A Basis for Formal Robustness Checking[*]

Görschwin Fey[1,2]
[1]VLSI Design & Education Center
University of Tokyo, Tokyo 113-0032, Japan

Rolf Drechsler[2]
[2]Institute of Computer Science
University of Bremen, 28359 Bremen, Germany

{fey,drechsle}@informatik.uni-bremen.de

Abstract

Correct input/output behavior of circuits in presence of internal malfunctions becomes more and more important. But reliable and efficient methods to measure this robustness *are not available yet.*

In this paper a formal measure for the robustness of a circuit is introduced. Then, a first algorithm to determine the robustness is presented. This is done by reducing the problem either to sequential equivalence checking or to a sequence of property checking instances. The technique also identifies those parts of the circuit that are not robust from a functional point of view and therefore have to be hardened during layout.

1 Introduction

The number of safety critical applications that rely on integrated circuits is growing, e.g. “stear-by-wire” in cars or important control functions in planes. The functional correctness is certified by massively applying simulation-based as well as formal verification methods.

At the same time the number of components integrated in a single circuit grows rapidly according to Moore’s Law. Meanwhile the area occupied by a single component shrinks continuously. The reliability of components decreases and a circuit becomes sensitive to faults that occur after production during in-field application. Among such faults are transient malfunctions due to environmental radiation that cause *Single Event Upsets* (SEU) or static faults caused e.g. by electro-migration due to aging of the material.

Architectural measures are already applied during circuit design to ensure that malfunctions of components do not impact the functional correctness. Thus, the malfunction is signaled while the input/output behavior is consistent with the original specification. A simple technique to achieve such robustness in presence of unreliable components is redundancy. Fault tolerant codes are more sophisticated.

A symbolic approach to analyze the reliability of circuits has recently been introduced in [7]. Outcome of the analysis is a probability for faults in the output response of the circuit. Hints to identify non-robust internal structures are not provided.

Simulation-based validation techniques are commonly used to ensure that the circuit fulfills the specification even in presence of malfunctions. These malfunctions are injected into the internal structures of the circuit. Then, simulation shows whether the malfunction produces faulty output responses. To improve the coverage of the state space, emulation techniques can be applied [1]. But these techniques are incomplete in the sense that not all states of the system can be covered. States that cause faulty behavior when a malfunction occurs may remain uncovered.

On the contrary, the application of formal methods proves that *any malfunction* in *any state* of the system *under any input sequence* (1) is detected and (2) does not cause erroneous input/output behavior. First approaches were proposed in [5, 4]. Both methods apply tools for formal verification as a “black box”. To prove the robustness of a circuit with respect to a given fault model, each individual fault has to be injected by applying a *mutant* to the circuit description. The resulting faulty circuit is then formally verified against the correct circuit. An explicit enumeration of all possible faults is necessary, which is not feasible to capture multiple faults occurring at the same time. Moreover, the types of covered faults is limited by the mutants that are applied. The method in [5] only returns “yes” or “no” to the question whether the circuit is robust with respect to a particular fault. Additionally, [4] determines the percentage of “robust states” of the system. These answers do not help when trying to identify parts of the circuit where the robustness has to be improved by architectural changes or by hardening the physical circuit structures [14].

Here, a first formal approach is presented to implicitly consider all faults with respect to different fault models. The proposed algorithm determines those locations in the circuit where the fault tolerance has to be improved. For each location that is not robust, a particular fault and a simulation trace exciting the faulty output response can be calculated. The robustness of a circuit with respect to a given fault model is formally defined. When 100% robustness is achieved, no fault of the given fault model has an impact on the input/output behavior of the circuit. The calculation of the robustness measure is reduced to sequential equivalence

[*] This work is partially funded by the Advantest Corporation.

checking. Using formal methods, the process to implicitly consider all faults is explained. A solver for *Boolean Satisfiability* (SAT) is applied as the proof engine. The basic technique has similarities to SAT-based diagnosis as introduced in [10]. An inductive approach to improve the performance for certain circuits is discussed. Experiments show the practical applicability.

This paper is structured as follows: Preliminaries are reviewed in the following section. The notion of robustness is introduced together with the appropriate fault models in Section 3. The algorithm to implicitly consider all faults according to a given fault model is presented in Section 4. A decomposition using an inductive approach is discussed in Section 5. First experimental results are reported in Section 6. Finally, the work is summarized in the last section.

2 Preliminaries

A *circuit* $\mathcal{C}$ consists of a set of components. Among these are primary inputs, primary outputs, state elements and internal combinational components $g \in \mathcal{C}$. A Boolean function is associated with each internal component. A single gate, a module or a *Register Transfer* (RT) level expression may correspond to a component. The structure of the circuit is defined by a graph. In particular, this graph uniquely provides predecessors and successors of a component.

The *size* of the circuit is given by the number of components, i.e. by $|\mathcal{C}|$. A *part* of a circuit is a subset $\mathcal{S} \subseteq \mathcal{C}$ of the components. The size of $\mathcal{S}$ is given by $|\mathcal{S}|$.

The input/output behavior of the circuit emerges from the composition of components and their functionality. Starting from a defined initial state, that is reached by a reset sequence, a particular input sequence leads to a unique output sequence [9].

For the manipulation of Boolean functions there exist different techniques. In this work SAT provers [3] are applied. The transformation of a circuit into a SAT instance requires run time and memory resources linear in the size of the circuit [13]. The decision whether a SAT instance is satisfiable is NP-complete [2]. Nonetheless, modern SAT solvers effectively handle large problem instances [6, 8].

3 Measuring Robustness

Fault models are introduced in this section and motivated by faults of practical relevance. Then, a formal measure of robustness is defined with respect to the fault models.

3.1 Fault Models

Faults occurring during in-field application can be grouped in transient faults, e.g. so called SEUs caused by radiation, and static faults, e.g. due to electro-migration processes. To differentiate the robustness of a circuit with respect to these types, appropriate fault models are introduced in the following.

Definition 1 *A circuit $\mathcal{C}$ and a part $\mathcal{S} \subseteq \mathcal{C}$ of this circuit are given.*

1. *Injecting a fault according to the* non-deterministic fault model $\mathcal{F}_N$*, means to replace the outputs of a component $g \in \mathcal{S}$ by new primary inputs.*

2. *Injecting a fault according to the* combinationally deterministic fault model $\mathcal{F}_C$*, means to replace a component $g \in \mathcal{S}$ by a new combinational subcircuit that has the same successors as g.*

3. *Injecting a fault according to the* locally deterministic fault model $\mathcal{F}_L$*, means to replace a component $g \in \mathcal{S}$ by a new combinational circuit that has the same predecessors and successors as g.*

The sequence of fault models $\mathcal{F}_N$, $\mathcal{F}_C$ and $\mathcal{F}_L$ imposes an increasing number of constraints onto the functional modification of the circuit. For example, each faulty output response that can be achieved by injecting a fault according to $\mathcal{F}_C$, can also be created by injecting a fault according to $\mathcal{F}_N$ – but not vice versa.

The fault models correspond to different realistic fault types. Certain types of static faults can be modeled by $\mathcal{F}_L$, e.g. transistor level faults like shorts may result in changing the function of a gate. On the other hand any fault that leads to faulty values on the functional level can be modeled by $\mathcal{F}_N$ due to the non-determinism introduced by this fault model. In particular transient faults like SEUs are caught by this model.

In the following the set $\mathbb{C}_{\mathcal{C},\mathcal{S},\mathcal{F},\eta}$ denotes the set of all circuits that can be derived from circuit $\mathcal{C}$ by injecting η faults according to fault model $\mathcal{F}$ into a part $\mathcal{S} \subseteq \mathcal{C}$.

3.2 Definition

A circuit is called robust if no fault changes the input/output behavior. Nonetheless, for example a SEU that occurs at a primary output of a circuit may inevitably modify the output response of the circuit. To avoid this, individual parts of a circuit can be hardened during fabrication, e.g. by using larger structures to realize the components. But this kind of robustness cannot be captured on a Boolean model of the circuit without layout or mapping information. Therefore a more sophisticated definition of robustness that can be applied to parts of the circuit is necessary.

Moreover, in some cases robustness with respect to single faults may not be sufficient, because even a local phenomenon may cause a malfunction of multiple components. Therefore the notion of robustness is defined with respect to multiple faults as well.

Both aspects – the consideration of parts of a circuit and multiple faults – are covered by the following definitions. A robust subcircuit is defined as follows.

Definition 2 *A circuit $\mathcal{C}$, a fault model $\mathcal{F}$ and an integer $\eta \geq 1$ are given. A part $\mathcal{S} \subseteq \mathcal{C}$ of $\mathcal{C}$ is called* $(\mathcal{F}, \eta)$-robust *if no injection of η faults into $\mathcal{S}$ according to $\mathcal{F}$ changes the input/output behavior of $\mathcal{C}$.*

This is the basis to define the robustness of a circuit $\mathcal{C}$ for η-fold faults with respect to a fault model $\mathcal{F}$. Using the largest $(\mathcal{F}, \eta)$-robust part S of the circuit is not sufficient in presence of multiple faults in general. Some other part T that is not $(\mathcal{F}, \eta)$-robust may share components with S. Therefore the largest part $\mathcal{S}$ of $\mathcal{C}$ is determined that does not have a component which occurs in an η-fold fault that changes the input/output behavior of $\mathcal{C}$.

Definition 3 *A circuit $\mathcal{C}$, a fault model $\mathcal{F}$ and an integer $\eta \geq 1$ are given. The $(\mathcal{F}, \eta)$*-robustness *of $\mathcal{C}$ is given by $R_{\mathcal{F},\eta} = \frac{|\mathcal{S}|}{|\mathcal{C}|}$, where $\mathcal{S}$ is a maximal subset of $\mathcal{C}$ such that forall $\mathcal{T} \subseteq \mathcal{C}$ if*

$$\mathcal{S} \cap \mathcal{T} \neq \emptyset \text{ and } |\mathcal{T}| \leq \eta$$

then

$$\mathcal{T} \text{ is } (\mathcal{F}, |\mathcal{T}|)\text{-robust.}$$

The robustness of a circuit with respect to a given formal property can be defined analogously. Consequently, the algorithm that is introduced in the next section can by applied to calculate the robustness with respect to a property.

4 Calculating Robustness

Sequential equivalence checking can obviously be used to calculate the robustness of a circuit. The sequential equivalence of the original circuit to all possible faulty circuits has to be verified to proof robustness. This is summarized by the following theorem.

Theorem 1 *A circuit $\mathcal{C}$ and a set of faulty circuits $\mathbb{C}_{\mathcal{C},\mathcal{S},\mathcal{F},\eta}$ are given. A part $\mathcal{S}$ is $(\mathcal{F}, \eta)$-robust if and only if each circuit $\mathcal{C}' \in \mathbb{C}$ is sequentially equivalent to $\mathcal{C}$.*

Despite the direct mapping of state elements between the faulty circuit and the original circuit, a simple reduction to combinational equivalence is not possible in general. A fault may change the state transition function without impact on the input/output behavior. Moreover, the number of derived faulty circuits is very large. An enumeration of all these circuits would be too time consuming. Therefore an algorithm to consider all faulty circuits in a single SAT instance is presented.

The new approach "borrows" ideas that were originally proposed for diagnosis based on SAT [11, 12]. During diagnosis a modification of the circuit is needed that allows to correct faulty behavior. In the context of robustness checking, a modification that causes incorrect behavior is required.

Initially, the approach is explained for fault model $\mathcal{F}_N$ and then extended to handle the other fault models. The creation of the SAT instance is explained in terms of a circuit that is transformed into conjunctive normal form afterward.

```
 1  function largestRobustPart($\mathcal{C}$, $\mathcal{F}$, $\eta$, $t_{\max}$)
 2    create a copy $\mathcal{C}'$ of $\mathcal{C}$;
 3    foreach component $g \in \mathcal{C}'$ do
 4      replace $g$ by $g'[g, f_g, \mathcal{F}]$;
 5    done;
 6    for $t = 1 \ldots t_{\max}$ do
 7      unroll $\mathcal{C}'$ and $\mathcal{C}$ for $t$ cycles;
 8      force at least one pair of POs to
             different values;
 9      convert to SAT instance;
10      for $k = 1 \ldots \eta$ do
11        constrain $\sum f_g = k$;
12        while (satisfiable) do
13          $G = \{g | f_g == 1\}$;
14          $\mathcal{T} := \mathcal{T} \cup G$;
15          add constraint $\bigvee_{g \in G}(f_g == 0)$;
16        done;
17      done;
18    done;
19    $\mathcal{S} := \mathcal{C} \setminus \mathcal{T}$;
20    return $\mathcal{S}$;
21  end function;
```

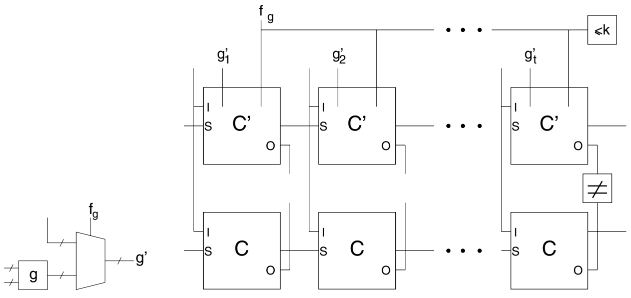
Figure 1. Algorithm to compute robustness

Figure 1 shows the overall flow in pseudo code. The algorithm computes the robustness of a circuit $\mathcal{C}$ with respect to fault model $\mathcal{F}$ and η-fold faults. At first all non-robust parts up to size η are determined and collected, then $\mathcal{S}$ is calculated. First a copy $\mathcal{C}'$ of $\mathcal{C}$ is created (line 2). As shown in Figure 2(a) a fault predicate f_g is associated with each component $g \in \mathcal{C}'$ (lines 3-4). If $f_g = 1$, the function of g is modified; otherwise g behaves as in the fault free case. Next, the sequential equivalence check of $\mathcal{C}'$ and $\mathcal{C}$ is performed. Both circuits are "unrolled" for t time steps (line 7). The fault predicate of each component remains the same for all time steps to reduce the complexity. Moreover, a difference at least at one pair of primary outputs of the two circuits is forced (line 8). The result is illustrated in Figure 2(b). This SAT instance is only satisfiable if the modification of a component causes different output responses of the circuits. If in-equivalence cannot be shown, the number of time steps considered is increased up to $t_{\max}$ (line 6). To guarantee that all components are calculated, the modification of which causes faulty behavior, $t_{\max}$ has to be at least equal to the maximal sequential depth of a product automaton of $\mathcal{C}$ and $\hat{\mathcal{C}} \in \mathbb{C}_{\mathcal{C},\mathcal{S},\mathcal{F},\eta}$.

Now, by calculating all satisfying assignments (lines 10-17), all components are determined that cause faulty behavior when modified according to $\mathcal{F}_N$. Additionally, the number of fault predicates set to 1 is restricted to at most k (line 11) and iteratively incremented up to η (line 10). By this, all non-robust subcircuits up to η-fold faults are retrieved. These non-robust components are joined into the set $\mathcal{T}$ (line 14). The complement set of $\mathcal{T}$ with respect to $\mathcal{C}$ yields the set $\mathcal{S}$ of Definition 3 (line 20).

The algorithm presented so far is restricted to $\mathcal{F}_N$. This results from the modification of a component g as shown in Figure 2(a). In the faulty case $f_g = 1$, the component g may behave non-deterministically like a primary input. For fault models $\mathcal{F}_C$ and $\mathcal{F}_L$ additional constraints are necessary that force g to behave deterministically.

For fault model $\mathcal{F}_C$ this means in more detail: If the assignment of state bits and primary inputs in time step t is



(a) Component (b) Structure

Figure 2. SAT instance

equal to that in time step t' then the output value of g has to be identical in both time steps.

For fault model $\mathcal{F}_L$ deterministic behavior is only required with respect to the direct predecessors of g.

From the previous explanation it is clear that the algorithm in Figure 1 computes the robustness of a circuit.

Theorem 2 *A circuit $\mathcal{C}$, a fault model $\mathcal{F}$ and a positive integer η are given. Furthermore let $\mathcal{S}$:=*largestRobustPart*($\mathcal{C}$, $\mathcal{F}$, η, $t_{\max}$). The circuit $\mathcal{C}$ has a robustness of $R_{\mathcal{F},\eta} = \frac{|\mathcal{S}|}{|\mathcal{C}|}$ if $t_{\max}$ is larger or equal to the sequential depth of the product automaton of $\mathcal{C}$ and $\mathcal{C}'$.*

Instead of calculating the exact robustness, the determination of an upper bound for $R_{\mathcal{F},\eta}$ is possible. For this purpose $t_{\max}$ is set to a smaller value than the sequential depth of $\mathcal{C}$ and $\hat{\mathcal{C}} \in \mathbb{C}_{\mathcal{C},\mathcal{S},\mathcal{F},\eta}$.

5 Using Induction

Sequential equivalence checking needs a large amount of resources regarding time and memory. Using property checking, the problem can be decomposed.

Often a fault tolerant circuit includes logic to signal the occurrence of an internal malfunction. This functionality can be instrumented to decompose the sequential equivalence checking problem. An inductive proof is applied that consists of multiple formal properties. Each individual property only argues over a few cycles. The base of this proof is an invariant that describes the fault free state of the system. The robustness of the circuit is then calculated with almost the same algorithm as introduced above. Instead of the fault free circuit, the property is used as the reference to model correct behavior. A disadvantage of this approach is that it is not fully automatic. The properties and, especially, the invariant (to avoid reachability analysis) have to be determined manually for each circuit.

The inductive proof is structured as follows:

1. **Precondition:**

   Starting from the initial state, the system state is captured by an invariant *Inv* in the fault free case.

2. **Step:**

   The assumption is that no fault occurred so far, i.e. the invariant *Inv* is valid. Then, a case split is done for the fault free and the faulty case.

   (a) There occurs no fault.

   A property proves that the circuit transitions from a fault free state into another fault free state and that the logic for fault detection does not signal a malfunction, i.e. the invariant *Inv* is verified.

   (b) A fault occurs.

   A property proves that a transition into a state that is unreachable if no fault occurs is recognized by the fault detection logic, i.e. if the invariant *Inv* becomes invalid, the occurrence of a fault is signaled.

The precondition and case (a) of the induction step are proven by a traditional property checker. Step 2.(b) requires the modeling technique presented in Section 4.

Knowing *Inv* makes reachability analysis for faulty circuits superfluous. The objective is to prove that any transition into a – in the fault free case – unreachable state is detected. The number of time steps that have to be considered depends on the functionality of the fault detection logic. In the simplest case, each occurrence of an unreachable state is detected immediately. Then the consideration of a single time step is sufficient.

6 Experimental Results

In the following several robust and non-robust circuits are considered. All experiments are carried out with respect to the fault model $\mathcal{F}_N$. All run times are measured on an AMD Athlon 64 3500+ with 1 GB running Linux. The resources were restricted to 1800 seconds of CPU time and to 768 MB of memory. A time out is denoted by 'TO' in all tables.

Three series of experiments were performed. First, ISCAS benchmark circuits that are typically non-robust are analyzed to evaluate the performance of the algorithm. Second, robust circuits were created and the robustness was determined. Finally, the inductive approach using property checking is exemplary carried out for one benchmark.

6.1 ISCAS'89 Benchmarks

Results for some of the ISCAS'89 benchmarks are shown in Table 1. Only "interesting data" is reported, i.e. for those benchmarks that provide some insight into the algorithms and where at least one algorithm finished. The $(\mathcal{F}_N, \eta)$-robustness of the circuits is determined for $\eta \in \{1, 2\}$. Here $t_{\max}$ was not determined analytically; fixed values considered instead, i.e. 5, 10 and 15. Given are the name of the circuit ($\mathcal{C}$), the value used for $t_{\max}$, the number of gates contained in the overall problem instance

Table 1. Run times for ISCAS benchmarks

					single			double		
$\mathcal{C}$	$t_{\max}$	#gt	tm$_V$	$\vert\mathcal{C}\vert$	$\vert\mathcal{S}\vert$	$R_{\mathcal{F}_N,1}$	tm$_1$	$\vert\mathcal{S}\vert$	$R_{\mathcal{F}_N,2}$	tm$_2$
s1196	5	1058	0.6	529	0	0.0%	3.6	0	0.0%	3.7
s1196	10	1058	0.6	529	0	0.0%	3.6	0	0.0%	3.8
s1196	15	1058	0.6	529	0	0.0%	3.6	0	0.0%	3.8
s1238	5	1016	0.5	508	0	0.0%	2.9	0	0.0%	3.0
s1238	10	1016	0.5	508	0	0.0%	2.8	0	0.0%	3.0
s1238	15	1016	0.5	508	0	0.0%	2.8	0	0.0%	2.9
s1423	5	1314	556.7	657	10	2.0%	1378.0	-	-	TO
s1423	10	1314	TO	657	0	0.0%	135.0	-	-	TO
s1423	15	1314	TO	657	0	0.0%	608.1	-	-	TO
s1488	5	1306	4.0	653	0	0.0%	8.1	0	0.0%	9.8
s1488	10	1306	117.6	653	0	0.0%	21.8	0	0.0%	41.1
s1488	15	1306	638.4	653	0	0.0%	58.3	0	0.0%	122.5
s1494	5	1294	2.1	647	0	0.0%	9.6	0	0.0%	10.9
s1494	10	1294	121.8	647	0	0.0%	20.2	0	0.0%	44.9
s1494	15	1294	680.4	647	0	0.0%	44.7	0	0.0%	127.7
s298	5	238	0.1	119	0	0.0%	0.2	0	0.0%	0.2
s298	10	238	0.4	119	0	0.0%	0.6	0	0.0%	0.7
s298	15	238	1.9	119	0	0.0%	3.0	0	0.0%	3.6
s382	5	316	0.1	158	2	1.0%	0.3	2	1.0%	0.3
s382	10	316	1.9	158	0	0.0%	0.9	0	0.0%	3.8
s382	15	316	26.9	158	0	0.0%	1.3	0	0.0%	87.2
s400	5	324	0.1	162	2	1.0%	0.3	2	1.0%	0.3
s400	10	324	1.5	162	0	0.0%	0.8	0	0.0%	3.6
s400	15	324	38.0	162	0	0.0%	1.5	0	0.0%	35.3
s420	5	436	0.0	218	0	0.0%	0.5	0	0.0%	0.5
s420	10	436	0.3	218	0	0.0%	0.9	0	0.0%	1.5
s420	15	436	8.7	218	0	0.0%	1.2	0	0.0%	5.8
s4863	5	4684	TO	2342	0	0.0%	255.9	-	-	TO
s4863	10	4684	TO	2342	0	0.0%	1600.4	-	-	TO
s4863	15	4684	TO	2342	-	-	TO	-	-	TO
s510	5	422	0.4	211	0	0.0%	1.0	0	0.0%	1.5
s510	10	422	35.8	211	0	0.0%	3.1	0	0.0%	16.2
s510	15	422	236.8	211	0	0.0%	7.9	0	0.0%	66.9
s526	5	386	0.3	193	0	0.0%	0.8	0	0.0%	1.0
s526	10	386	23.7	193	0	0.0%	13.4	0	0.0%	41.1
s526	15	386	831.0	193	0	0.0%	13.6	0	0.0%	1023.5
s526n	5	388	0.3	194	0	0.0%	0.8	0	0.0%	1.0
s526n	10	388	21.3	194	0	0.0%	12.9	0	0.0%	40.5
s526n	15	388	975.9	194	0	0.0%	14.5	0	0.0%	791.8
s5378	5	5558	16.0	2776	137	5.0%	393.4	124	4.0%	415.5
s5378	10	5558	249.7	2779	-	-	TO	-	-	TO
s5378	15	5558	TO	2779	-	-	TO	-	-	TO
s641	5	758	6.4	379	0	0.0%	3.2	0	0.0%	35.8
s641	10	758	TO	379	0	0.0%	13.2	-	-	TO
s641	15	758	TO	379	0	0.0%	21.4	-	-	TO

(#gt), and the run time in seconds to verify the sequential equivalence of two correct versions of the circuit for $t_{\max}$ time frames (tm$_V$). This was done by unrolling the two circuits into a SAT instance (i.e. without exploiting internal equivalence, structural knowledge etc.). Then, data for calculating the robustness with respect to single and double faults is reported. The number of gates in the faulty circuit ($|\mathcal{C}|$), the number of components not contained in any non-robust subcircuit ($|\mathcal{S}|$), the robustness ($R_{\mathcal{F}_N,\eta}, \eta \in \{1,2\}$), and the run time in seconds (tm$_\eta, \eta \in \{1,2\}$) are presented.

The ISCAS circuits are non-robust, i.e. $R_{\mathcal{F}_N,\eta} = 0, \eta \in \{1,2\}$). On first sight this seems to contradict the known number of untestable *Stuck-At Faults* (SAF) in these circuits. An untestable SAF is a signal line with a constant value. In $\mathcal{F}_N$ this value can be switched non-deterministically – which usually changes the behavior of the circuit.

In most cases the exact robustness of 0 can already be determined for $t_{\max} = 5$. Only for a few exceptions a larger value is needed (e.g. s1423, s382, s400).

The run time is tightly correlated to the value of $t_{\max}$. In comparison to the plain equivalence check, checking the robustness for a small value of $t_{\max}$ typically needs a longer run time. For $t_{\max} = 10$, robustness checking is often faster. In our interpretation this is due to the benchmark circuits that are “highly non-robust”: injecting faults leads to easily satisfiable problem instances that are efficiently handled by the SAT solver.

Robustness checking with respect to double faults is always slower than that for single faults – often by more than one factor of magnitude. The search space grows exponentially with the number of faults. Nonetheless, especially for long run times, sequential equivalence checking takes even longer than robustness checking (e.g. s1494, s420).

6.2 Robust Circuits

Results for two robust benchmark circuits are reported in Table 2. Robustness was achieved by *Triple Modular Redundancy* (TMR). The output values are determined by taking the majority of three instances of the circuit. The reported data are the same as previously with the exception of $\#cmp$ and $|\mathcal{C}|$. Here, the problem instances were derived from Verilog code and an expression in the Verilog code was considered as a single component [12].

The circuit *r_s1269* is a TMR version of the ISCAS’89 benchmark *s1269*. This circuit could only be handled for $t_{\max} = 5$. Only faults in non-redundant parts of the circuit (e.g. the reset logic) may cause incorrect behavior. As a result a robustness of 99.4% is achieved. The robustness significantly drops to 14.8% for double faults. But this is only an upper bound as a higher value of $t_{\max}$ may yield more inconsistencies.

The circuit *rCounter* is a counter with TMR. This circuit contains fault detection logic. If the internal value of one instance deviates, a fault is signaled. Therefore a deviation from the specification is detected immediately for single faults. As a result the circuit is $(\mathcal{F}_N, 1)$-robust. Again, some parts of the circuit are not redundant. Therefore the robustness is below 100%. The robustness with respect to double faults is only 1.9%.

The run time for robustness checking is quite high and increases drastically for larger values of $t_{\max}$. In particular the maximal sequential depth of the product automaton of the fault free circuit and the faulty circuit cannot be met to determine the exact robustness. For this purpose an improvement of the efficiency of the technique is necessary.

In case of *rCounter* this can be done by using the inductive approach to exploit the fault detection logic. The precondition and case (a) of the induction step are proven using a property checker. Table 3(a) shows the results. The number of components contained in the problem instance is larger compared to Table 2. This is due to the fault detection logic that is contained in the problem instance now.

Table 2. Run times for robust circuits

				single				double			
$\mathcal{C}$	$t_{\max}$	#cmp	tm$_V$	$\|\mathcal{C}\|$	$\|\mathcal{S}\|$	$R_{\mathcal{F}_N,1}$	tm$_1$	$\|\mathcal{C}\|$	$\|\mathcal{S}\|$	$R_{\mathcal{F}_N,2}$	tm$_2$
r_s1269b	5	2459	14.59s	933	927	99.0%	2727.68s	933	-	-	TO
rCounter	5	204	0.18s	52	50	96.0%	0.79s	52	1	2.0%	3.99s
rCounter	10	204	1.99s	52	50	96.0%	9.24s	52	1	2.0%	78.31s
rCounter	15	204	13.25s	52	50	96.0%	97.61s	52	1	2.0%	3966.72s

Table 3. Induction

(a) Induction: precondition and step, case(a)

				tm$_V$	
$\mathcal{C}$	$\|\mathcal{C}\|$	#FF	#gt	prec.	case (a)
rCounter	62	25	370	<0.1s	<0.1s

(b) Step; case(b)

η	$\|\mathcal{S}\|$	$R_{\mathcal{F}_N,\eta}$	tm$_\eta$
single	62	100%	0.2s
double	40	65%	0.43s
triple	8	13%	4.14s

The numbers of flip flops (#FF) and gates (#gt) are also reported. Finally, the run time in CPU seconds to verify the precondition (prec.) and case (a) of the induction step (case (a)) are shown. These times are moderate because only two time frames are considered within the property. Both steps are independent of the number of faults considered.

Robustness checking is performed in case (b) of the induction step. Table 3(b) provides the data for single, double and triple faults. For single faults 100% robustness is determined. In contrast to the sequential equivalence check, a fault in the reset logic is not detected because the property asserts that no reset occurs. For double faults 65% robustness is determined which is much higher than the value computed by equivalence checking. Faults often either occur in the reset logic or in one instance of the counter leading to an increased robustness. For triple faults 13% robustness is retrieved – again, these are faults in the reset logic that are not covered by the property. The robustness derived from property checking depends on the particular formulation of the property. Therefore the interpretation is more difficult compared to equivalence checking.

The most important result is the drastic reduction of run time using the inductive approach. Even for triple faults the run time is acceptable.

Overall, measuring robustness by implicitly enumerating all faults is possible. A significant improvement of the efficiency is achieved by using induction.

7 Summary

An approach to automatically calculate the robustness of circuits was proposed. Using sequential equivalence checking a fully automatic flow is created. The run time to calculate the robustness is significant in this case. By an inductive approach the run time can be reduced. The method is only semi-automatic in this case, because the corresponding properties have to be created manually.

The presented algorithm still suffers from the high computational complexity of the underlying problem. Therefore, future work mainly aims at improving the efficiency of the process, e.g. by applying random simulation or sequential test generation as a preprocessing step to determine non-robust circuit areas. The consideration of other techniques than TMR to achieve robustness is another important direction of future work. Finally, the exact relationship between the fault models and empirical data from in-field failures has to be investigated.

References

[1] P. Civera, L. Macchiarulo, M. Rebaudengo, M. S. Reorda, and M. Violante. An FPGA-based approach for speeding-up fault injection campaigns on safety-critical circuits. *Jour. of Electronic Testing: Theory and Applications*, 18(3):261–271, 2002.

[2] S. Cook. The complexity of theorem proving procedures. In *3. ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[3] M. Davis, G. Logeman, and D. Loveland. A machine program for theorem proving. *Comm. of the ACM*, 5:394–397, 1962.

[4] 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*, pages 176–181, 2006.

[5] R. Leveugle. A new approach for early dependability evaluation based on formal property checking and controlled mutations. In *IEEE International On-Line Testing Symposium*, pages 260–265, 2005.

[6] J. Marques-Silva and K. Sakallah. Conflict analysis in search algorithms for propositional satisfiability. In *IEEE International Conference on Tools with Artificial Intelligence*, 1996.

[7] M. Miskov-Zivanov and D. Marculescu. Circuit reliability analysis using symbolic techniques. *IEEE Trans. on CAD*, 25(12):2638–2649, 2006.

[8] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Design Automation Conf.*, pages 530–535, 2001.

[9] C. Pixley. A theory and implementation of sequential hardware equivalence. *IEEE Trans. on CAD*, 11(12):1469–1478, 1992.

[10] A. Smith, A. Veneris, M. Ali, and A. Viglas. Fault diagnosis and logic debugging using boolean satisfiability. *IEEE Trans. on CAD*, 24(10):1606–1621, 2005.

[11] A. Smith, A. Veneris, and A. Viglas. Design diagnosis using Boolean satisfiability. In *ASP Design Automation Conf.*, pages 218–223, 2004.

[12] S. Staber, G. Fey, R. Bloem, and R. Drechsler. Automatic fault localization for property checking. In *Haifa Verification Conference*, volume 4383 of *LNCS*, pages 50–64. Springer, 2006.

[13] G. Tseitin. On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125, 1968. (Reprinted in: J. Siekmann, G. Wrightson (Ed.), Automation of Reasoning, Vol. 2, Springer, Berlin, 1983, pp. 466-483.).

[14] Q. Zhou and K. Mohanram. Gate sizing to radiation harden combinational logic. *IEEE Trans. on CAD*, 25(1):155–166, 2006.