A Flexible and Portable Performance Evaluation Framework for Instruction Set Simulations

Conrad Foik[1], Karan Kedia[1], Robert Kunzelmann[1],
Daniel Müller-Gritschneder[2] and Ulf Schlichtmann[1]

[1]Technical University of Munich, [2]TU Wien
conrad.foik@tum.de, daniel.mueller-gritschneder@tuwien.ac.at

Abstract

Simulation-based design space exploration plays a crucial part in the development of efficient processors for modern embedded systems. Common tools in this context are performance simulators, but these usually inflexibly target a single processor variant and cannot easily be combined with established simulation environments. This paper presents a performance simulation framework that combines high accuracy and simulation speed with high flexibility and quick integration into existing simulation environments. It consists of a trace-based performance estimator, which can be adapted to new processor variants through code generation based on a simple structural description of the microarchitecture. Applying our approach to the state-of-the-art CVA6 RISC-V application class processor shows an average relative error of 3.88% and an average simulation speed of 15 million instructions per second (MIPS) over the Embench benchmark suite.

Introduction

Modern embedded systems increasingly rely on highly workload-tailored designs to meet ever stricter performance and power requirements. Prominent examples of such designs are application-specific instruction set processors (ASIPs). However, to identify the optimal solutions for such a processor from numerous design options, designers rely on early simulation-based software profiling and design space exploration (DSE), resulting in a new demand for fast and accurate simulators.

While the processor's register transfer level (RTL) description theoretically allows for cycle-accurate simulations, this description is typically too late available in the design process for an effective DSE. In addition, RTL simulations are very slow, making them unsuitable for DSE, which requires running numerous simulations. A more suitable approach is the use of so-called instruction set simulators (ISSs), which model the processor's behavior on the instruction set architecture (ISA) level. This increased level of abstraction enables an ISS to run considerably faster than RTL simulations while still correctly modeling the processor's functional behavior. However, since an ISS does not consider microarchitectural details, it cannot reliably estimate the processor's performance, which is a crucial metric for a convincing DSE. To address this limitation, several performance simulators have been proposed in the literature. These simulators typically combine the ISS approach with non-functional timing models of the target processor's microarchitecture to achieve a suitable trade-off between accurate performance estimates and simulation speeds.

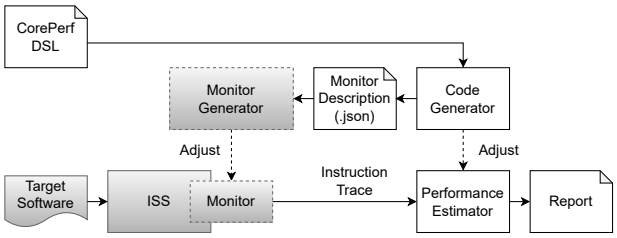

Figure 1: *Performance evaluation framework (External components in gray)*

However, although a flexible adaptation to new processor variants is essential for an effective DSE, most performance simulators either target a single processor variant or compromise speed and accuracy. In addition, current approaches are typically simulator-specific, thus prohibiting the extension of established simulation environments. In this paper, we, therefore, present an accurate and fast performance simulation framework, which offers both flexible adaptation to new microarchitecture variants and quick integration into existing simulation environments.

Simulation Framework

Fig. 1 presents our proposed simulation framework. It comprises two main components: a performance estimator and a code generator [1].

The *performance estimator* approximates the processor's performance by considering the instruction trace provided by an arbitrary ISS. Its estimation method is based on as-soon-as-possible (ASAP) scheduling of instructions on microarchitecture resources and is

This work was supported in part by the German Federal Ministry of Education and Research (BMBF) and ITEA within the project GenerIoT under contract no. 01IS22084G

Table 1: *Experimental results for the CVA6 over the Embench benchmark suite*

Benchmark	Cycles per instr. (CPI)			Relative error over total cycle count			Avg. abs. error per instr.	
	RTL	Perf.Est.	Abs. diff.	$ISS_{CPI=1}$	$ISS_{CPI=1.45}$	Perf.Est.	$ISS_{CPI=1}$ [CC]	Perf.Est. [CC]
aha-mont64	1.38	**1.37**	0.01	27.4%	5.22%	**0.334%**	0.38	**0.0066**
crc32	1.46	**1.42**	0.04	31.5%	0.656%	**2.86%**	0.71	**0.042**
cubic	1.43	**1.49**	0.06	30.2%	1.18%	**4.30%**	0.53	**0.16**
edn	1.23	**1.13**	0.1	18.7%	17.9%	**7.80%**	0.51	**0.27**
huffbench	1.47	**1.46**	0.01	31.7%	1.03%	**0.659%**	0.88	**0.12**
matmult-int	1.21	**1.13**	0.08	17.5%	19.7%	**7.13%**	0.57	**0.10**
minver	1.56	**1.53**	0.03	36.0%	7.18%	**2.35%**	0.71	**0.073**
nbody	1.41	**1.40**	0.01	29.2%	2.72%	**0.857%**	0.50	**0.045**
nettle-aes	1.03	**1.03**	0	3.19%	40.4%	**0.567%**	0.33	**0.013**
nettle-sha256	1.04	**1.13**	0.09	4.06%	39.1%	**8.40%**	0.31	**0.13**
nsichneu	2.44	**2.56**	0.12	58.9%	40.5%	**4.92%**	1.44	**0.40**
picojpeg	1.38	**1.30**	0.08	27.6%	5.03%	**5.50%**	0.70	**0.17**
qrduino	1.64	**1.53**	0.11	39.1%	11.7%	**7.14%**	0.90	**0.13**
sglib-combined	1.59	**1.65**	0.06	37.2%	8.96%	**3.90%**	0.91	**0.26**
slre	1.43	**1.40**	0.03	30.1%	1.33%	**2.51%**	0.65	**0.12**
st	1.46	**1.43**	0.03	31.6%	0.801%	**1.89%**	0.60	**0.080**
statemate	1.30	**1.28**	0.02	22.8%	11.9%	**1.48%**	0.97	**0.044**
ud	1.65	**1.50**	0.15	39.3%	11.9%	**8.91%**	0.95	**0.20**
wikisort	1.48	**1.45**	0.03	32.4%	1.96%	**2.21%**	0.78	**0.11**
Average	1.45	**1.43**	0.056	28.9%	12.1%	**3.88%**	0.70	**0.13**

capable of modeling both single- and multi-issue concepts. The estimates are calculated and updated on an instruction-by-instruction level. This enables fine-grained profiling of the performance over the course of the target software and makes it possible to track performance variations.

The *code generator* enables a quick and flexible adaptation of the performance estimator to new processor variants. It uses a compact input format, called CorePerfDSL [2], consisting mainly of a simple structural description of the target microarchitecture. From this, the code generator derives the microarchitecture-specific models for the performance estimator.

Since the performance estimator is trace-based, it can conveniently be integrated into existing environments by extending the ISS with a matching instruction monitor. To facilitate the automated adaptation of the monitor to new processor variants, the code generator produces a simulator-independent, JSON-based monitor description that defines the required trace format. This allows users to develop an additional generator to modify the monitor based on this description. Although this approach requires some initial effort from the user, it is a one-time task that further enhances the flexibility of the environment.

Experimental Results

We demonstrate the capabilities of our approach by applying it to the state-of-the-art CVA6 application class processor [3]. The CVA6 is a 6-stage RISC-V processor incorporating advanced microarchitecture concepts, such as a scoreboard, dual-commit, instruction buffering, and dynamic branch prediction. We use the *Embench* benchmark suite [4] compiled for the RV64MI instruction set.As a reference, we use an RTL implementation of the CVA6 embedded in a SystemC environment.

As shown in Tab. 1, our approach provides accurate performance estimates, even though the processor's cycles-per-instruction (CPI) ratio varies over the benchmarks. Considering the overall cycle count, the average relative error rate is 3.88%, which is significantly lower than that of ISS-based approaches that assume a fixed CPI. Additionally, the average deviation between the observed and estimated number of clock cycles (CCs) per instruction is 0.13, highlighting the effectiveness of our approach in profiling performance throughout program execution. Our approach reaches an average simulation speed of up to 15 million instructions per second (MIPS), which is significantly faster than detailed RTL simulations and comparable to custom-implemented simulators.

References

[1] Conrad Foik et al. "Flexible Generation of Fast and Accurate Software Performance Simulators From Compact Processor Descriptions". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43.11 (2024).

[2] Conrad Foik, Daniel Mueller-Gritschneder, and Ulf Schlichtmann. "CorePerfDSL: A Flexible Processor Description Language for Software Performance Simulation". In: *Forum on Specification & Design Languages (FDL)*. 2022.

[3] OpenHW Group. *CVA6 User Manual*. `https://docs.openhwgroup.org/projects/cva6-user-manual/`. Accessed: 2024-03-20. 2023.

[4] Free and Open Source Silicon Foundation. *Embench: A Modern Embedded Benchmark Suite*. `https://www.embench.org/`. Accessed: 2024-03-20. 2021.