Fuzz-Testing of SpinalHDL Designs *

Katharina Ruep, Daniel Große
Institute for Complex Systems, Johannes Kepler University Linz, Austria
katharina.ruep@jku.at, daniel.grosse@jku.at

Abstract

In this extended abstract, we summarize our work from [11], where we proposed to bring *Coverage-guided Fuzzing* (CGF) to the SpinalHDL design flow. We demonstrated for a wide range of SpinalHDL designs the effectiveness of our tool in comparison to *Constrained Random Verification* (CRV). In addition, we present recent developments going beyond [11].

1 Extended Abstract

Confronted with the end of Moore's law and Dennard scaling, a new golden age for computer architecture has started [5]. However, to make this a reality, domain-specific architectures and designs are essential and, hence, the design productivity needs an enormous boost. Therefore, alternatives to *Register Transfer Level* (RTL) design, which is based on the classical *Hardware Description Languages* (HDLs) Verilog or VHDL, moved in the focus of research. A very promising alternative is SpinalHDL [1]. SpinalHDL has been realized as an embedded *Domain Specific Language* (DSL) in Scala and allows to describe RTL designs by using object-oriented and functional programming. Moreover, the meta-programming features of Scala can be used for parametrization and hardware code generation. In the final step of the SpinalHDL flow, a Verilog design is generated for the design. However, while productivity gains by a factor of 3 or more have been reported using meta-modeling and code generation [2], verification must keep up, since otherwise the verification gap is widening even faster.

In this work, we focus on simulation-based verification of SpinalHDL designs. To define the stimuli for simulation, the verification engineer can either create directed tests or make use of more advanced techniques, like *Constrained Random Verification* (CRV) [13, 4]. The latter has been further improved by integrating coverage feedback from simulations. This, however, requires adjustments of weights and constraints, design-specific Bayesian networks [3], or utilization of data mining techniques [7], each with high manual effort.

In software testing *Fuzzing* is a well-established technique [9]. Fuzzing is a process where the *Program Under Test* (PUT) is executed repeatedly with random-generated inputs to find software bugs and security vulnerabilities. State-of-the-art *Coverage-Guided Fuzzing* (CGF) aims to maximize code coverage of the PUT. The core principle of CGF is as follows: A *corpus* is created which contains at least one initial test case. Then, the CGF *feedback loop* starts, which consists of two main steps: (1) From the corpus a test case is taken, mutated to create a new one and then fed to the PUT. (2) If the overall coverage increases during PUT execution, the new test case is added to the corpus, otherwise it is discarded. With this prioritization of interesting (coverage-increasing) test cases, CGF boosts the efficiency of basic fuzzing significantly.

However, for hardware, only very few fuzzing approaches have been presented so far (e.g. [10, 14],[6],[8],[12]). None of these approaches target SpinalHDL and they all require a lot of user interaction to setup fuzzing. Moreover, a common challenge for the hardware domain is the additional ingredient of a *harness*. The harness has to (1) translate a test case from the corpus, typically a byte stream, to the input signals of the *Device Under Test* (DUT) and (2) support the sequential behavior of hardware, i.e. input values over time.

In this extended abstract, we summarize our work from [11], where we presented SPINALFUZZ, a CGF approach for SpinalHDL designs to overcome these mentioned challenges. During the development of SPINALFUZZ our primary objective was to make fuzzing of a SpinalHDL DUT as easy as possible and, hence, automate as much as possible. For the integration of fuzzing into the SpinalHDL design flow, we (a) leverage the SpinalHDL/Scala language features for various generation tasks and (b) benefit from existing software fuzzers. In several experiments, we were able to demonstrate the effectiveness of SPINALFUZZ in comparison to CRV on a wide range of SpinalHDL designs. In all cases SPINALFUZZ outperformed CRV and reached a coverage greater than 90%. In addition, we present recent developments going beyond [11].

2 Literature

[1] SpinalHDL. `https://github.com/SpinalHDL`.

[2] W. Ecker and J. Schreiner. Metamodeling and code generation. In S. Ha and J. Teich, editors, *Handbook of Hardware/Software Codesign*, pages 1–41. Springer, 2017.

[3] S. Fine and A. Ziv. Coverage directed test generation for

*This work has partially been supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria.

functional verification using bayesian networks. In *Design Automation Conf.*, pages 286–291, 2003.

[4] 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*, pages 1–7, 2012.

[5] J. L. Hennessy and D. A. Patterson. A new golden age for computer architecture. *Comm. of the ACM*, 62(2):48–60, 2019.

[6] V. Herdt, D. Große, H. M. Le, and R. Drechsler. Verifying instruction set simulators using coverage-guided fuzzing. In *Design, Automation and Test in Europe*, pages 360–365, 2019.

[7] C. Ioannides and K. I. Eder. Coverage-directed test generation automated by machine learning – a review. *ACM Trans. on Design Automation of Electronic Systems*, 17(1): 7:1–7:21, 2012.

[8] K. Laeufer, J. Koenig, D. Kim, J. Bachrach, and K. Sen. RFUZZ: Coverage-directed fuzz testing of RTL on FPGAs. In *International Conference on Computer-Aided Design*, pages 1–8, 2018.

[9] V. J. Manès et al. The art, science, and engineering of fuzzing: A survey. *IEEE Transactions on Software Engineering*, 47(11):2312–2331, 2021.

[10] M. Muench et al. What you corrupt is not what you crash: Challenges in fuzzing embedded devices. In *Network and Distributed System Security Symposium*, 2018.

[11] K. Ruep and D. Große. SpinalFuzz: Coverage-guided fuzzing for SpinalHDL designs. In *European Test Symposium*, pages 1–4, 2022.

[12] T. Trippel, K. G. Shin, A. Chernyakhovsky, G. Kelly, D. Rizzo, and M. Hicks. Fuzzing hardware like software, 2021. arXiv:2102.02308.

[13] J. Yuan, C. Pixley, and A. Aziz. *Constraint-based Verification*. Springer, 2006.

[14] Y. Zheng, A. Davanian, H. Yin, C. Song, H. Zhu, and L. Sun. FIRM-AFL: High-throughput greybox fuzzing of IoT firmware via augmented process emulation. In *USENIX Security Symposium*, 2019.