RISC-V Processor Verification with Coverage-guided Aging*

Niklas Bruns[1], Vladimir Herdt[1,2], Eyck Jentzsch[3], Rolf Drechsler[1,2]
[1]Institute of Computer Science, University of Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
[3]MINRES Technologies GmbH, 85579 Neubiberg, Germany
nbruns@uni-bremen.de, vherdt@uni-bremen.de, eyck@minres.com, drechsler@uni-bremen.de

Abstract

In this extended abstract we present an efficient approach for processor verification at the *Register-Transfer Level* (RTL), using a cross-level setting with an *Instruction Set Simulator* (ISS) as a reference model. We leverage a custom instruction stream generator tailored for RISC-V, which produces one endless instruction stream at runtime. Moreover, we employ a coverage-guided aging concept which ensures a more uniform distribution of the generated instructions by tracking and updating coverage information in the ISS and dynamically providing feedback to the instruction stream generator. Our case study with an industrial pipelined 32 bit RISC-V processor demonstrates the effectiveness of our approach.

1 Extended Abstract

RISC-V [10, 11] is a free and open-source ISA that enables a royalty-free processor design and implementation. It is designed in a very modular way with optional standard instruction set extensions around a mandatory base integer instruction set and the ability to integrate additional custom instruction sets to build highly application-specific processors. These properties made RISC-V very popular in industry and academia. From the verification perspective, however, the extensive modularity adds additional complexity. Besides the modern features provided by RISC-V and any micro-architectural specific optimizations of the processor, such as pipelining and branch prediction, the verification tools also need to be able to deal with the large configuration space offered by RISC-V.

Recently, approaches specifically tailored for RISC-V verification have emerged. The baseline is provided by the official RISC-V unit and compliance tests [2, 1], which are directed test suites. More comprehensive simulation-based testing is enabled by approaches that rely on continuous assembly test generation. Different approaches have been devised that generate tests by integrating randomized instruction templates [3], leverage constraint-based specification mechanism to define coverage requirements [6], utilize coverage-guided fuzzing techniques to generate tests [8, 5], and use symbolic execution techniques to find specific inputs [9]. A particularly promising approach in this context is pursued by Google's open-source RISC-V *Design Verification* (DV) framework. It leverages a method based on co-simulation, i.e. it employs an *Instruction Set Simulator* (ISS) as a functional reference model for the RTL processor under test. As a backbone it applies constraint-based specification techniques in SystemVerilog to generate RISC-V assembly tests one after another. Different RISC-V instruction sets are supported by selecting and combining the respective constraint-based specifications. Execution results between the ISS and RTL processor core are compared through execution log files. While this feature set makes RISC-V DV very powerful in general, it also has some major weaknesses. In order to keep the framework generic, the generated tests use a restricted instruction set to avoid problems with infinite loops and platform-dependent memory access operations. Moreover, by generating tests one by one, only comparatively short instruction sequences are considered, and the state of the processor under test is regularly reset for each new test execution. Furthermore, the co-simulation has an inherent performance overhead due to the extensive filesystem communication, since each RISC-V assembly test needs to be compiled, loaded onto the respective simulator, and produce a log file for comparison. Finally, the test generator is not designed to be dynamically guided by coverage information obtained from the test execution progress. Many of these issues have been addressed by a recent academic work [7]. It generates endless instruction streams and integrates the ISS with the RTL core in a very efficient co-simulation compiled into a single binary with in-memory communication. The setup allows to generate instructions without any restrictions, i.e., arbitrary combinations of load/store and *Control and Status Registers* (CSRs)[1] instructions, as well as infinite loops, are supported, which enables a very comprehensive test approach. However, the approach is still limited as it does not collect or employ runtime coverage information to assess and guide the test generation process. Instead, the instruction stream generators are based on a simple randomized test strategy which makes it very difficult to continuously achieve a broad and deep test coverage in endless instruction streams.

In this extended abstract, we summarize our work from [4], where we proposed a novel approach for cross-level ver-

*This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within the project Scale4Edge under contract no. 16ME0127 and no. 16ME0135, and within the project VerSys under contract no. 01IW19001.

[1]In the CSRs, the processor stores additional instruction results to enable sophisticated hardware/software interactions.

ification that conceptually builds upon the previous academic work [7] and addresses the aforementioned limitations. The foundation is a randomized coverage-guided instruction stream generator that produces an endless and unrestricted instruction stream that evolves dynamically at runtime based on observed coverage information. We also leverage an ISS as a reference model in a tight co-simulation setting. Coverage information is continuously updated based on the execution state of the ISS and we employ the novel concept of coverage-guided aging to smooth out the coverage distribution of the randomized instruction stream over time. Our experiments with the 32-bit pipelined RISC-V core of the MINRES *The Good Core* (TGC) series demonstrate the effectiveness of our approach in achieving a much more regular coverage distribution of the randomized instruction stream via coverage-guided aging. For more details please refer to [4].

2 References

[1] RISC-V compliance task group. `https://github.com/riscv/riscv-compliance`.

[2] RISC-V ISA tests. `https://github.com/riscv/riscv-tests`.

[3] RISC-V torture test generator. `https://github.com/ucb-bar/riscv-torture`.

[4] N. Bruns, V. Herdt, E. Jentzsch, and R. Drechsler. Cross-level processor verification via endless randomized instruction stream generation with coverage-guided aging. In *DATE*, 2022. accepted.

[5] V. Herdt, D. Große, and R. Drechsler. Closing the RISC-V compliance gap: Looking from the negative testing side. In *DAC*, 2020.

[6] V. Herdt, D. Große, and R. Drechsler. Towards specification and testing of RISC-V ISA compliance. In *DATE*, pages 995–998, 2020.

[7] V. Herdt, D. Große, E. Jentzsch, and R. Drechsler. Efficient cross-level testing for processor verification: A RISC-V case-study. In *FDL*, 2020.

[8] V. Herdt, D. Große, H. M. Le, and R. Drechsler. Verifying instruction set simulators using coverage-guided fuzzing. In *DATE*, pages 360–365, 2019.

[9] V. Herdt, S. Tempel, D. Große, and R. Drechsler. Mutation-based compliance testing for RISC-V. In *ASP-DAC*, 2021.

[10] A. Waterman and K. Asanović, editors. *The RISC-V Instruction Set Manual; Volume I: Unprivileged ISA*. 2019.

[11] A. Waterman and K. Asanović, editors. *The RISC-V Instruction Set Manual; Volume II: Privileged Architecture*. 2019.