Applying Symbolic Model-Checking Techniques to Circuit Electric Verification

Context

Aniah is a Start-up that offers tools for analyzing semiconductor manufacture circuits. 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]. 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 analysing 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 believe symbolic model checking tools (BDD and SAT-solvers) could speed up verification even more [11, 10].

Objectives of the internship

The objective of the internship is to explore the applicability of state-of-the-art tools like BDD and SAT-Solvers in the context of Aniah's electric verification. Among the challenges raised by this question:

- How to model the physical details of a circuit using tools initially designed to handle only boolean values? The voltage values of different power-supplies are currently modeled as a finite set of possible values in Aniah's tool, hence a straightforward way to handle the multi-valued logic is to use several boolean variables per signal in the circuit (either a $\log(n)$ encoding or a one-hot encoding).
- At the bottom of the hierarchy, the circuit is composed of a set of signals and transistors. The verification can be done by encoding the circuit into a boolean formula C, and the verification boils down to checking the satisfiability of $C \wedge E$ where E represents the set of error states. But the hierarchical analysis needs to be able to deal with circuits that

contains sub-circuits, and the sub-circuits needs to be handled properly. The abstraction of these sub-circuits needs to be modeled in the boolean formula representing the circuit.

The internship is a preliminary study. We expect the student to understand existing algorithm, propose new ones, and experiment them with a prototype implementation. This internship could lead to a CIFRE thesis funding (i.e. a Ph.D. co-supervised by a laboratory and a company), where we would push the experimental aspects further.

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. Basic knowledge of electronic circuits would be appreciated, but is not required.

How to apply

Send an email to matthieu.moy@univ-lyon1.fr, Pascal.Raymond@univ-grenoble-alpes.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/`
- Mehdi Khosravian, Algorithm engineer in Aniah, `https://www.linkedin.com/in/mehdikhosravian/`

Place

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 organised frequently.

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

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].