Improving Diagnosis Quality and Performances of a Formal Verification Tool for Electric Circuits at Transistor Level

2022-2023

Keywords: SAT Solving; SMT Solving; Logical Formulas; Integrated Circuits; User Requirements; Debugging; `OCaml`

Context

Aniah is a Start-up that offers tools for analyzing integrated circuits at an industrial scale[1]. Aniah has introduced algorithms that significantly pushes the boundaries of the size of analyzable circuits, from a few hundred thousand elements to several trillion. Aniah is starting a collaboration with the Laboratoire de l'Informatique du Parallélisme (LIP) and the Verimag laboratory to consolidate and generalize its approach by supplementing its practical results with a theoretical backbone. One of the objectives of this study is to explore the applicability of state-of-the-art model-checking techniques to the problem of circuit electric verification.

Model-checking [12] consists in exploring all the reachable states of a system, typically to check the unreachability of a set of error states. It is a well-established technique, and has successfully been applied both to software [1, 7, 6] and hardware [2, 4]. It is usually applied to check properties on the *behavior* of a system. For example, hardware model-checking usually considers boolean values (0 and 1, possibly extended with X and Z to model short-circuits and disconnected signals), but abstracts away the physical details (typically, voltage values are not modeled). Model-checking can be either enumerative (reachable states are explored one by one), or symbolic. Symbolic model-checking consists in representing a possibly very large set of states using a symbolic formula, that can be exponentially more efficient in terms of memory footprint. Common tools for symbolic model-checking are Binary Decision Diagrams (BDD) [5] and SAT-solvers [3] that allow manipulating boolean logical formulas. Satisfiability Modulo Theory (SMT) solvers extend SAT-solvers with non-boolean variables (e.g. rational numbers, integers, or other data structures). Among other work, these approaches have successfully been applied by the supervisors of this internship for Lustre program verification [9] and SystemC program verification [8].

Aniah proposed a graph based algorithm to detect electrical errors in a hierarchical design circuit. In this regard, the algorithm first assigns a finite set of values to the input variables of the circuit. Then, by analyzing the behavior of each net within the circuit, the algorithm detects electrical errors. One of the main issues in this analysis is the time and space complexity that is exponential with respect to the size of input variables. While the existing algorithm is usually fast enough in practice thanks to the good properties of the circuit topology, we are working on using symbolic model checking tools (BDD, SAT- and SMT-solvers) to speed up verification even more, as has been done in previous works [11, 10]. Z3 either proves the unsatisfiability of the formula (which mean that no electrical state can lead to an error), or provides a model for the formula. The model represents an electrical configuration leading to an error, and can be displayed to the user as a diagnosis. Unfortunately, this provides only a single electrical configuration, while a user may want to see all configurations leading to the same error to properly fix the root cause of the issue. We already experimented a *blocking* strategy to query Z3 repeatedly, adding the negation of the model as a clause to avoid getting the same model multiple times. However, this leads to an enumeration of model that can be very large, or even infinite, which is counter-productive as a debugging tool.

[1] `https://www.aniah.fr/`

Objectives of the internship

The objective of the internship is to provide tools to help the precise diagnosis and debugging of errors. Among the topics to study:

- Analyze the existing tool and blocking strategy
- Implement a new strategy. A first strategy, like "enumerate electrical configurations leading to different transistor states for all transistors on the path from the error to any supply" can be implemented, but further development should be done after discussions with Aniah's engineers to understand better the need of electrical engineers.
- Complement the model enumeration with further analysis, to understand the cause-effect relationships between errors. For example, a short circuit or a floating net can be the cause of other errors in other parts of the circuit. These other errors should not be investigated before their cause, because fixing the cause should also fix them. An analysis of the circuit giving the graph of cause-effect relationship between errors would be a key tool to help debugging.

Context of the Collaboration and Physical Location

The internship is proposed as part of the collaboration between LIP laboratory (Lyon), Verimag laboratory (Grenoble), and Aniah company (Grenoble). A post-doc (Bruno Ferres) and a CIFRE Ph.D (Oussama Oulkaid) student are already working on the subject. The student recruited for this internship will interact closely with them. A continuation on a Ph.D on a related subject is possible if the student is motivated.

The internship's goal is to use theoretical tools for a very practical concern, that is to provide the best possible tool for debugging. Depending on the student's motivation, the internship can focus more on theory, implementation, or requirement analysis with some discussions with the actual users.

The internship is proposed by LIP, Verimag and Aniah. The physical location of the internship is to be discussed with applicants. The student will visit other sites and meetings with all co-supervisors will be organized frequently.

- Laboratoire de l'Informatique du Parallélisme (LIP) – École Normale Supérieure de Lyon.
- Laboratoire Verimag, Grenoble.
- Aniah, Grenoble.

Required profile

The candidate should be familiar with algorithm design, understand the basics of Boole's algebra and logic. Good programming skills are required for the experimental validation of the approach. Since our tool is implemented in OCaml, prior knowledge of OCaml is appreciated, but the student can learn OCaml's basics during the internship. While the application domain is electronics, knowledge of electronics is not a strict prerequisite to perform this internship.

How to apply

Send an email to matthieu.moy@univ-lyon1.fr, Pascal.Raymond@univ-grenoble-alpes.fr, bruno.ferres@inria.fr and mehdi.khosravian@aniah.fr with your CV, a short text describing your motivation, and any document that can support your application.

Advisors

- Matthieu Moy, maître de conférences UCBL/LIP, `https://matthieu-moy.fr/`
- Pascal Raymond, chargé de recherche CNRS/Verimag, `http://www-verimag.imag.fr/~raymond/`
- Bruno Ferres, post-doc researcher at LIP, `https://perso.ens-lyon.fr/bruno.ferres`
- Mehdi Khosravian, Algorithm engineer in Aniah, `https://www.linkedin.com/in/mehdikhosravian/`
- Oussama Oulkaid, Ph.D student at LIP, Verimag and Aniah, `https://perso.ens-lyon.fr/oussama.oulkaid`

References

[1] Thomas Ball, Vladimir Levin, and Sriram K Rajamani. A decade of software model checking with slam. *Communications of the ACM*, 54(7):68–76, 2011.

[2] Ilan Beer, Shoham Ben-David, Cindy Eisner, and Avner Landver. Rulebase: An industry-oriented formal verification tool. In *33rd Design Automation Conference Proceedings, 1996*, pages 655–660. IEEE, 1996.

[3] Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. Symbolic model checking without bdds. In *International conference on tools and algorithms for the construction and analysis of systems*, pages 193–207. Springer, 1999.

[4] Aaron R Bradley. Sat-based model checking without unrolling. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 70–87. Springer, 2011.

[5] Jerry R Burch, Edmund M Clarke, Kenneth L McMillan, David L Dill, and Lain-Jinn Hwang. Symbolic model checking: 1020 states and beyond. *Information and computation*, 98(2):142–170, 1992.

[6] Patrice Godefroid. Software model checking: The verisoft approach. *Formal Methods in System Design*, 26(2):77–101, 2005.

[7] Daniel Kroening and Michael Tautschnig. Cbmc–c bounded model checker. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 389–391. Springer, 2014.

[8] Matthieu Moy, Florence Maraninchi, and Laurent Maillet-Contoz. Lussy: an open tool for the analysis of systems-on-a-chip at the transaction level. *Design Automation for Embedded Systems*, 10(2):73–104, 2005.

[9] Pascal Raymond. Synchronous program verification with lustre/lesar. *Modeling and Verification of Real-Time Systems*, page 7, 2008.

[10] S Rodriguez-Chavez, AA Palma-Rodriguez, E Tlelo-Cuautle, and SX-D Tan. Graph-based symbolic and symbolic sensitivity analysis of analog integrated circuits. In *Analog/RF and Mixed-Signal Circuit Systematic Design*, pages 101–122. Springer, 2013.

[11] Guoyong Shi. A survey on binary decision diagram approaches to symbolic analysis of analog integrated circuits. *Analog Integrated Circuits and Signal Processing*, 74(2):331–343, 2013.

[12] Wikipedia contributors. Model checking — Wikipedia, the free encyclopedia, 2021. [Online; accessed 21-September-2021].