta = \mathsf{itp}(A', B')$ computes states that are not reachable from the initial state in l or less steps, but A' implies δ, i.e., states in δ fulfill the property. These states are possibly reachable from the initial state in more than l steps or are non-reachable states.

Theorem 1. *Given a transition system $M = (I, S, T)$. There exists an $l \leq l'$ and an $k \leq k'$ with a non-empty set $D \subseteq V$ of components proven to be not non-robust with respect to l and k. Then, $\phi_{\mathsf{reach}}(l, k, D)$ is unsatisfiable and $\delta = \mathsf{itp}(A', B')$ is an adequate approximation.*

Proof: Setting $l = l'$ and $k = k'$, $\mathsf{reach}(l)$ models all reachable states and $\phi_{\mathsf{reach}}(l, k, D)$ is unsatisfiable with $|D| \geq 1$. An interpolant $\delta = \mathsf{itp}(A', B')$ is computed. All states for which A' is true satisfy δ. These states are only non-reachable states since $\delta \wedge B'$ is unsatisfiable and B' models all reachable states for $l = l'$. ■

This proves that an adequate over-approximation is computed when setting l to l'. However, in practice an adequate approximation is often computed before l reaches l'. In order to verify that the computed interpolant δ is an adequate approximation, a separate model checking step is performed, i.e., the interpolant is checked for reachable

states. We employ an interpolation-based model checker to check the invariant $\bar{\delta}$.

While checking each combination of l and k to determine non-robust components using $\phi_{\mathsf{reach}}(l,k,D)$, interpolants are computed based on Theorem 1. If an interpolant is an adequate approximation, the interpolant is added to the set Δ, where Δ contains all adequate approximations.

B. Approximation of Robust Components

Given a set of adequate approximations $\Delta = \{\delta_1, \ldots, \delta_n\}$ non-robust and dangerous components are over-approximated using the following formulas. All remaining components are robust components.

At first, the formula for over-approximating non-robust components is constructed:

$$\hat{\phi}(l,k,D,\Delta) = \bigwedge_{\delta \in \Delta} \bar{\delta}(x_l) \wedge \mathsf{inj}(l,D) \wedge \mathsf{prop}_{\mathrm{NR}}(l,k). \tag{11}$$

Lemma 3. *Given a non-empty set $D \subseteq V$ and a set of adequate approximations Δ. Let $\mathbb{S}_l^k = \mathsf{sol}(\phi(l,k,D))$ and $\hat{\mathbb{S}}_l^k = \mathsf{sol}(\hat{\phi}(l,k,D,\Delta))$, then $\mathbb{S}_l^k \subseteq \hat{\mathbb{S}}_l^k$ is true for any l and k. That means, $\hat{\mathbb{S}}_l^k$ is an over-approximation of non-robust components.*

Since the formula $\hat{\phi}$ may consider more states that ϕ because an over-approximation is constrained and thus non-robust components are over-approximated.

Next, an over-approximation of the set of dangerous components is introduced. As described in Section II, a fault injected into a dangerous component corrupts the state but is not observable at the primary outputs. The formula to over-approximate dangerous components is given by:

$$\hat{\psi}(l,k,D,\Delta) = \bigwedge_{\delta \in \Delta} \bar{\delta}(x_l) \wedge \mathsf{inj}(l,D) \wedge \mathsf{prop}_{\mathrm{D}}(l,k) \tag{12}$$

Lemma 4. *Given a non-empty set $D \subseteq V$ and a set of adequate approximations Δ. Let $\mathbb{D}_l^k = \mathsf{sol}(\psi(l,k,D))$ and $\hat{\mathbb{D}}_l^k = \mathsf{sol}(\hat{\psi}(l,k,D,\Delta))$, then $\mathbb{D}_l^k \subseteq \hat{\mathbb{D}}_l^k$ is true for any l and k. That means, $\hat{\mathbb{D}}_l^k$ is an over-approximation of dangerous components.*

The over-approximated sets of non-robust and of dangerous components determine a subset of robust components as stated in the following lemma.

Lemma 5. *Given an over-approximated set of non-robust $\hat{\mathbb{S}}_l^k$ and dangerous $\hat{\mathbb{D}}_l^k$ components, respectively. A set of robust components is given by: $\check{\mathbb{T}}_l^k = V \setminus (\hat{\mathbb{S}}_l^k \cup \hat{\mathbb{D}}_l^k)$.*

By checking all combinations of l and k the final result of non-robust and robust components is determined, i.e., $\mathbb{S} = \bigcup_{l,k} \mathbb{S}_l^k$ and $\mathbb{T} = \bigcup_{l,k} \check{\mathbb{T}}_l^k$. Reaching l' and k', the classifications are complete.

However, components are unbounded dangerous on certain circuits even when the thresholds for l and k are met. All these unbounded dangerous components are reconsidered in a next iteration. In order to prove that these components are unbounded dangerous, it is required to prove that any fault of these components will not affect the primary outputs for any combination of l and k. The following section provides the corresponding proof procedure.

C. Fixed-point Computation on the Property

Suppose arbitrary values for l and $k > 0$, and a non-empty set of components $D \subseteq V$ which are to be proven or to be refuted to be unbounded dangerous components. Formula (11), $\hat{\phi}(l,k,D)$ is unsatisfiable, therefore an interpolant of $\sigma = \mathsf{itp}(X,Y)$ can be computed where:

$$\begin{aligned} X := & \bigwedge_{\delta \in \Delta} \bar{\delta}(x_l) \wedge \mathsf{inj}(l,D) \\ & \wedge T(x_{l+1}, x_{l+2}) \wedge T(\dot{x}_{l+1}, \dot{x}_{l+2}) \end{aligned} \tag{13}$$

$$\begin{aligned} Y := & \bigwedge_{i=l+2}^{l+k+1} T(x_i, x_{i+1}) \wedge T(\dot{x}_{i+1}, \dot{x}_{i+1}) \\ & \wedge P(x_{l+k+1}, \dot{x}_{l+k+1}) \end{aligned} \tag{14}$$

The interpolant σ contains state variables x_{l+2} and $\dot{x}_{l+2}$ of transition relation T, i.e., $\mathsf{Var}(\sigma) = \mathsf{Var}(A'') \cap \mathsf{Var}(B'')$. Intuitively, σ computes an approximated set of pairs of successor states of the correct states and faulty states reached after injecting a fault. The variables x_{l+2} and $\dot{x}_{l+2}$ of the interpolant σ are replaced by x_{l+1} and $\dot{x}_{l+1}$, respectively. The interpolant is added to X, i.e.,

$$\begin{aligned} X' = & \left(\left(\bigwedge_{\delta \in \Delta} \bar{\delta}(x_l) \wedge \mathsf{inj}(l,D) \right) \vee \sigma(x_{l+1}, \dot{x}_{l+1}) \right) \\ & \wedge T(x_{l+1}, x_{l+2}) \wedge T(\dot{x}_{l+1}, \dot{x}_{l+2}) \end{aligned} \tag{15}$$

Next, the extended formula $X' \wedge Y$ is checked for satisfiability. If the formula is satisfiable, the classification is potentially spurious. In that case, k is increased by one and the computation proceeds clearing problem instances and interpolants but keeping Δ. Otherwise, i.e., the formula is unsatisfiable, a new interpolant is computed and it is checked whether the disjunction of all previously computed interpolants implies the new interpolant – a fixed-point has been reached. This proves that no fault injection in components of D is observable at the outputs. Thus, all components of D are proven unbounded dangerous and constitute the set U.

D. Algorithm

The overall procedure which exploits Theorem 1 and the fixed-point iteration on the property from Section IV-C is

Algorithm 1: ROB-COMPL

Input: $C = (V, E, L)$ a sequential circuit
Output: $(\mathbb{S}, \mathbb{T}, \mathbb{U})$ with $\mathbb{S} \subseteq V$ the non-robust and $\mathbb{T} \subseteq V$ the robust as well as $\mathbb{U}$ unbounded dangerous comps.

```
1  begin
2      k = l = 0; Δ = ∅;
3      S = T = U = ∅;
4      D = V;
5      while true do
6          S = S ∪ sol(φ(l, k, D));
7          D = D \ S;
8          if (S ∪ T = V) or (k = k') then return (S, T, U);
9          δ = itp(inj(l, D) ∧ prop_NR(l, k), reach(l));
10         if δ is an adequate approximation then
11             Δ = Δ ∪ δ;
12         end
13         if necessary states are checked with respect to k then
14             l = 0;
15         else
16             l = l + 1;
17             continue;
18         end
19         Ŝ_l^k = sol(φ̂(l, k, D, Δ));
20         D̂_l^k = sol(ψ̂(l, k, D \ Ŝ_l^k, Δ));
21         T̃_l^k = V \ (Ŝ_l^k ∪ D̂_l^k);
22         D = D \ T̃_l^k;
23         T = T ∪ T̃_l^k;
24         if T ∪ S = V then
25             return (S, T, U)
26         else
27             if k = 0 then k = 1; continue;
28             Q = ⋀_{δ∈Δ} δ̄(x_l) ∧ inj(l, D̂_l^k);
29             X = T(x_{l+1}, x_{l+2}) ∧ T(ẋ_{l+1}, ẋ_{l+2});
30             Y = ⋀_{i=l+2}^{l+k+1} T(x_i, x_{i+1}) ∧ T(ẋ_{i+1}, ẋ_{i+1}) ∧
               P(x_{l+k+1}, ẋ_{lk+1});
31             repeat
32                 σ = itp(Q ∧ X, Y);
33                 if Q → σ then
34                     U = U ∪ D̂_l^k;
35                     D = D \ D̂_l^k;
36                     continue;
37                 else
38                     Q = Q ∨ σ(x_{l+1}, ẋ_{l+1});
39                 end
40             until Q ∧ X ∧ Y is satisfiable ;
41             k = k + 1;
42         end
43     end
44 end
```

shown as pseudo-code in Algorithm 1. The algorithm gets the circuit under verification as input and determines the set of non-robust as well as robust components. Lines 2-4 initialize the required sets and the values for l and k. All components are to be classified, i.e., $D = V$. The `while`-loop iterates until all components are classified or the threshold for k is reached (line 8). In each iteration at first non-robust components are determined (line 6) and are excluded from the components to be classified in further iterations (line 7). Next, in line 9 an interpolant is computed based on Formula (10) and it is checked whether the interpolant is an adequate approximation using interpolation-based model checking (line 10). If the computed interpolant is an adequate approximation, then the interpolant is added to the set Δ (line 11). If all necessary states are covered as checked by interpolation-based fixed-point computation, then l is set to zero and the algorithm proceeds with line 19. Otherwise, the value of l is increased by one (line 16) if not all necessary states are checked for the current value of k and the outer loop restarts. That means either further classifications are performed or an adequate approximation will be found by strengthening A' by increasing l by one. Eventually an adequate approximation will be found according to Theorem 1 and the procedure continues with line 19. Here, the over-approximations of the set of non-robust and dangerous components are computed, such that a subset of the robust components is determined. In line 27–41 the fixed-point computation on the property is performed. If a fixed-point is found, all components previously classified as dangerous in line 22 are proven to be unbounded dangerous components. If the formula becomes satisfiable, k is increased by one to get a weaker approximation by interpolation on the property. In a next iteration further non-robust components may be classified from the set of remaining dangerous components.

E. Adequate Approximation in Model Checking

Since, the computed adequate approximations are invariants specifying reachable states of the circuits they can be used in model checking in general. While model checking to prove a property, adequate approximations can be derived in the same way as described above. These adequate approximations may be applied as invariants to prune the search space while proving other properties.

V. Experiments

An evaluation of the proposed approaches is presented in this section. Experiments have been carried out on a Dual-Core AMD Opteron™ Processor with 3.0 Ghz and 64GB main memory under Linux. Reaching a timeout of 15 hours is marked by *timeout*. As SAT-solver MiniSAT's proof logging version [20] has been used and interpolants are computed based on McMillan's interpolation system [10]. Preliminary experiments have shown that the HKP [17], [18], [19] interpolants yielded consistently longer run times of the proposed algorithms than using McMillan's interpolants. Interpolants are represented as *And-Inverter-Graphs* (AIG) by the Aiger software package[1].

A. ITC'99 Benchmarks

For the first evaluations ITC'99 benchmarks were taken and enhanced with TMR techniques to catch single transient faults. TMR circuits are marked with the suffix `-tmr`. Faults are injected into gates and flipflops, i.e., initially $D = V$. The two proposed approaches from Section III and Section IV as well as the approach from the work of [9] are compared in this section.

The circuits are known to be hard for formal robustness checking, because fault effects propagate until reaching the

[1] Available under: http://fmv.jku.at/aiger/

				Approach of [9]			Interp.-Based BMC			Complete Approach							
Circuit	|IN|	|OUT|	|FF|	R_{lb}	R_{ub}	Runtime	k	R_{ub}	Runtime	l	k	invalid	valid	∅ size	R_{lb}	R_{ub}	Runtime
b01-tmr	2	2	15	7.9	98.2	*timeout*	236	97.8	*timeout*	17	4	15	4	3k	97.8	97.8	3
b02-tmr	1	1	12	1.7	98.2	*timeout*	351	98.0	*timeout*	25	9	23	4	1k	98.0	98.0	<1
b03-tmr	4	4	90	1.2	98.7	*timeout*	40	98.7	*timeout*	24	4	22	4	91k	98.7	98.7	31
b04-tmr	11	8	198	0.6	100.0	*timeout*	1	100.0	*timeout*	6	1	4	1	3583k	0.0	99.2	*timeout*
b05-tmr	1	36	102	6.9	99.0	*timeout*	70	98.9	*timeout*	16	4	14	3	29k	98.9	98.9	319
b06-tmr	2	6	27	3.6	97.0	*timeout*	28	96.7	*timeout*	9	3	7	3	6k	96.7	96.7	11
b07-tmr	1	8	147	0.9	99.5	*timeout*	72	99.4	*timeout*	21	4	19	3	22k	99.4	99.4	1621
b08-tmr	9	4	63	1.2	99.4	*timeout*	25	99.4	*timeout*	14	5	12	3	4k	99.4	99.4	68
b09-tmr	1	1	84	0.3	99.6	*timeout*	84	99.6	*timeout*	19	4	17	4	6k	99.6	99.6	44
b10-tmr	11	6	51	1.5	97.8	*timeout*	9	97.8	*timeout*	16	4	14	3	1286k	97.8	97.8	778
b11-tmr	7	6	93	0.6	99.4	*timeout*	11	99.4	*timeout*	13	2	11	3	537k	99.4	99.4	373
b12-tmr	5	6	363	0.3	99.8	*timeout*	12	99.8	*timeout*	19	2	17	3	1012k	99.8	99.8	395
b13-tmr	10	10	159	2.3	99.0	*timeout*	4	99.0	*timeout*	7	3	5	3	9k	99.0	99.0	213

TABLE I: Determined robustness of ITC'99 circuits

majority voter and are then masked late in the design. This structure is hard for most SAT-solvers to handle causing long run times. Furthermore, faults on most components do not affect the primary outputs of the circuit, i.e., the faults are masked by the majority voter, but modify the state. That means, in each iteration most components are reconsidered as dangerous until they are finally proven to be unbounded dangerous components. The approach of [9] and from Section III are practically not able to compute the unbounded dangerous components, due to the large value of k'. The new complete approach that does not explicitly unroll the transition relation for k' time frames.

Results are shown in Table I. Properties of the circuit like name, number of primary inputs, number of primary outputs and number of state elements are shown in the first four columns. Values for the robustness are computed as follows:

$$R_{lb} = \frac{|\mathbb{T} \cup \mathbb{U}|}{|V|}, \qquad R_{ub} = 1 - \frac{|\mathbb{S}|}{|V|}. \tag{16}$$

The runtimes are given in seconds.

Once the computation is finished by the evaluated approaches, the robustness is computed based on the respective sets of non-robust and robust components returned by the approaches. The approach from Section III computes only a set of non-robust components and therefore only the upper bound of the robustness R_{ub} is computed. Furthermore, the values for l and k are shown for the complete approach in Table I, to denote when the approach finished with a complete answer. Furthermore, details about the interpolants are given. The approach computes interpolants to get adequate approximations. Column *invalid* denotes the number of interpolants which are not an adequate approximation. Column *valid* denotes the number of adequate approximations. Column *∅ size* denotes the average size of the interpolants given as number of nodes of the AIGs. The run times to check whether an interpolant is an adequate approximation are quite short, i.e., less than 5 minutes accumulated over all instances.

For the approach from [9] the gap of the bounds is very large for each considered circuit. This happens, because almost all faults are masked by the majority voter of the TMR circuit, but are not corrected such that one of the TMR modules is corrupted unless resetting the circuit. That means, the approach of [9] was not able to prove that the dangerous components will not affect the primary outputs at any time.

The approach from Section III computes a tight upper bound. As shown in the table the approach classifies some more components as non-robust than the approach of [9] for the circuits, `b01-tmr`, `b02-tmr`, `b04.tmr`, `b05-tmr`, `b06-tmr`, `b07-tmr`. Rather than a pre-defined under-approximation of reachable states [9], necessary reachability information is computed.

In principle, the approach of [9] uses fixed values $l = 10$ and $k = 10$ that prevent a complete classification. The BMC-based approach does not terminate even for a large observation window as no fixed-point iteration after fault injection is performed. However, a complete classification is possible even for a small observation window k which also requires precise reachability information. This shows that both interpolation steps are required for an effective classification.

The complete approach proves lower and upper bounds very effectively and provides an exact analysis for almost all circuits. In these cases lower bound and upper bound meet.

B. IBM Benchmarks

Next, we checked the proposed complete method on a set of IBM design blocks. In this setting, two groups of design blocks were used: 1) blocks taken from a data-path design, and 2) blocks taken from a micro-processor control unit. All of these design blocks contain self error checking and correction mechanisms implemented in hardware. In this benchmark, an Intel i5 processor running at 3.1GHz with 4GB RAM was used. Table II summarizes the benchmark results: blocks D1-13 are the data-path blocks, D14-D27

Circuit	|IN|	|OUT|	|FF|	Classified [%]	l	k	Runtime
D1	204	259	1430	9.30%	2	0	2728
D2	228	65	1424	17.49%	5	1	783
D3	727	293	1395	7.74%	3	0	220
D4	700	497	1038	70.52%	7	1	678
D5	364	142	940	100.00%	2	1	60
D6	105	60	699	99.86%	2	57	1699
D7	284	262	513	84.99%	18	3	3797
D8	112	56	456	100.00%	6	2	144
D9	268	99	447	89.26%	8	1	8281
D10	734	194	435	87.36%	2	1	611
D11	155	120	394	100.00%	2	1	11
D12	53	37	322	100.00%	2	1	19
D13	124	67	222	48.20%	5	3	37
D14	119	112	878	81.55%	5	3	1492
D15	140	55	804	88.56%	31	0	710
D16	29	24	555	15.86%	61	1	1201
D17	377	25	506	70.16%	6	6	1504
D18	176	154	464	70.26%	55	1	2044
D19	252	131	451	56.54%	7	7	1714
D20	327	102	428	67.06%	3	44	9050
D21	173	256	412	88.35%	8	4	486
D22	135	206	247	90.28%	2	33	23170
D23	218	96	231	95.24%	2	86	3631
D24	119	57	231	96.54%	2	2	580
D25	227	114	216	95.37%	4	95	7589
D26	70	51	210	17.62%	131	0	3697
D27	103	63	207	91.30%	7	6	598
D28	130	79	195	95.90%	2	80	35888
D29	100	37	126	100.00%	5	5	353
D30	139	94	123	100.00%	4	5	59

TABLE II: IBM Benchmarks

are the control logic blocks, number of flip flops, primary inputs and primary outputs are given for each design in the |IN|, |OUT| and |FF| columns respectively. In addition, the percentage of classified flip-flops are given, as well as the l and k bounds which were reached during each run.

One can see in Table II, the method was able to classify significant percentage of flip-flops in most of the blocks. In case where the implementation ran out of memory less than 100% of the flip flops were classified. This can be attributed to several reasons: 1) The underlying SAT solver simply ran out of memory trying to solve a particular instance, 2) problem instance itself may get too large to fit into memory. In addition it is worth noting that in the current implementation the interpolants are not being optimized, which might also contribute to sub-optimal performance. However, the implementation of the newly introduced approach was able to effectively classify industrial design coming from a micro-processor design.

VI. Conclusion

This work proposed a new complete approach for robustness checking utilizing interpolants to overcome complexity problems. We over-approximate the exact set of reachable states and compute a fixed-point on the property. Our approach is effective and provides exact results for hard benchmarks.

References

[1] S. Krishnaswamy, S. M. Plaza, I. L. Markov, and J. P. Hayes, "Enhancing design robustness with reliability-aware resynthesis and logic simulation," in *ICCAD*, 2007, pp. 149–154.

[2] A. Pellegrini, V. Bertacco, and T. Austin, "Fault-based attack of RSA authentication," in *DATE*, 2010, pp. 855–860.

[3] J. Hayes, I. Polian, and B. Becker, "An analysis framework for transient-error tolerance," in *VTS*, may 2007, pp. 249–255.

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

[5] M. Bozzano, A. Cimatti, and F. Tapparo, "Symbolic fault tree analysis for reactive systems," in *ATVA*, ser. LNCS, vol. 4762, 2007, pp. 162–176.

[6] R. Leveugle, "A new approach for early dependability evaluation based on formal property checking and controlled mutations," in *IOLTS*, 2005, pp. 260–265.

[7] S. A. Seshia, W. Li, and S. Mitra, "Verification-guided soft error resilience," in *DATE*, 2007, pp. 1442–1447.

[8] M. Hunger, S. Hellebrand, A. Czutro, I. Polian, and B. Becker, "ATPG-based grading of strong fault-secureness," in *IOLTS*, 2009, pp. 269 –274.

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

[10] K. L. McMillan, "Interpolation and SAT-Based model checking," in *CAV*, ser. LNCS, 2003, pp. 1–13.

[11] J. Marques-Silva, "Interpolant learning and reuse in SAT-based model checking," *Electron. Notes Theor. Comput. Sci.*, vol. 174, no. 3, pp. 31–43, May 2007.

[12] V. D'Silva, M. Purandare, G. Weissenbacher, and D. Kroening, "Interpolant strength," in *VMCAI*, ser. LNCS, vol. 5944, 2010, pp. 129–145.

[13] G. Cabodi, M. Murciano, S. Nocco, and S. Quer, "Boosting interpolation with dynamic localized abstraction and redundancy removal," *TODAES*, vol. 13, no. 1, pp. 1–20, 2008.

[14] V. D'Silva, M. Purandare, and D. Kroening, "Approximation refinement for interpolation-based model checking." in *VMCAI*, ser. LNCS, 2008, pp. 68–82.

[15] S. F. Rollini, O. Sery, and N. Sharygina, "Leveraging interpolant strength in model checking," in *CAV*, Springer. Berkeley, California, USA: Springer, 2012.

[16] W. Craig, "Linear reasoning. A new form of the Herbrand-Gentzen theorem," *The Journal of Symbolic Logic*, vol. 22, no. 3, pp. 250–268, 1957.

[17] G. Huang, "Constructing Craig interpolation formulas," in *Annual International Conference on Computing and Combinatorics*, 1995, pp. 181–190.

[18] J. Krajicek, "Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic," *The Journal of Symbolic Logic*, vol. 62, no. 2, pp. 457–486, 1997.

[19] P. Pudlák, "Lower bounds for resolution and cutting plane proofs and monotone computations," *The Journal of Symbolic Logic*, vol. 62, no. 3, pp. 981–998, 1997.

[20] N. Eén and N. Sörensson, "An extensible SAT solver," in *SAT 2003*, ser. LNCS, 2003, pp. 502–518.