A *gem5*-based CVA6 Framework for Microarchitectural Pathfinding

Pierre Ravenel[1,2], Arthur Perais[2], Benoît de Dinechin[1] and Frédéric Pétrot[2] *

[1]Kalray, [2]Univ. Grenoble Alpes, CNRS, Grenoble INP[†], TIMA

Abstract

Succesfully designing a well-balanced general purpose processor targeting an ASIC implementation is an arduous task. As a result, software timing models are generally developped to enable fast design space exploration in the pathfinding phase. Those models are later tightly correlated to RTL as it becomes available, but can still provide valuable insight late in the design phase as they can simulate large workloads and provide non-intrusive access to internal information. In this paper, we leverage the gem5 timing simulator infrastructure to provide a fast timing simulation environment to drive microarchitectural improvements for the RISC-V CVA6 processor. We also depict a correlation flow to ensure that the timing performance model keeps projecting meaningful performance numbers.

Introduction & Motivation

The use of current time computers to simulate future computers has a necessity since the infancy of computer design, see e.g. [1]. Design decisions cover very different matters, from low level implementation details to instruction set architecture (ISA) definition to system-wide architecture dimensioning. In this work we are interested in evaluating microarchitectural choices.

Designers rely on software models to validate the functionality and performance of general purpose processors. Several models are usually maintained. Typical frameworks feature three levels of abstraction: i) The RTL itself, which accurately describes the chip but is excruciatingly slow to simulate ii) A functional simulator which is not actually tied to the design but is helpful in developping low level software in advance and iii) A C/C++ performance model that provides agility through higher abstraction and simulation speed than RTL yet reasonable timing accuracy. The performance model is the obvious tool of the pathfinding phase of a project, when novel functionalities are studied and introduced in the microarchitecture.

Performance models are hard to develop and maintain, and are often kept in-house by processor makers. As a result, academia relies on a few projects to project processor performance, with the understanding that models may be quite far from state of the art [2, 3]. While this allows many researchers to use the same tools, there are no readily available performance models for existing chips.

Given the liveliness of the RISC-V ecosystem, one can ask how the performance models used to develop open RISC-V chips should be made available. We argue that ideally, the RISC-V community should leverage existing open source projects to provide a one-stop infrastructure able to model specific designs. To that extent, this paper walks through the process of building an accurate CVA6 model within the *gem5* [4] infrastructure. The CVA6 was chosen for two reasons. First, it is now maintained by the OpenHW Group, suggesting a reasonably long support ahead. Second, it is quite simple and has low performance, making it a prime candidate for revisiting possible microarchitectural improvements using a high level model. *gem5* was chosen as it is the de facto standard in the computer architecture community.

In this paper, we introduce a *gem5* performance model of the CVA6, propose a correlation suite and calibration methodology to ensure the CVA6 model accuracy, and provide future directions for improving the CVA6 using our framework.

Experimental Methodology & Results

A performance model should provide two things. The first is statistics on the microarchitecture, such as Instruction per Cycle (IPC). The second is ease of modification, to allow evaluation of design decisions. Such a model should ensure performance correlation, but also some amount of code correlation, i.e. similarity between hardware objects and performance model objects. This is a desirable property because a change in the performance model will lead to a related change in the RTL.

Design of a Performance Model in *gem5*

gem5 provides an event-driven simulation environment containing numerous RISC-V-compliant microarchitectural and architectural components. In *gem5*, the ISA model is decoupled from the microarchitecture model. This means that a change to the ISA usually results in minor changes to the processor model code. The counterpart is a rigidity in the programming of performance models. Given its wide use, it is therefore a good candidate for implementing a CVA6 performance model.

*Corresponding author: pierre.ravenel@kalray.eu
[†]Institute of Engineering Univ. Grenoble Alpes

Model Accuracy To design a performance model, one must first define the expected accuracy. The idea is to find a compromise between cycle accuracy and microarchitectural abstraction level. Indeed, it is desirable not to model all hardware components, but rather a functional representation of the main blocks. Thus, rather than realizing a strict cycle-accurate simulator, it is a matter of specifying components with relevant boundaries and their associated interface and timing.

Performance Modeling The construction of a model is done iteratively in two steps: Modeling then validation. We call model validation the fact of verifying that the execution times of the programs on the model are close to those provided by the RTL.

As the RTL of CVA6 already exists, it is question of modeling it from the elementary bricks of *gem5*. The implemented method is as follows:

1. System modeling. It is necessary to have the same memory system and the same peripherals. Emphasis is put on correctly modelling latency and bandwidth.
2. Identification and modeling of memories and flip-flops in the processor, with correct datapaths width.
3. Front end modeling. It is necessary to identify and model the resources of the frontend, e.g. branch predictor, Branch Target Buffer (BTB), ...
4. Back end modeling. This involves identifying the functional units as well as their latency and throughput. The critical part is the identification of how the instructions are issued to the different functional units.

Validation Methodology

Once the model has been built, it needs to be validated. To do this, the use of micro-benchmarks with asymptotic behavior allows to isolate processor bottlenecks [5]. A first set of programs evaluates the performance of the frontend.

- Independent instruction sequences are used to measure front-end throughput.
- Miss-generating jumps validate the instruction cache.

A second set of programs evaluates the backend by placing all test instructions in the instruction cache. It is then possible to evaluate:

- Latency and throughput of the functional units,
- Penalty induced by Read-after-Write, Write-after-Write and Write-after-Read dependencies,
- Latency and throughput of the data/memory cache.

Validation results

To validate our performance model, we need to compare the execution times obtained using the performance model with those of the RTL obtained by simulation or emulation. To that end we use two different benchmarks: riscv-test, that is unit-testing oriented, and Polybench [6], that is compiler optimization oriented. Figures 1 and 2 plot the relative standard deviation (RSD) between the measured IPC of *gem5* and CVA6.

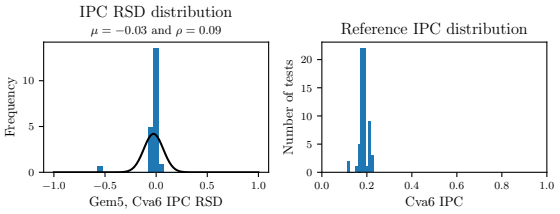

Figure 1: *Riscv-test IPC correlation*

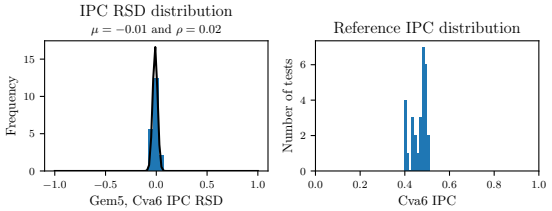

Figure 2: *Polybench IPC correlation*

The variation in the IPC measured between the gem5 model and the CVA6 for the different benchmarks is often close to 0, indicating high correlation.

Summary & Conclusion

This paper presents the method used to build a performance model and validate it from an RTL design. The constructed CVA6 performance model highlights design bottlenecks while facilitating design exploration. With a bottom-up approach, we identified performance losses in corner cases. With a top-down approach, the idea is to improve the frontend to increase throughput under the constraint of supporting RISC-V compressed instructions. Also, working on CVA6 memory accesses should greatly improve backend performance.

References

[1] M Lehman, Rayna Eshed, and Z Netter. "The checking of computer logic by simulation on a computer". In: *The Computer Journal* 6.2 (1963), pp. 154–162.

[2] Tony Nowatzki et al. "Architectural simulators considered harmful". In: *IEEE Micro* 35.6 (2015), pp. 4–12.

[3] Anastasiia Butko et al. "Accuracy evaluation of gem5 simulator system". In: *7th International workshop on reconfigurable and communication-centric systems-on-chip (ReCoSoC)*. IEEE. 2012, pp. 1–7.

[4] Jason Lowe-Power et al. "The gem5 Simulator: Version 20.0+". In: *CoRR* abs/2007.03152 (2020). arXiv: 2007.03152. URL: https://arxiv.org/abs/2007.03152.

[5] Matthew Walker et al. "Hardware-validated CPU performance and energy modelling". In: *IEEE International Symposium on Performance Analysis of Systems and Software*. 2018, pp. 44–53.

[6] Louis-Noël Pouchet and Tomofumi Yuki. *Polybench/C 4.1*. http://polybench.sourceforge.net. 2015.