A Better-Than-Worst-Case Robustness Measure

Stefan Frehse Görschwin Fey Rolf Drechsler
Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
{sfrehse,fey,drechsle}@informatik.uni-bremen.de

***Abstract*—In presence of increasing soft error rates due to shrinking feature sizes, design tools are required to analyze fault tolerance and robustness of circuits.**

Here, we propose a new measure that identifies hot-spots in the design. On the one hand the measure is more accurate than a "worst-case analysis" that ignores excitation probabilities. On the other hand the computation of the new measure is more efficient than a "probabilistic analysis" that considers excitation probabilities at the cost of a higher computational complexity. Both of these extremes can be embedded in the new measure. Experimental results on circuits with protection against soft errors show that the new measure can be calculated effectively.

I. Introduction

Continuously shrinking feature sizes of digital circuits allow for the integration of more and more components in a single chip. Shrinking feature sizes have several positive side effects, e.g., low-power circuitry operating at high frequency. But there are substantial drawbacks as well. Manufacturing failures and transient faults may increasingly tamper the functionality, consequently the *Soft Error Rate* (SER) increases. Precautions against soft errors are taken at different levels, e.g., architectural level, algorithmic level, or layout level [1]. But the implementation of these techniques has to be verified. During implementation bugs may be introduced. Thus, checking the fault tolerance of a given implementation early in the design flow becomes an important step in the design process. Several formal and non-formal verification methods have been applied for this purpose.

Non-formal methods, like simulation or emulation [2] perform fault injection to check for a limited number of simulation traces whether those lead to faulty behavior. Due to the simulation-based nature, this is a fast but not a complete method with respect to all potential scenarios or all faults.

Formal methods can cover all inputs and any fault covered by the given fault model. Various methods have been proposed based on symbolic methods using *Binary Decision Diagrams* (BDDs) [3]–[6] or *Boolean Satisfiability* (SAT) [7]–[9]. The approaches of [6], [8], [9] are most tightly related to our approach.

Based on the *Stuck-At Fault Model* (SAFM) the method in [6] computes the probability for transient faults to be propagated to primary outputs of combinational circuits. The given theoretical extension to sequential circuits is too complex to be applied in practice as the analysis relies on BDDs. Internally, all testpatterns are calculated for each fault to be considered. By this, the probability of applying a testpattern for a certain fault can be calculated. In the following we refer to [6] as a *probabilistic approach*. In [10] the authors proposed an alternative probabilistic approach to approximate the signal probabilities for combinational circuits.

This work has been funded in part by DFG grants DR 287/19-1 and FE 797/5-1.

The work in [9] analyzes the *self-checking fault secureness* [11] for combinational circuits and provides a robustness measure with a lower and an upper bound for the SAFM. This method analyzes a circuit model by using ATPG algorithms. For the analysis assumptions on the environment and the checker functionality of the circuit are required.

In [8] a model similar to *Bounded Model Checking* (BMC) has been proposed for the analysis of soft errors. As well as [9] this method provides a lower and an upper bound for the robustness. Both methods can be seen as a worst-case analysis: a component is classified non-robust if there exists at least a single testpattern, such that a fault of this component may change the output behavior. We call this classification *worst-case analysis*, i.e. the probability for excitation and propagation is ignored by the robustness measure. In contrast the *probabilistic approach* of [6] considers all testpatterns. But this approach relies on BDDs for the analysis restricting the application to very small circuits.

In this work we propose a robustness measure that constitutes a trade-off between the worst-case analysis and the consideration of all testpatterns. Thus, a limited number of testpatterns that show the faulty computation is considered. Therewith a more detailed view of the robustness is given, which can be computed in feasible run time. We use the model of [8] which considers soft-errors in sequential circuits.

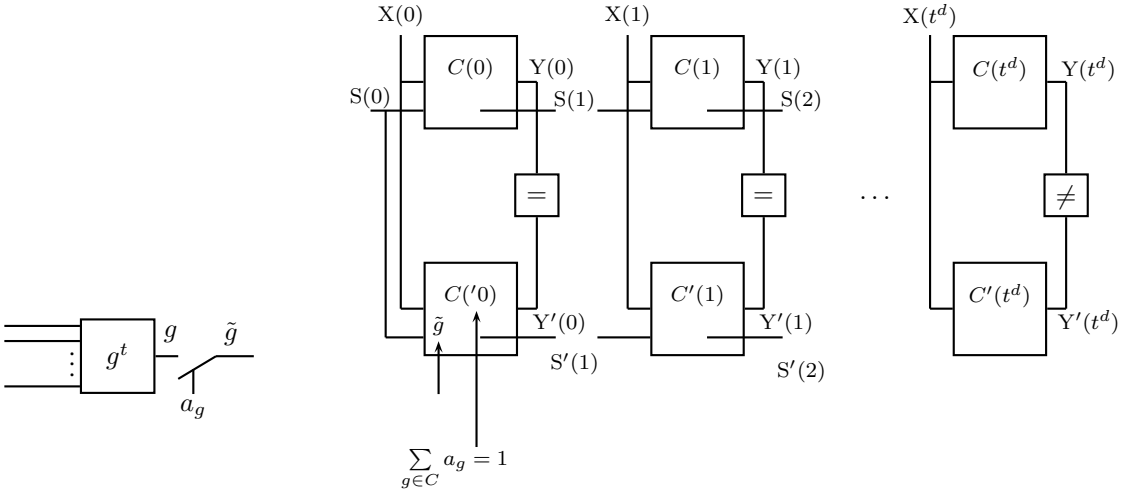
Technically, a sequential ATPG engine is used to consider a bounded time window for the calculation. When calculating only a single testpattern for each soft error, worst-case analysis is performed. By calculating all testpatterns the probabilistic analysis is performed at high computational costs. Our approach finds up to a predefined number of testpatterns. As a result hot-spots in the circuit are identified, that are easily sensitized to propagate soft errors. This knowledge guides the designer to increase the robustness of the circuit. The previous measures, worst-case analysis and usage of all input stimuli, can be embedded into the new measure.

The remainder of this paper is structured as follows: Section II reviews the preliminaries. The underlying circuit model as well as the approach of robustness computation based on [8] are described. Section III introduces the new measure in detail. An algorithm is presented in Section IV that computes the testpatterns. Experimental results are reported in Section V. Finally, conclusions are stated in Section VI.

II. Preliminaries

A. Boolean Satisfiability

The *Boolean Satisfiability* (SAT) problem is a well-known $\mathcal{NP}$-complete decision problem [12] that asks whether a Boolean formula $f : \mathbb{B}^n \rightarrow \mathbb{B}$ is *satisfiable* or not (*unsatisfiable*). If a formula f is satisfiable, a satisfying value assignment of the variables can be found. Typically, the problem is given in *Conjunctive Normal Form* (CNF), whereas every



(a) Fault injection

(b) Sequential model

Fig. 1. Sequential model with fault injection logic

Boolean formula can be converted into a CNF. Nowadays, large formulas related to practical problems with millions of clauses and variables can usually be solved in feasible time by state-of-the-art SAT solvers [13], [14].

In this work multiple satisfying assignments are of interest. These assignments are computed by iteratively calling the SAT solver and adding *blocking clauses*. A blocking clause excludes a (previously computed) solution.

B. Circuit Model

We consider sequential and combinational circuits C with *Primary Inputs* $\mathrm{PI}(C)$, *Primary Outputs* $\mathrm{PO}(C)$ and *State Elements* $\mathrm{S}(C)$. For combinational circuits $S(C) = \emptyset$ holds. Furthermore a circuit consists of various components. For example, such components can be primitive gates (*AND*, *OR*, etc.), modules (*Adder*, *Multiplier*, etc.) or statements of a hardware description language (`if (...) then ...endif;`, etc.). The number of components of a circuit C is denoted by $|C|$. A circuit C can be converted into CNF in time and space linear to $|C|$ [15].

C. Classification of Components

As a basis for the robustness computation we use the following classification of each component $g \in C$ as presented in [8]. A soft error in component g is either a bit-flip from 0 to 1 or from 1 to 0 on one or on multiple output signals of g[1]. A component $g \in C$ belongs to one of three disjoint classes [8]:

1) *non-robust* – A soft error of component g leads to an abnormal output behavior under at least one input trace τ^t for $\mathrm{PI}(C)$ at a certain time frame t, i.e. the primary outputs differ from the fault-free computation. Additionally, if the circuit is equipped with a fault detection signal *flt*, this signal does not report a fault, i.e. *flt* $= 0$. The input trace τ^t is also called a *testpattern* for the soft error.
2) *non-classified* – The classification is similar to the non-robust classification, with the exception that the states differ from the fault-free states but the primary outputs are equal. This case shows *Silent Data Corruption* (SDC).
3) *robust* – A soft error on component g is reported by the fault detection signal (*flt* $= 1$) or is corrected by the internal logic. Consequently, the primary outputs are correct with respect to the fault-free computation under every possible input.

All components of the circuit are partitioned into the set $\mathbb{T}$ of robust components, the set $\mathbb{S}$ of non-robust components and the set $\mathbb{U}$ of non-classified components, i.e. $C = \mathbb{S} \cup \mathbb{T} \cup \mathbb{U}$. For combinational circuits the set $\mathbb{U}$ is empty, because without state elements the second case above is not applicable.

D. Sequential Modeling

In [8] the sets $\mathbb{T}, \mathbb{S}$ and $\mathbb{U}$ were computed by using a SAT solver. Basically, a *Sequential Equivalence Check* (SEC) of a circuit C and a circuit C' with additional logic to model soft errors is performed. The circuit is unrolled up to a given time limit t^d. The schematic view of the model is shown in Figure 1. A soft error is assumed to be corrected or detected (i.e. signalized by a fault signal *flt*) within a short period of time or the fault remains undetected. Consequently, the analysis can be bounded by a certain time limit t^d. Combinational circuits can be easily embedded in the sequential model.

For the approach in [8] it is sufficient to compute at least one testpattern as mentioned in Section II-C for a component g, to classify g as non-robust. Given the sets $\mathbb{T}, \mathbb{S}$ and $\mathbb{U}$ at a certain time frame $\mathbf{t}$ the robustness is defined as follows:

$$R^{\mathbf{t}}_{\mathrm{lb}} = \frac{|\mathbb{T}^{\mathbf{t}}|}{|C|} = 1 - \frac{|\mathbb{S}^{\mathbf{t}}| + |\mathbb{U}^{\mathbf{t}}|}{|C|}$$

$$R^{\mathbf{t}}_{\mathrm{ub}} = \frac{|\mathbb{U}^{\mathbf{t}}| + |\mathbb{T}^{\mathbf{t}}|}{|C|} = 1 - \frac{|\mathbb{S}^{\mathbf{t}}|}{|C|}$$

[1]For simplicity we consider both cases as the same fault, i.e. a soft error at g. If a finer differentiation is required this can be easily integrated into the algorithms.

For combinational circuits the measure yields $R^{\mathbf{t}}_{\mathrm{lb}} = R^{\mathbf{t}}_{\mathrm{ub}}$, since $|\mathbb{U}| = 0$ and $\mathbf{t} = 0$. For sequential circuits the lower and upper bound may differ even if an unlimited number of time frames would be considered. In particular, if SDC occurs and the faulty system state is not corrected, the divergence between the faulty system and the fault free system may persist.

As mentioned before for this robustness measure it is sufficient to find one testpattern to classify g as non-robust, or to prove that no testpattern exists – the component is robust. This robustness measure considers a *Single Testpattern* (ST). In the following we denote this by $R^{\mathbf{t}}_{\mathrm{ST}}$ as well as $R^{\mathbf{t}}_{\mathrm{ST,lb}}$ and $R^{\mathbf{t}}_{\mathrm{ST,ub}}$ for lower bound and upper bound at time frame $\mathbf{t}$, respectively.

III. Analysis using Multiple Testpatterns

In the following we illustrate drawbacks of the measure introduced in Section II using an example. Next, the new measure is introduced that overcomes those limitations using an analysis based on *Multiple Testpatterns* (MT). Then, the relation to previously defined measures is analyzed.

A. Motivating Example

The robustness measure discussed in Section II can be considered as a "worst-case analysis": a component is considered non-robust as soon as there exists a single testpattern that shows faulty behavior of this component at least at one primary output. The probability to apply such a pattern, i.e. the excitation probability for the fault, is ignored. Consider the following example.

Example 1: Consider a combinational circuit C with four primary inputs, i.e. $|\mathrm{PI}(C)| = 4$. Furthermore, let $a, b \in C$ be two non-robust and $c, d, e \in C$ be robust components. The worst-case analysis yields $R_{\mathrm{ST}} = 3/5 = 60\%$.

Further assume, that there are only two testpatterns that excite a fault in a, denoted by $\psi(a) = 2$. Given the total number of $2^{|\mathrm{PI}(C)|} = 2^4 = 16$ input traces, the probability to excite the fault in a is only $2/16 = 12.5\%$.

Moreover, let any input trace be a testpattern for a fault at b, i.e. $\psi(b) = 16$ and the excitation probability at b is 100%.

The worst-case analysis does not differentiate the two components. Both are simply classified as non-robust, even though b can be considered as a hot-spot while a is relatively save.

The exact computation of excitation probabilities along the lines of [6] would overcome this limitation. But in that case all testpatterns – potentially a number exponential in the number of primary inputs – have to be found. In [6] a BDD-based symbolic analysis was used for this purpose. But a BDD-based analysis is typically limited to small circuits.

B. New Robustness Measure

Instead of considering only a single testpattern per component, the new robustness measure takes *Multiple Testpatterns* (MT) into account. By this, components that only have a few testpatterns can be differentiated from those components having many testpatterns. As a result a grading of the non-robust components is achieved which can be utilized to identify hot-spots in the circuit.

In the following the combinational case is considered first, i.e. lower bound and upper bound for the robustness are identical. Let $\psi(g)$ denote the number of testpatterns at component g. Then, the quotient between the number of testpatterns $\psi(g)$ and all input traces $\Psi = 2^{|\mathrm{PI}(C)|}$ yields the excitation probability for a soft error on g:

$$e(g) = \frac{\psi(g)}{\Psi}$$

Or alternatively, as a measure for robustness, the probability that a soft error is not observable, is given by:

$$f(g) = 1 - e(g) = 1 - \frac{\psi(g)}{\Psi}$$

As explained, calculating $\psi(g)$ is often too expensive. Instead we limit the number of testpatterns by a user defined parameter $0 < \lambda \leq 1$. If the portion of input traces that are testpatterns exceeds λ, a component is considered non-robust. For such hot-spots no further differentiation is required. Only below this percentage a more fine grained resolution is determined. Then, the robustness of a component g is determined by

$$r(g, \lambda) = 1 - \frac{\min\{\psi(g), \lceil \lambda \Psi \rceil\}}{\lceil \lambda \Psi \rceil}. \tag{1}$$

Consequently, the robustness of the circuit is measured by

$$R_{\mathrm{MT}}(\lambda) = \frac{1}{|C|} \sum_{g \in C} r(g, \lambda). \tag{2}$$

Example 2: Consider again Example 1 and let $\lambda = 0.5$ such that up to 50% of all input traces are considered. Again let $\psi(a) = 2, \psi(b) = 16$ and $\psi(c) = 0, \psi(d) = 0, \psi(e) = 0$. Then, $r(a, 0.5) = 3/4$, $r(b, 0.5) = 0$ and

$$R_{\mathrm{MT}}(0.5) = \frac{1}{5}(3/4 + 0 + 3) = 75\%.$$

The overall value of the new measure increases compared to the worst-case analysis. More important is the observation that components a and b can be differentiated.

The presented measure can be extended to the sequential case. In this case testpatterns span multiple time frames and up to $\mathbf{t}$ time frames are considered in the analysis. The number of all input traces is given by $\Psi = 2^{|\mathrm{PI}(C)| \cdot \mathbf{t}}$. Also a component may cause SDC and is considered non-classified in this case. Such components are collected in a set $\mathbb{U}$. Then a lower bound and an upper bound for the robustness are determined by assuming that all non-classified components may turn out to be non-robust or robust, respectively. For non-classified components no testpattern can be found that shows a soft error at one of the primary outputs, i.e. $\psi(g) = 0$ holds for $g \in \mathbb{U}$. We retrieve the following bounds:

$$\begin{aligned} R^{\mathbf{t}}_{\mathrm{MT,lb}}(\lambda) &= \frac{1}{|C|} \sum_{g \in C} r(g, \lambda) - |\mathbb{U}| \\ R^{\mathbf{t}}_{\mathrm{MT,ub}}(\lambda) &= \frac{1}{|C|} \sum_{g \in C} r(g, \lambda) \end{aligned}$$

C. Embedding Previous Measures

First, the new measure is compared to the worst-case analysis previously proposed in [8] and described in Section II. Assume that λ' is close to zero. Then, $\lceil \lambda' \Psi \rceil$ as used in Equation (1) becomes 1. In this case the robustness of a component g as defined in Equation (1) becomes:

$$r(g, \lambda') = 1 - \min\{\psi(g), 1\}$$

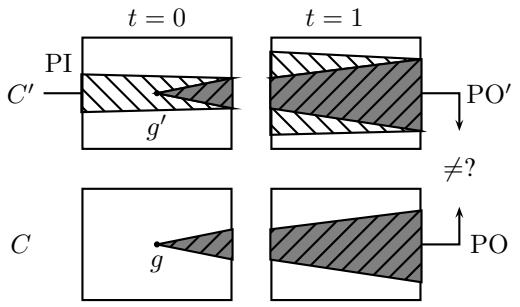


Fig. 2. Sequential-ATPG model

If there exists at least one testpattern, the robustness of a component becomes 0. If no testpattern exists, the component is considered robust. Consequently, for λ close to zero the new measure converges to the one of [8].

The probabilistic approach of [6] considers exact excitation probabilities to determine the robustness of components[2]. This is achieved using our measure by setting λ to 1, i.e. Equation (1) becomes

$$r(g,1) = 1 - \frac{\min\{\psi(g), \Psi\}}{\Psi} = 1 - \frac{\psi(g)}{\Psi} = f(g).$$

This is the probability of a soft error to remain undetected.

IV. Computation

In this section we present an algorithm using a SAT-based sequential ATPG engine to calculate the new robustness measure and discuss potential extensions.

A. Algorithm

The model for sequential SAT-based ATPG is shown in Figure 2. Given are a component g and a limit $\mathbf{t}$ for the number of time frames to be considered. For time frame 0 the fan-out cone of g is modeled. For all outputs in this fan-out cone, the transitive fan-in of these outputs is also modeled until the primary inputs or a state element are reached. In time frame 0 the state elements remain unconstrained during the analysis or they are constrained as proposed in [8]. For time frame 1, the same copy is created. But the traversal does not stop when reaching state elements in time frame 1. Instead the model also includes the driving circuitry of time frame 0 to adequately model the sequential behavior. This process is repeated until time frame $\mathbf{t}$ is reached.

In principle this is a standard procedure for sequential ATPG. In our case faults are only injected in time frame 0, because soft errors are modeled and all states are covered by leaving the initial state unconstrained or allowing all reachable states, respectively. Moreover, the problem instance for analyzing t time frames is reused and extended to analyze $t+1$ time frames. This improves the efficiency as learned information is reused by the SAT solver [16].

The pseudocode to determine the testpatterns for a soft error in a component g is given in Algorithm 1. A circuit C, a considered component g, the current time t, the maximum considered time frame $\mathbf{t}$, the SAT solver object s as well as the parameter λ are given as input to the algorithm. The algorithm creates a SAT instance for the component g at time frame t. If $t > 0$, the existing problem instance is extended as explained above and as shown in Figure 2. Only the primary inputs that are contained in the cone of g are considered. The maximum number of testpatterns to be considered is configured by λ and then stored in Ψ' in line 5. From line 6 to line 9, the testpatterns are computed. While a satisfying assignment exists and the maximum number of testpatterns Ψ' is not exceeded, a new testpattern is extracted from the SAT model and the number of testpatterns is incremented. To compute another testpattern, the currently computed satisfying assignment is excluded by adding a blocking clause that contains values of primary inputs and state elements in time frame $t = 0$. The loop continues until all testpatterns are found or the limit has been reached.

Algorithm 1: computeTestpatterns

Input: circuit C, component g, current time frame t, bounded time frame $\mathbf{t}$, solver object s, accuracy-parameter λ
Output: $\psi(g)$

```
1  begin
2      createOrAppendSATInstance(s, C, g, t);
3      Cone_PI = PI^t(C) ∩ computeTransitiveInputCone(g, t);
4      ψ(g) = 0;
5      Ψ' = ⌈λ · 2^|Cone_PI|⌉;
6      while s.solve() = SATISFIABLE ∧ ψ(g) < Ψ' do
7          ψ(g)++;
8          s.addClause( ⋁_{pi ∈ Cone_PI} ¬s.model(pi));
9      end
10     if s.solve() = UNSATISFIABLE ∧ ψ(g) < Ψ' then
11         return ψ(g) + computeTestpatterns(C, g, t+1, t, s, λ)
12     end
13     else
14         return ψ(g)
15     end
16 end
```

The computation is finished, if 1) the number of considered time frames is exceeded, 2) the SAT instance becomes unsatisfiable, i.e, there are no more testpatterns, or 3) the number of testpatterns exceeds Ψ'. Finally, the number of computed testpatterns is returned.

Further performance improvements can be achieved using a *Minimal Assignment Analysis* (MAA) similar to [17], [18] for each extracted testpattern. Given a testpattern this analysis computes which of the primary inputs justify at least one differing output. Starting from a faulty primary output, one controlling input or all non-controlling inputs are traced at each gate until reaching the primary inputs. This analysis is repeated for the fault signal, if the circuit is equipped with such a signal. The remaining primary inputs not contained in these traces do not influence the output value. These primary inputs can be considered as don't care. Using this analysis the number of SAT calls can be decreased significantly. The number of testpatterns can be calculated by simple arithmetic operations.

B. Discussion

In [8] an algorithm to perform the worst-case analysis was proposed. That algorithm was based on SEC. All potential

[2]Indeed a different fault model has been used in [6], but the fault models can be aligned.

TABLE I
RESULTS FOR COMBINATIONAL CIRCUITS

CIRCUIT			"WORST-CASE"		NEW MEASURE (500 TESTS)				NEW MEASURE (10,000 TESTS)				
NAME	$\|\mathrm{PI}(C)\|$	$\|C\|$	R_{ST}	$\|\mathbb{S}\|$	R_{MT}	λ	t	t (MAA)	R_{MT}	λ	$> \lambda\Psi$	t	t (MAA)
par_5xp1	7	391	88.44%	49	**95.35%**	100.00%	5.96	8.55					
par_9sym	9	655	98.69%	9	**99.71%**	97.66%	3.73	9.6	**99.71%**	100%	0	3.89	9.64
par_apex7	49	720	77.48%	275	**77.48%**	$< 0.01\%$	465.14	32.41	**77.48%**	$< 0.01\%$	204	Ab.	170.75
par_cm42a	4	81	85.71%	15	**92.02%**	100.00%	0.17	0.21					
par_cm82a	5	62	73.17%	22	**86.97%**	100.00%	0.2	0.26					
par_cmb	16	136	63.16%	70	**90.27%**	0.76%	5.55	5.52	**95.91%**	15.25%	6	76.64	44.89
par_comp	32	385	41.36%	285	**41.36%**	$< 0.01\%$	300.79	864.64	–	–		Ab.	Ab.
par_con1	7	65	81.11%	17	**95.01%**	100.00%	0.26	0.36					
par_cordic	23	2866	97.65%	69	**97.65%**	0.01%	524.98	Ab.	–	–		Ab.	Ab.
par_cu	14	166	76.92%	51	**79.63%**	3.05%	11.77	2.86	**92.88%**	61.03%	3	108.04	29.95
par_duke2	22	976	76.23%	255	**76.42%**	0.01%	384.39	620.65	–	–		Ab.	Ab.
par_e64	65	1409	88.45%	193	**89.59%**	$< 0.01\%$	553.68	740.04	–	–		Ab.	Ab.
par_f51m	8	318	91.76%	29	**95.09%**	100.00%	4.62	6.82					
par_frg1	28	393	92.74%	35	**92.74%**	$< 0.01\%$	25.15	13.85	**92.74%**	$< 0.01\%$	35	822.3	324.25
par_rd84	8	1157	82.81%	204	**96.14%**	100.00%	48.73	111.83					
par_rot	135	1932	76.17%	583	**76.17%**	$< 0.01\%$	4550.92	170.28	**76.17%**	$< 0.01\%$	583	Ab.	556.64
par_sao2	10	502	82.16%	96	**94.41%**	48.83%	19.39	46.65	**96.96%**	100%	0	21.24	50.74
par_sqrt8ml	8	447	60.80%	187	**91.92%**	100.00%	19.46	54.26					
par_squar5	5	273	80.87%	57	**91.04%**	100.00%	1.2	1.71					
par_t481	16	1752	99.11%	16	**99.11%**	0.76%	62.64	450.69	**99.11%**	15.26%	16	1304.48	Ab.
par_table5	17	1455	70.58%	448	**75.07%**	0.38%	858.44	1963.05	–	–		Ab.	Ab.

faults are modeled in a single problem instance and learned information can potentially be reused for all other faults. Preliminary experiments have shown that this is more efficient in some cases than using a large number of ATPG calls. But, here we also calculate multiple testpatterns for each fault. Therefore we chose an ATPG engine to keep the size of the problem instances smaller.

V. EXPERIMENTAL RESULTS

This section presents the evaluation of the new robustness measure. Combinational circuits were taken from the LGsynth93 benchmark suite and sequential circuits from the ITC'99 benchmark suite, respectively. For every circuit a parity checker was implemented and optimized using SIS [19]. The parity is checked at the primary outputs and on the state elements. A wrong parity is reported by setting the fault signal *flt* to 1. The robustness of the parity circuits is less than 100%: fault masking may occur, i.e. a single soft error flips an even number of outputs and state elements.

All experiments were carried out on an AMD Dual-Core Opteron Processor with 32GB main memory under Linux. The algorithm is implemented in C++. As underlying SAT solver MiniSat v2.0 [13] with a feature to allow for incremental satisfiability is used. A time out was set to 5000 CPU seconds. Exceeding this limit is denoted by *Ab.*

A. Combinational Circuits

Table I shows the results for the combinational circuits. The first three columns describe properties of the circuit: the name, the number of primary inputs and the number of gates in the circuit. Note that for combinational circuits SDC cannot occur, i.e. all components are classified as robust or as non-robust. Consequently, there is only a single value for the robustness of such circuits (see Section II-D and Section III-B).

The results of the worst-case analysis, the new measure using 500 testpatterns and the new measure using 10,000 testpatterns are given in the following columns. The parameter λ has been adjusted accordingly. For the worst-case analysis, the robustness value and the number of non-robust components are shown in columns R_{ST} and $|\mathbb{S}|$, respectively. For the new measure, the robustness value R_{MT}, the parameter λ, the overall run time t in CPU seconds without MAA, and the run time t_{MAA} with MAA are shown in the respective columns. Additionally, column $> \lambda\Psi$ gives the number of components having more than 10,000 testpatterns. Blank cells denote that no computation with 10,000 testpatterns was required as all components had less than 500 testpatterns.

The run times are longer for the new measure as usually more than a single testpattern has to be considered before the classification of a non-robust component is completed. For small circuits the use of MAA increases the run time (e.g. for *par_5xp1*). In these cases the SAT solver efficiently enumerates multiple solutions. In contrast, when the number of inputs increases, MAA often yields shorter run times as a single solution of the SAT solver is generalized to many testpatterns (e.g. for *par_apex9* and for *par_rot*).

The robustness value for the new measure is typically larger than the one for the worst-case analysis. As soon as a single testpattern exists a component in is classified as "completely non-robust" in the worst-case analysis. For the new measure non-robust components are graded by the number of testpatterns. As long as the number of testpatterns is below the predefined limit, a component contributes to the circuit's robustness. For example, this case occurs for the circuits *par_cmb* and *par_cu*. In such cases a fine grain differentiation between non-robust components is available. The designer decides whether further protection is required for some of these components.

If the number of testpatterns always exceeds the predefined limit, the robustness values are identical for the worst-case analysis and the new measure. For example, this occurs in case of *par_rot* and *par_t481*. All non-robust components must be considered as hot spots and further precautions have to be taken to handle transient faults.

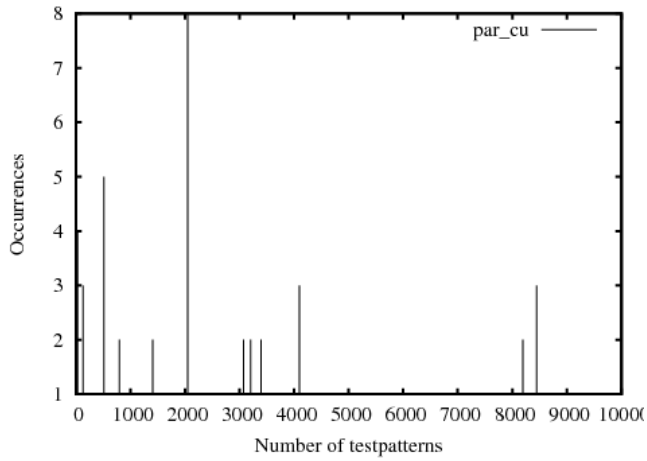


Fig. 3. Histogram for circuit *par_cu*

B. Detailed Example

A more detailed evaluation for the combinational circuit `par_cu` is shown by the histogram in Figure 3. The x-axis depicts the number of testpatterns. The y-axis gives the number of components that had a certain number of testpatterns. In total there are $2^{14} = 16,384$ input traces. The number testpatterns considered was limited to 10,000 as in the previous experiment. The worst-case analysis yields 51 non-robust components. For most of these components there exist less than 10,000 testpatterns. A very fine grain differentiation between non-robust components is determined.

C. Sequential Circuits

Table II shows results for sequential circuits. Up to 10 time frames are analyzed and up to 100 testpatterns are taken into account for the new measure. All components of all circuits are classified, i.e. upper and lower bound are identical for both measures. In the sequential case the number of testpatterns is typically more than 100 such that the predefined limit is exceeded for the new measure. For many of the smaller circuits, a finer differentiation between non-robust components becomes available when using the new measure, e.g. for *par_b06* and *par_b07*.

VI. Conclusion

In this paper we proposed a new robustness measure. This measure constitutes a trade-off between worst-case analysis and a probabilistic approach. The worst-case analysis as well as the probabilistic approach can be embedded in the proposed measure. With the new measure a more detailed view on the robustness can be achieved in feasible run time. Furthermore, the measure identifies hot-spots in the design, i.e. easily sensitizable components. A sequential ATPG engine considering a bounded time window was used to compute the testpatterns efficiently.

VII. Acknowledgment

We would like to thank Andre Sülflow for helpful discussion.

TABLE II
R_{ST}– and $R_{\mathrm{MT}}(\lambda)$–robustness for sequential circuits

Circuit			"worst-case"			New measure (100 tests)			
Name	$\|PI\|$	$\|C\|$	R_{ST}	$\|\mathbb{S}\|$	Time	λ	R_{MT}	$> \lambda\Psi$	Time
par_b01	2	145	**88.57%**	20	0.26s	100.00%	**88.57%**	0	0.61s
par_b02	1	76	**87.76%**	12	0.10s	100.00%	**88.14%**	0	0.10s
par_b03	4	408	**85.14%**	81	8.23s	$< 0.01\%$	**85.14%**	74	21.99s
par_b04	11	2130	**78.95%**	513	131.5s	$< 0.01\%$	**78.95%**	513	290.88s
par_b05	1	2145	**68.56%**	730	114.6s	100.00%	**68.58%**	0	130.07s
par_b06	2	152	**80.60%**	39	0.43s	100.00%	**81.84%**	0	0.76s
par_b07	1	1143	**76.33%**	320	80.89s	78.12%	**84.25%**	14	88.45s
par_b08	9	457	**74.91%**	144	20.33s	$< 0.01\%$	**74.91%**	144	32.73s
par_b09	1	454	**80.59%**	111	6.32s	19.53%	**80.59%**	3	11.99s
par_b10	11	571	**81.32%**	127	12.61s	$< 0.01\%$	**81.32%**	127	23.06s
par_b11	7	1633	**78.72%**	380	1023.44s	$< 0.01\%$	**78.72%**	380	1136.27s

References

[1] C. Zhao and S. Dey, "Improving transient error tolerance of digital VLSI circuits using RObustness COmpiler (ROCO)," in *Int'l Symp. on Quality Electronic Design*, 2006, pp. 133–140.

[2] A. Pellegrini, K. Constantinides, D. Zhang, S. Sudhakar, V. Bertacco, and T. Austin, "CrashTest: A fast high-fidelity FPGA-based resiliency analysis framework," in *Int'l Conf. on Comp. Design*, 2008.

[3] 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*, 2006, pp. 176–181.

[4] M. Miskov-Zivanov and D. Marculescu, "Circuit reliability analysis using symbolic techniques," *IEEE Trans. on CAD*, vol. 25, no. 12, pp. 2638–2649, 2006.

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

[6] J. Hayes, I. Polian, and B. Becker, "An analysis framework for transient-error tolerance," in *VLSI Test Symp.*, 2007, pp. 249–255.

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

[8] G. Fey, A. Sülflow, and R. Drechsler, "Computing bounds for fault tolerance using formal techniques," in *Design Automation Conf.*, 2009, pp. 190–195.

[9] M. Hunger, S. Hellebrand, A. Czutro, I. Polian, and B. Becker, "ATPG-Based grading of strong fault-secureness," in *IEEE International On-Line Testing Symposium*, 2009, pp. 269–274.

[10] I. Polian, S. M. Reddy, and B. Becker, "Scalable calculation of logical masking effects for selective hardening against soft errors," in *IEEE Annual Symposium on VLSI*, 2008, pp. 257–262.

[11] J. Smith and G. Metze, "Strongly fault secure logic networks," *IEEE Trans. on CAD*, vol. 27, no. 6, pp. 491–499, 1978.

[12] S. Cook, "The complexity of theorem proving procedures," in *3. ACM Symposium on Theory of Computing*, 1971, pp. 151–158.

[13] N. Eén and N. Sörensson, "An extensible SAT solver," in *SAT 2003*, ser. LNCS, vol. 2919, 2004, pp. 502–518.

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

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

[16] J. Whittemore, J. Kim, and K. Sakallah, "SATIRE: A new incremental satisfiability engine," in *Design Automation Conf.*, 2001, pp. 542–545.

[17] K. Ravi and F. Somenzi, "Minimal assignments for bounded model checking," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS, vol. 2988, 2004, pp. 31–45.

[18] S. Eggersglüß and R. Drechsler, "Improving test pattern compactness in SAT-based ATPG," in *Asian Test Symp.*, 2007, pp. 445–452.

[19] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," University of Berkeley, Tech. Rep., 1992.