A Hybrid Algorithm to Conservatively Check the Robustness of Circuits

Niels Thole*†, Lorena Anghel‡, Görschwin Fey*†
*Institute of Computer Science, University of Bremen, Bremen, Germany
{nthole, fey}@informatik.uni-bremen.de
†Institute of Space Systems, German Aerospace Center, Bremen, Germany
‡TIMA Laboratory, CNRS-UJF-INPG, Grenoble, France
lorena.anghel@imag.fr

***Abstract*—As systems become more complex, the size of transistors decreases. This effect leads to an increased probability of transient faults as well as higher variability of the transistors. Verifying that circuits are robust against transient faults and variability is mandatory. While formal verification may be used to prove robustness, a model that includes extracted electrical parameters and the corresponding timing information is usually too complex in practice. The contribution of this paper consists in a hybrid algorithm that can decide robustness. The algorithm uses Boolean reasoning as well as simulation to decompose the problem into feasible SAT formulas and still achieves completeness. In our experiments, we compare the algorithm against our previous implementation and achieve an average speed up of 1500 on the ISCAS-85 benchmarks and fault tolerant modifications.**

I. Introduction

Technological improvements allow to continually decrease the size of transistors. This enables the construction of more powerful systems that use more transistors and reduces the required energy for each transistor. As a side effect of this development, circuits become more vulnerable against transient faults like *Single Event Transients* (SET). Especially critical systems need to be *robust* and must handle transient faults to prevent visible errors. To ensure that a system is robust, the robustness needs to be verified.

Most work that focuses on formal verification for robustness checking are limited to certain masking effects. The approaches [3, 4, 5, 9, 11] focus on logical masking while [8, 2] in addition consider timing effects. In [7] only electrical masking is considered. In [6], a complex mathematical model is used.

The contributed algorithm can decide if a circuit is robust against a given SET. Basic ideas similar to [1] are used for this check. We describe a front area of the circuit as *Boolean Satisfiability* (SAT) formula and utilize a solver to generate counterexamples that affect the back area. If such a counterexample exists, it is simulated on the whole circuit. When the counterexample affects the primary outputs, it disproves equivalence. Otherwise, we refine the original SAT formula to exclude the detected spurious counterexample.

This work has been supported by the University of Bremen's Graduate School SyDe, funded by the German Excellence Initiative, as well as a fellowship within the FITweltweit program of the German Academic Exchange Service (DAAD), and the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644905 (IMMORTAL).

Algorithm 1: ROBUST_CHECK

input : a circuit c and an SET s
output: a counterexample that disproves robustness of c under s or $\top$ if c is robust

1. Split c under s into c_f and c_b
2. $f :=$ CREATE_SAT(*front*, s)
3. **while** $\big((cex := \mathsf{SAT}(f)) \neq \mathsf{UNSAT}\big)$ **do**
4.   **if** SIMULATE(*cex*) **then return** *cex*
5.   Block GENERALIZE(*cex*) in f
6. **end**
7. **return** $\top$

Our algorithm

- separates the circuit into a front area and a back area and thus provides a simpler SAT formula for the SAT solver,
- uses both SAT solving and simulation where simulation is used to check if detected counterexamples are real or spurious,
- considers logical, timing, and electrical masking, and
- considers the used technology as well as process variation or other effects that affect the timing or variability of the gates.

II. Algorithm

The presented algorithm decides whether a given SET s on a circuit c could lead to erroneous output or the circuit is robust against the SET under all possible inputs and variability of the gates. The idea is sketched in Algorithm 1. For this decision, we separate c into a front area c_f and a back area c_b (1) as sketched in Figure 1. We separate the circuit such that all gates where the effects of the SET reconverge are within c_b. The area c_f is described as a SAT formula that includes logical, timing, and electrical masking as well as variability (2). If s cannot affect the signals that reach from c_f into c_b, c is robust. Otherwise, the SAT solver generates a counterexample *cex* (3). We simulate the assignments of *cex* on the whole circuit c to check if the assignment affects the primary outputs of c. In that

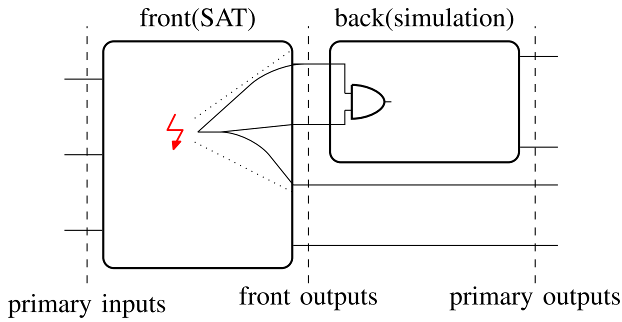

Fig. 1. Separation into front and back

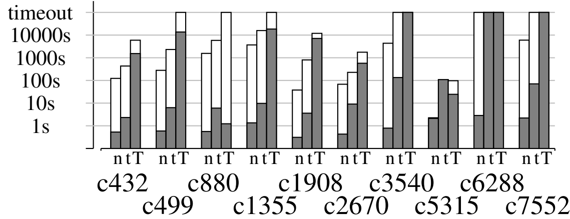

Fig. 2. Runtime experiments comparing our current algorithm (gray) against our previous implementation (white) on ISCAS-85 circuits, using the normal (n), TMR (t), and TTMR (T) version

case, *cex* is a real counterexample and disproves the robustness of *c* (4). If the primary outputs are not affected by *cex*, we generalize the assignment of *cex* and block it for future runs of the SAT solver (5). This process is repeated until either a real counterexample is found or the SAT formula is refined enough and does not generate any more spurious counterexamples and returns that the circuit is robust(6).

III. Experiments

In our experiments, we compared the runtime of our new algorithm to the runtime of [10]. In this implementation, we split the circuit such that all gates in which the SET reconverges are within the back area. We used the algorithms on the circuits from ISCAS-85 and modified each circuit into two fault tolerant versions. The modifications were *Triple Modular Redundancy* (TMR) and *Timed TMR* (TTMR) like described in [10].

For the injection location of the SET, we picked a random gate near the inputs for each circuit. Picking a gate near the inputs usually leads to a high number of gates that are affected by the SET. Picking a gate farther away from the inputs would lead to simpler SAT formulas as less gates are affected by the SET. This would also result in a smaller back area which leads to fewer spurious counterexamples and thus shorter runtimes. The results are shown in Figure 2. If an execution did not terminate after six hours, it was ended and resulted in a timeout.

The average speed up on the considered circuits compared to [10] is 1500. In the unmodified circuits, our new algorithm is very fast, as the SAT formula that describes the front is of low complexity and a real counterexample is quickly found since the original circuits do not provide fault tolerance. In fault tolerant versions, we can provide a significant speed up as the counterexamples that are generated by the SAT solver can be generalized. Especially the TMR versions profit from generalization as each voter at the outputs only requires the two inputs from the copied circuits that are not affected by the SET to be correct and does not need to consider any signals within the circuit that contains the SET.

While the exact speed up depends on the specific circuit, we have shown that the runtime decreases in most cases and only increases slightly in the remaining ones.

References

[1] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM*, pages 752–794, 2003.

[2] Mehdi Dehbashi and Görschwin Fey. SAT-based speed-path debugging using waveforms. In *IEEE European Test Symposium*, pages 1–6, 2014.

[3] Stefan Frehse, Görschwin Fey, Eli Arbel, Karen Yorav, and Rolf Drechsler. Complete and effective robustness checking by means of interpolation. In *Formal Methods in Computer-Aided Design*, pages 82–90, 2012.

[4] Jie Han, Hao Chen, Jinghang Liang, Peican Zhu, Zhixi Yang, and Fabrizio Lombardi. A stochastic computational approach for accurate and efficient reliability evaluation. *Computers, IEEE Transactions on*, pages 1336–1350, 2014.

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

[6] Natasa Miskov-Zivanov and Diana Marculescu. Multiple transient faults in combinational and sequential circuits: A systematic approach. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, pages 1614–1627, 2010.

[7] Martin Omaña, Giacinto Papasso, Daniele Rossi, and Cecilia Metra. A model for transient fault propagation in combinatorial logic. In *IEEE International On-Line Testing Symposium*, pages 111–115, 2003.

[8] Matthias Sauer, Alexander Czutro, Ilia Polian, and Bernd Becker. Small-delay-fault ATPG with waveform accuracy. In *Proceedings of the International Conference on Computer-Aided Design*, pages 30–36, 2012.

[9] Ashwin Seshia, Wenchao Li, and S Mitra. Verification-guided soft error resilience. In *Design, Automation Test in Europe Conference Exhibition*, pages 1–6, 2007.

[10] Niels Thole, Görschwin Fey, and Alberto Garcia-Ortiz. Conservatively analyzing transient faults. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, 2015.

[11] Michael Yoeli and Shlomo Rinon. Application of ternary algebra to the study of static hazards. *Journal of the ACM*, pages 84–97, 1964.