RISC-V Opt-VP: An Application Analysis Platform Using Bounded Execution Trees

Jan Zielasko\textsuperscript{1,2}, Rune Krauss\textsuperscript{1}, Marcel Merten\textsuperscript{1}, Rolf Drechsler\textsuperscript{1,2}\textsuperscript{*}

\textsuperscript{1}Institute of Computer Science, University of Bremen, Germany
\textsuperscript{2}Cyber-Physical Systems, DFKI GmbH, Germany

Abstract

Tailoring hardware to applications significantly increases their performance, which is required to meet the rising demand for resource-limited devices in the area of embedded systems. While RISC-V facilitates application-specific solutions due to its extensibility, Virtual Prototypes (VPs) enable early software development before the actual hardware is built shortening the time-to-market. Although the RISC-V VP ecosystem already offers many useful tools to aid development there is still room for improvement, especially in analyzing applications for hardware optimization. To address the aforementioned issue and expand the mentioned ecosystem with a tool, this work presents RISC-V Opt-VP, which generates bounded execution trees in order to analyze applications. An embedded application analysis case study illustrates that promising instruction sequences are found which can also be merged to further improve their execution coverage, enabling efficient hardware designs.

Introduction

With the ever-increasing demand for high-performance applications in areas such as embedded systems, it is becoming increasingly important to optimize hardware in order to meet time-to-market constraints [1].

RISC-V is an instruction set architecture that is characterized by its modularity and extensibility [2]. It facilitates efficient and application-specific solutions, which is particularly ideal for resource-limited devices in areas such as embedded systems and IoT.

Virtual Prototypes (VPs) such as RISC-V VP\textsuperscript{1} [3] enable early software development and testing by providing an executable hardware platform implemented using SystemC transaction-level modeling [4].

The RISC-V VP ecosystem offers some efficient tools to aid development. These include a co-simulator [5] and 3D visualizations of symbolic execution for debugging purposes [6]. However, there is still a need to analyze applications for hardware optimization.

To address this issue and expand tool diversity, in this extended abstract we propose RISC-V Opt-VP, which constructs bounded execution trees based on applications for tailoring hardware. Using an embedded application analysis case study, we illustrate that promising instruction sequences are found which can be merged to increase execution coverage, enabling efficient hardware designs. To stimulate further research, our tool is provided as open-source software\textsuperscript{2} [7, 8].

\*Corresponding author: Jan.Zielasko@DFKI.de. This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within projects Scale4Edge under grant no. 16MEO127, ECXL under grant no. 01IW22002 and VE-HEP under grant no. 16KIS1342.

\textsuperscript{1}https://github.com/agra-uni-bremen/riscv-vp
\textsuperscript{2}https://github.com/agra-uni-bremen/opt-vp
In this extended abstract we presented a VP-driven tool called RISC-V Opt-VP to analyze applications for hardware optimization. Using an embedded application analysis case study, we found that the generation of bounded execution trees can be used to identify promising instruction sequences that cover a large fraction of the total execution. Additionally, merging instructions more than doubles the achieved coverage, which together enables efficient hardware designs. Based on these high-level results, hardware can be tailored to applications by accelerating analyzed sequences that require a high execution time. By identifying similarities in different sequences and merging them, the design of a single hardware accelerator or new instruction that covers a large fraction of the total execution is possible, while the performance loss is negligible compared to building multiple accelerators, thus saving expensive hardware costs.

Future work will be directed towards further investigation of hardware design and construction of actual hardware. On the one hand, further metrics used by our scoring function, such as input and output of instruction sequences, are to be studied in order to fine-tune the identified sequences to a specific optimization. On the other hand, it is planned to design coarse-grained reconfigurable architectures to accurately measure possible hardware acceleration for the covered sequences.

### References


