Verifying Next Generation Electronic Systems

(Invited Paper)

Rolf Drechsler Daniel Große

Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
{drechsle, grosse}@informatik.uni-bremen.de

***Abstract*—The application domains of electronic systems range from consumer devices to safety-critical systems. Of course, for systems of the latter areas a thorough verification is required. However, due to increasing complexity, verification is still the major bottleneck. Hence, new approaches are required.**

In this paper the state-of-the-art on verification is reported. Furthermore, recent developments are listed and finally the most pressing challenges for industry and academia are identified.

I. Introduction

Modern electronic systems are all around us: telecommunication, home automation, personal computers, automobiles, avionics and satellites, to name just a few. Since several decades a strong increase in complexity has been observed. With the advances in computation and communication new trends such as *Internet of Things* and *Industry 4.0* emerged as well as new opportunities in "old" domains, see for instance *Artificial Intelligence* (AI) with self-driving cars or self-learning systems. On top of all these developments *security* is of major importance.

Concerning verification, which is inevitable for safety-critical systems, it is well known that classical simulation-based approaches are not sufficient since they cannot prove the absence of errors and often miss corner-case bugs. In contrast, *formal verification* allows for proving the functional correctness in a mathematical sense. Formal verification has been around for 30 years now. However, formal verification is not always easy-to-use and complexity issues still limit its application. In summary, verification remains highly relevant, but new approaches are needed that improve the state-of-the-art by several orders of magnitude.

In this paper, first the state-of-the-art on hardware verification is reported. Then, recent developments are listed, formal verification at high-level of abstraction and self-verification is discussed, and finally the most pressing challenges for industry and academia are identified.

II. State of the Art Verification Techniques

In order to check whether complex systems are free of functional errors, verification methods are applied which check whether the system meets its specified requirements. Current industry practice applies the following verification techniques in the design flow:

- *Simulation:* Based on a model of the circuit, the inputs are explicitly assigned, propagated through the circuit, and the outputs are compared to the expected values.
- *Emulation:* Emulation realizes simulation directly in hardware thereby achieving an acceleration of some orders of magnitudes.
- *Formal verification:* Formal verification considers the problem mathematically and proves that the behavior of the circuit is correct.

From this we can conclude that the best possible quality can be achieved using formal verification. Therefore, we provide some more details in this direction. In particular, we focus on *Model Checking* (MC) also called *Property Checking* (PC). In PC a temporal property and a circuit are given and it is checked whether this property holds or fails for the circuit. If the property fails, a counter-example is generated. To capture this problem formally Boolean techniques are used. The most prominent are *Binary Decision Diagrams* (BDDs) and *Boolean Satisfiability* (SAT). In *Symbolic Model Checking* [1], [2] the system (and the already traversed state-space) are represented symbolically using BDDs. However, for larger circuits the generated BDDs become too large. Hence, as an alternative *Bounded Model Checking* (BMC) [3] has been proposed. Essentially, BMC uses SAT on the unrolled circuit plus logic for the property, and then checks whether the property holds for all behaviors up to a specified bound. BMC is very successful for industrial designs, see for instance [4]. With temporal induction BMC can be used for proving safety properties [5], [6]. Furthermore, based on the concept of incremental induction, which can be viewed as an over-approximation of the reachable state space, further improvements in terms of scalability can be achieved. Such techniques have been proposed recently as IC3/PDR in [7], [8].

Besides the just described fully automatic MC techniques, an alternative is *Theorem Proving*. Theorem proving makes uses of higher order logic and constructs a formal axiomatic proof of correctness. While theorem proving is more expressive, it is highly interactive, i.e. a lot of user interaction is required.

For more information on the basics on formal verification we refer the reader to [9].

The ever increasing design complexity dramatically widened the verification gap. Therefore different approaches and methodologies have been proposed to attack this problem. In the next section we briefly review two approaches.

III. Recent Verification Approaches

Raising the level of abstraction for developing circuits and systems has become a major approach for attacking the increasing complexity [10]. In the first subsection we consider formal verification at high-level of abstraction. After that, we consider an orthogonal approach to allow for verification even after shipping of the system.

A. Formal Verification at High-Level of Abstraction

SystemC-based virtual prototyping has become an established standard for modeling systems at high-level of abstraction. In this so called *Electronic System Level* (ESL) design flow, the SystemC-based *Virtual Prototype* (VP) serves as an executable specification for subsequent development steps in the design flow. Therefore, ensuring the correctness of high-level SystemC designs is crucial as undetected errors will propagate and become very costly. However, formal verification of SystemC is very challenging due to its object oriented nature and event-driven simulation semantics [11]. The challenge in developing a SystemC verifier is threefold:

1) First, it must obviously consider all possible inputs of the *Design-Under-Verification* (DUV).
2) Second, a typical high-level SystemC DUV consists of multiple asynchronous processes, whose different orders of execution (also referred to as schedules) can lead to different behaviors, these must also be considered to the full extent by the verifier.
3) Third, the verifier is required to deal with the full complexity of C++ to extract a suitable formal model.

The *Intermediate Verification Language* (IVL) for SystemC has been proposed to address the third challenge [12], by separating the development of a SystemC verifier into two components: a front-end to translate a DUV into IVL and a back-end to verify this IVL description. The IVL is compact and easily manageable but at the same time powerful enough to model the behavior of SystemC designs. As part of an ongoing effort to develop a fully automatic C++/SystemC verification frontend, the IVL is iteratively extended to bridge the VP modeling gap, e.g. recently, OOP support has been added [13]. Consequently, one can focus on addressing the first two challenges to increase the scalability and efficiency of the back-end in handling large state spaces.

Symbolic simulation [14], [15], [12], which is basically a combination of symbolic execution [16] with complete exploration of all possible process schedules, proved very effective in handling large state spaces. In [15], [12], symbolic simulation is further combined with *Partial Order Reduction* (POR) [17], [18] to significantly increase scalability by avoiding visiting redundant schedules. Recently, *Compiled Symbolic Simulation* (CSS) has been proposed to further boost scalability [19], [20]. CSS is a major enhancement that augments the DUV to integrate the symbolic execution engine and the POR based scheduler. Then, a standard C++ compiler is used to generate a native binary, whose execution performs exhaustive verification of the DUV. The whole state space of a DUV, which consists of all possible inputs and process schedules, can thus be exhaustively and efficiently explored.

Cyclic state space support for symbolic simulation, which allows to prove safety properties, has been provided in [21]. This stateful symbolic simulation approach for SystemC applies symbolic subsumption checking for efficient detection of revisited symbolic states.

B. Self-Verification

An alternative approach to deal with the verification problem is considered in this section. The general idea of *self-verification* is to realize a system which is capable of completing all the verification tasks that could not be finished during development time.

While the principle concept of self-verification may be realized in different fashions and scenarios, in general the considered system must be enabled to perform the following three *core verification tasks*:

1) *Monitoring*: observing the control and data flow which allows the system to keep track of the performed computations in terms of particular patterns or used scenarios. This allows for the recognition of what functionality is usually triggered and what outputs are generated by it.
2) *Verifying*: checking the correctness of parts of a system, the validity of properties, the completeness of coverage of a verification result, etc.
3) *Controlling*: deciding which verification task should be considered next based on an analysis scheme that takes previously obtained information into account, such as properties still left to be verified, frequently occurring patterns, or application scenarios of the system.

The concept of self-verification as well as some results based on this concept have been presented in [22], [23], [24], [25].

IV. Challenges

This section provides a list of challenges in the context of formal verification. The list is not complete in the sense that all difficulties are covered, but many pressing ones have been identified. By this, a better understanding of the current problems in verification of the next generation electronic systems becomes possible and directions for future research are suggested.

Complexity: Even though an end of Moores Law is coming close, the complexity of systems is steadily increasing. Hence, new abstraction levels and techniques are needed. In addition, if formal approaches reach their limits alternatives need to be devised which give better verification quality than classical simulation.
Further reading: [26], [27], [28], [29], [30], [31]

Formal verification at ESL: Formal verification at the *Electronic System Level* (ESL) is inevitable since virtual prototyping has become industrial practice today. Despite the substantial progress on formal VP verification as discussed in Section III-A, the existing techniques

still need further improvements to scale to full VP systems.
Further reading: [32], [14], [33], [12], [21], [34], [13], [19], [20], [35]

Coverage metrics: Strong coverage metrics for formal verification at the *Register Transfer Level* (RTL) have been proposed. However, coverage at higher levels of abstraction is still very challenging.
Further reading: [36], [37], [38], [39], [40], [41], [42]

Arithmetic: BDD and SAT/SMT techniques suffer from limitations when applied to complex arithmetic, e.g. multipliers. Recently, algebraic methods based on Gröbner bases and Ideal membership testing received renewed interest, since they have been successfully applied to different circuits including very complex arithmetic.
Further reading: [43], [44], [45], [46], [47]

Analog Mixed Signal: To separate the system in analog and digital parts is not sufficient anymore. New integrated languages as well as verification solutions are necessary in particular due to increasing sensor information available in *Internet-of-Things* (IoT) or *Cyber-Physical Systems* (CPS).
Further reading: [48], [49], [50], [51], [52]

Learning systems, especially self-adapting and -learning: Driven by the enormous advances in computing power, learning for instance in the form of neural networks, machine learning etc. can now be integrated into these systems. While new intelligent systems become reality now, the verification of such systems is much more challenging compared to classical systems.
Further reading: [53], [54]

Security: Due to the increasing amount of sensitive information and personal data being stored in embedded devices, increasing connectivity to other systems as well as the security-critical functions they perform, security has become a major requirement. Along with proofs of correctness, security should not become an afterthought and new respective security verification techniques for hardware and software have to be developed.
Further reading: [55], [56], [57], [58], [59]

V. Conclusion

In this paper formal verification of the next generation electronic systems has been addressed. After briefly reporting the state-of-the-art from a circuit perspective, two verification approaches have been reviewed, i.e. formal verification at a high-level of abstraction and self-verification.

Finally, a list of major challenges has been provided to stimulate research.

Acknowledgments

The work has (partially) been supported by the German Research Foundation (DFG) within the Reinhart Koselleck project DR 287/23-1, by the German Federal Ministry of Education and Research (BMBF) within the project EffektiV under contract no. 01IS13022E, within the BMBF project SELFIE under grant no. 01IW16001, within the BMBF project CONFIRM under contract no. 16ES0565, within the BMBF project CONVERS under contract no. 16ES0656, the DFG, as part of Collaborative Research Center (Sonderforschungsbereich) 1320 EASE – Everyday Activity Science and Engineering, University of Bremen (http://www.ease-crc.org/; the research was conducted in subproject P04) and by the University of Bremen's graduate school SyDe, funded by the German Excellence Initiative.

References

[1] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 10^{20} states and beyond," *Information and Computation*, vol. 98(2), pp. 142–170, 1992.
[2] K. L. McMillan, *Symbolic Model Checking*. Kluwer Academic Publisher, 1993.
[3] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," in *Tools and Algorithms for the Construction and Analysis of Systems*, 1999, pp. 193–207.
[4] M. Ganai and A. Gupta, *SAT-Based Scalable Formal Verification Solutions (Series on Integrated Circuits and Systems)*. Springer, 2007.
[5] M. Sheeran, S. Singh, and G. Stålmarck, "Checking safety properties using induction and a SAT-solver," in *Int'l Conf. on Formal Methods in CAD*, 2000, pp. 108–125.
[6] P. Bjesse and K. Claessen, "SAT-based verification without state space traversal," in *Int'l Conf. on Formal Methods in CAD*, 2000, pp. 372–389.
[7] A. R. Bradley, "SAT-based model checking without unrolling," in *International Conference on Verification, Model Checking, and Abstract Interpretation*, 2011, pp. 70–87.
[8] N. Een, A. Mishchenko, and R. Brayton, "Efficient implementation of property directed reachability," in *Int'l Conf. on Formal Methods in CAD*, 2011, pp. 125–134.
[9] T. Kropf, *Introduction to Formal Hardware Verification*. Springer, 1999.
[10] R. Drechsler, Ed., *Formal System Verification*. Springer, 2017.
[11] M. Y. Vardi, "Formal techniques for SystemC verification," in *Design Automation Conf.*, 2007, pp. 188–192.
[12] H. M. Le, D. Große, V. Herdt, and R. Drechsler, "Verifying SystemC using an intermediate verification language and symbolic simulation," in *Design Automation Conf.*, 2013, pp. 116:1–116:6.
[13] H. M. Le, V. Herdt, D. Große, and R. Drechsler, "Towards formal verification of real-world SystemC TLM peripheral models – a case study," in *Design, Automation and Test in Europe*, 2016, pp. 1160–1163.
[14] C.-N. Chou, Y.-S. Ho, C. Hsieh, and C.-Y. Huang, "Symbolic model checking on SystemC designs," in *Design Automation Conf.*, 2012, pp. 327–333.
[15] C.-N. Chou, C.-K. Chu, and C.-Y. R. Huang, "Conquering the scheduling alternative explosion problem of SystemC symbolic simulation," in *International Conference on Computer-Aided Design*, 2013, pp. 685–690.
[16] J. C. King, "Symbolic execution and program testing," *Commun. ACM*, vol. 19, no. 7, pp. 385–394, Jul. 1976.
[17] P. Godefroid, *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*. Springer, 1996.
[18] C. Flanagan and P. Godefroid, "Dynamic partial-order reduction for model checking software," in *Symposium on Principles of Programming Languages*, 2005, pp. 110–121.
[19] V. Herdt, H. M. Le, D. Große, and R. Drechsler, "ParCoSS: efficient parallelized compiled symbolic simulation," in *Computer Aided Verification*, 2016, pp. 177–183.
[20] ——, "Compiled symbolic simulation for SystemC," in *International Conference on Computer-Aided Design*, 2016, pp. 52:1–52:8.

[21] V. Herdt, H. M. Le, and R. Drechsler, "Verifying SystemC using stateful symbolic simulation," in *Design Automation Conf.*, 2015, pp. 49:1–49:6.

[22] R. Drechsler, H. M. Le, and M. Soeken, "Self-verification as the key technology for next generation electronic systems," in *Symposium on Integrated Circuits and System Design*, 2014, invited Talk.

[23] R. Drechsler, M. Fränzle, and R. Wille, "Envisioning self-verification of electronic systems," in *Int'l Symp. on Reconfigurable Communication-centric Systems-on-Chip*, 2015.

[24] C. Lüth, M. Ring, and R. Drechsler, "Towards a methodology for self-verification," in *International Conference on Reliability, Infocom Technologies and Optimization*, 2017.

[25] F. Bornebusch, R. Wille, and R. Drechsler, "Towards lightweight satisfiability solvers for self-verification," in *International Symposium on Embedded Computing and System Design*, 2017.

[26] M. Soeken and R. Drechsler, *Formal Specification Level - Concepts, Methods, and Algorithms*. Springer, 2015.

[27] D. Große, H. M. Le, M. Hassan, and R. Drechsler, "Guided lightweight software test qualification for IP integration using virtual prototypes," in *Int'l Conf. on Comp. Design*, 2016, pp. 606–613.

[28] M. Hassan, V. Herdt, H. M. Le, M. Chen, D. Große, and R. Drechsler, "Data flow testing for virtual prototypes," in *Design, Automation and Test in Europe*, 2017, pp. 380–385.

[29] P. G. de Aledo, N. Przigoda, R. Wille, R. Drechsler, and P. S. Espeso, "Towards a verification flow across abstraction levels verifying implementations against their formal specification," *IEEE Transactions on Computer Aided Design of Circuits and Systems*, vol. 36, no. 3, pp. 475–488, 2017.

[30] F. Haedicke, H. M. Le, D. Große, and R. Drechsler, "CRAVE: An advanced constrained random verification environment for SystemC," in *International Symposium on System-on-Chip*, 2012, pp. 1–7.

[31] M. F. S. Oliveira, C. Kuznik, W. Müller, F. Haedicke, H. M. Le, D. Große, R. Drechsler, W. Ecker, and V. Esen, "The system verification methodology for advanced TLM verification," in *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2012, pp. 313–322.

[32] D. Große, H. M. Le, and R. Drechsler, "Proving transaction and system-level properties of untimed SystemC TLM designs," in *ACM & IEEE International Conference on Formal Methods and Models for Codesign*, 2010, pp. 113–122.

[33] A. Cimatti, I. Narasamdya, and M. Roveri, "Software model checking SystemC," *IEEE Transactions on Computer Aided Design of Circuits and Systems*, vol. 32, no. 5, pp. 774–787, 2013.

[34] V. Herdt, H. M. Le, D. Große, and R. Drechsler, "Boosting sequentialization-based verification of multi-threaded C programs via symbolic pruning of redundant schedules," in *Automated Technology for Verification and Analysis*, 2015, pp. 228–233.

[35] ——, "Towards fully automated TLM-to-RTL property refinement," in *Design, Automation and Test in Europe*, 2018.

[36] K. Claessen, "A coverage analysis for safety property lists," in *Int'l Conf. on Formal Methods in CAD*, 2007, pp. 139–145.

[37] D. Große, U. Kühne, and R. Drechsler, "Estimating functional coverage in bounded model checking," in *Design, Automation and Test in Europe*, 2007, pp. 1176–1181.

[38] J. Bormann, S. Beyer, A. Maggiore, M. Siegel, S. Skalberg, T. Blackmore, and F. Bruno, "Complete formal verification of Tricore2 and other processors," in *Design and Verification Conference*, 2007.

[39] D. Große, U. Kühne, and R. Drechsler, "Analyzing functional coverage in bounded model checking," *IEEE Transactions on Computer Aided Design of Circuits and Systems*, vol. 27, no. 7, pp. 1305–1314, 2008.

[40] H. M. Le, D. Große, and R. Drechsler, "Towards analyzing functional coverage in SystemC TLM property checking," in *IEEE International High Level Design Validation and Test Workshop*, 2010, pp. 67–74.

[41] F. Haedicke, D. Große, and R. Drechsler, "A guiding coverage metric for formal verification," in *Design, Automation and Test in Europe*, 2012, pp. 617–622.

[42] R. Drechsler, M. Diepenbeck, D. Große, U. Kühne, H. M. Le, J. Seiter, M. Soeken, and R. Wille, "Completeness-driven development," in *International Conference on Graph Transformation*, 2012, pp. 38–50.

[43] J. Lv, P. Kalla, and F. Enescu, "Efficient Gröbner basis reductions for formal verification of Galois field arithmetic circuits," *IEEE Transactions on Computer Aided Design of Circuits and Systems*, vol. 32, no. 9, pp. 1409–1420, Sept 2013.

[44] F. Farahmandi and B. Alizadeh, "Gröbner basis based formal verification of large arithmetic circuits using gaussian elimination and cone-based polynomial extraction," *Microprocessors and Microsystems*, vol. 39, no. 2, pp. 83–96, 2015.

[45] A. Sayed-Ahmed, D. Große, U. Kühne, M. Soeken, and R. Drechsler, "Formal verification of integer multipliers by combining Gröbner basis with logic reduction," in *Design, Automation and Test in Europe*, 2016, pp. 1048–1053, (Best paper candidate).

[46] A. Sayed-Ahmed, D. Große, M. Soeken, and R. Drechsler, "Equivalence checking using Gröbner bases," in *Int'l Conf. on Formal Methods in CAD*, 2016, pp. 169–176.

[47] D. Ritirc, A. Biere, and M. Kauers, "Column-wise verification of multipliers using computer algebra," in *Int'l Conf. on Formal Methods in CAD*, 2017.

[48] S. Steinhorst and L. Hedrich, "Trajectory-directed discrete state space modeling for formal verification of nonlinear analog circuits," in *International Conference on Computer-Aided Design*, 2012, pp. 202–209.

[49] C. Radojicic, C. Grimm, F. Schupfer, and M. Rathmair, "Verification of mixed-signal systems with affine arithmetic assertions," *VLSI Design*, vol. 2013, pp. 239 064:1–239 064:14, 2013.

[50] M. Barnasconi, M. Dietrich, K. Einwich, T. Vörtler, J. Chaput, M. Louërat, F. Pêcheux, Z. Wang, P. Cuenot, I. Neumann, T. Nguyen, R. Lucas, and E. Vaumorin, "UVM-SystemC-AMS framework for system-level verification and validation of automotive use cases," *IEEE Design & Test*, vol. 32, no. 6, pp. 76–86, 2015.

[51] *IEEE SystemC AMS standard*, IEEE Std. 1666.1-2016, 2016.

[52] M. Hassan, D. Große, H. M. Le, T. Vörtler, K. Einwich, and R. Drechsler, "Testbench qualification for SystemC-AMS timed data flow models," in *Design, Automation and Test in Europe*, 2018.

[53] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Computer Aided Verification*, 2017, pp. 3–29.

[54] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Computer Aided Verification*, 2017, pp. 97–117.

[55] W. Hu, J. Oberg, A. Irturk, M. Tiwari, T. Sherwood, D. Mu, and R. Kastner, "Theoretical fundamentals of gate level information flow tracking," *IEEE Transactions on Computer Aided Design of Circuits and Systems*, vol. 30, no. 8, pp. 1128–1140, Aug 2011.

[56] P. Schaumont, M. O'Neill, and T. Güneysu, "Introduction for embedded platforms for cryptography in the coming decade," *ACM Trans. Embedded Comput. Syst.*, vol. 14, no. 3, pp. 40:1–40:3, 2015.

[57] P. Subramanyan, S. Malik, H. Khattri, A. Maiti, and J. Fung, "Verifying information flow properties of firmware using symbolic execution," in *Design, Automation and Test in Europe*, 2016, pp. 337–342.

[58] A. Ardeshiricham, W. Hu, J. Marxen, and R. Kastner, "Register transfer level information flow tracking for provably secure hardware design," in *Design, Automation and Test in Europe*, 2017, pp. 1691–1696.

[59] M. Hassan, V. Herdt, H. M. Le, D. Große, and R. Drechsler, "Early SoC security validation by VP-based static information flow analysis," in *International Conference on Computer-Aided Design*, 2017, pp. 400–407.