Snooper: A Flexible Tracing Solution for Fast Simulation and Analysis in RISC-V

Santiago Monserrat Campanello[†], Julian Pavon[†*] and Adrian Cristal[†]

[†]Barcelona Supercomputing Center
[*]Universitat Politecnica de Catalunya
E-mail: **santiago.monserrat@bsc.es**

Abstract

Hardware simulation and modeling are essential for computer architecture research, enabling early-stage evaluation without full hardware implementation. Trace-based simulation tools are advantageous for quickly modeling CPU and memory system performance. Nonetheless, a proper instruction tracing tool for the RISC-V ISA is currently missing. We present Snooper, a fast and flexible RISC-V instruction tracer built as a QEMU TCG plugin. Snooper extracts information (e.g., source/destination registers) per executed instruction and generates customizable trace files, making it compatible with state-of-the-art trace-based CPU simulators. Supporting both user-mode and full-system execution, Snooper enables in-depth RISC-V analysis, including OS-level evaluation. We validate the traces generated by Snooper using ChampSim achieving an average 89% modeling accuracy compared to a RTL Out-of-order CPU running in an FPGA.

Introduction

Computer architecture simulation is essential for research, design verification, and accelerating hardware development by enabling early evaluation without RTL (Register Transfer-Level) development overhead.

In this context, trace-based simulators (e.g.,[1]), are ideal for early-stage architectural exploration, offering fast CPU and memory system modeling when high-level abstraction is sufficient. They achieve this by replaying pre-recorded execution traces instead of dynamically executing binaries, avoiding the overhead of full instruction simulation. This makes trace-based simulators significantly faster than event-driven and cycle-accurate simulators [2].

However, state-of-the-art trace-based simulators lack RISC-V ISA support, as available traces and instruction tracing tools are primarily for x86 and ARM. As a result, developing and testing new ideas based on these available traces is unrepresentative of RISC-V systems and unsuitable for accurate performance analysis.

To bridge the gap between state-of-the-art trace-based simulators and the RISC-V ISA, we propose Snooper, a fast and flexible RISC-V tracer built as a QEMU TCG (Tiny Code Generator) plugin [3]. Snooper leverages QEMU's dynamic binary translation support to efficiently extract 25 metadata characteristics (e.g., source/destination registers, branch behavior, among others) for each executed instruction and generate customizable trace files. Users can define which metadata to include and its order, through an input configuration file. This flexibility enables, for example the use of simulators originally designed for x86 (e.g., ChampSim [1]) with other Instruction Set Architectures (ISAs). Additionally, Snooper supports both user-mode and full-system tracing, enabling RISC-V Operating System (OS) evaluation.

Snooper Design and Features

Snooper, implemented as a QEMU TCG plugin, enables RISC-V trace generation without requiring native execution. QEMU's TCG backend provides an Application Programming Interface (API) to access translation and execution details. By leveraging this API, Snooper efficiently instruments every executed instruction, extracting detailed metadata for trace generation and analysis.

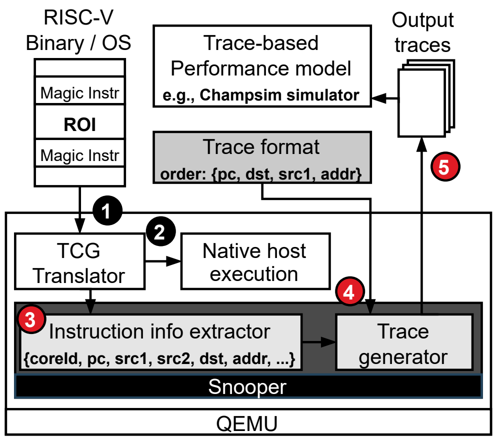

Figure 1: *Overview of Snooper plugging for QEMU.*

Figure 1 presents Snooper's system overview. QEMU loads an executable ❶ —either a user-mode program or an OS —and supports magic instructions to define the Region of Interest (ROI), similar to other tracing tools [1, 4]. Using dynamic binary translation, QEMU translates and executes the input binary ❷, while *Snooper's information extractor* simultaneously gathers metadata for each instruction ❸. The extracted data is then processed by the *trace generator*, which organizes it based on the user-defined configuration ❹, producing per-core trace files ❺.

Snooper supports two trace generation modes: (i) Plain binary compatible with simulators such as ChampSim [1] and Snipersim [5]; and (ii) a Comma Separated Value (CSV) format, which can be used with simple/custom analytical models.

User-mode and full-system support. Snooper supports both user-mode and full-system execution in QEMU. In user-mode, OS calls are emulated, and traces capture only user-space instructions, similar to available traces for x86 and ARM. In full-system mode, QEMU boots a complete OS (e.g., Ubuntu), emulating I/O devices and peripherals, allowing Snooper to trace all executed instructions, including OS calls.

Current trace-based simulators do not support full-system simulation primarily due to the absence of OS-level information in available traces. Unlike other tracing tools, in full-system mode, Snooper includes OS data —such as virtual-to-physical address translations, system calls, among others —in the output traces.

By doing this, snooper enables implementing full-system support in trace-based simulators.

Evaluation

Simulator. We used the ChampSim simulator to model an in-house Out-of-Order RISC-V SoC and configured Snooper via its configuration file to generate traces compatible with ChampSim's format.

Benchmarks. We use Snooper to trace three graph processing benchmarks: Breadth-first search (BFS), Pagerank (PR) and Single-Source Shortest Path (SSSP) from the GAP benchmark [6].

Experiments. To validate the accuracy of Snooper's traces, all experimental results are normalized to our in-house RTL SoC execution time (running on an FPGA). Additionally, we aim to highlight how existing traces, generated for other ISAs, are unsuitable for evaluating RISC-V performance due to differences in ISA design, compilation tools, and software stack maturity. To this end, we compile the evaluated benchmarks for both RISC-V and x86, generate traces using Snooper and Pin respectively, and evaluate them under the same ChampSim configuration.

Results. Figure 2 shows the accuracy results for all the experiments. We make two observations:

(i) Snooper+ChampSim achieves an average 89% accuracy, indicating effective performance modeling.

(ii) On average, Pin-based traces perform 2.6× higher error rates, underscoring the importance of RISC-V trace generation tools for accurate performance evaluation.

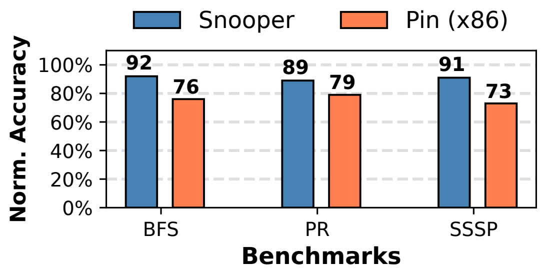

Figure 2: *Simulation accuracy comparison between Snooper and Pin traces, normalized to an RTL RISC-V core implementation.*

Discussion

We proposed Snooper, a flexible RISC-V tracer based on the QEMU TCG plugins. Our evaluation shows that combining Snooper with a state-of-the-art trace-based simulator allows quick evaluation of RISC-V architectures with a adequate accuracy (89%).

Multi-ISA support. Although Snooper generates RISC-V traces, the QEMU TCG plugin shares common data structures across all supported ISAs (e.g., x86 and ARM). This makes it easy to extend Snooper for other ISAs, offering a robust tool for computer architecture research.

Simple performance models. In our evaluation we used ChampSim for the probe of concept of Snooper. However, Snooper can be used together with simpler analytical models to accelerate simulation time.

References

[1] Nathan Gober et al. *The Championship Simulator: Architectural Simulation for Education and Competition.* 2022. arXiv: 2210.14324 [cs.AR]. URL: https://arxiv.org/abs/2210.14324.

[2] Ayaz Akram and Lina Sawalha. "A Survey of Computer Architecture Simulation Techniques and Tools". In: *IEEE Access* 7 (2019), pp. 78120–78145. DOI: 10.1109/ACCESS.2019.2917698.

[3] QEMU Project Developers. *QEMU, TCG Emulation.* https://www.qemu.org/docs/master/devel/index-tcg.html. [Online; accessed 23-Jan-2025]. 2025.

[4] Intel. *Pin.* http://pintool.intel.com/. [Online; accessed 25-Jan-2025]. 2025.

[5] Trevor E Carlson, Wim Heirman, and Lieven Eeckhout. "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation". In: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis.* 2011.

[6] Scott Beamer, Krste Asanović, and David Patterson. "The GAP benchmark suite". In: *arXiv preprint arXiv:1508.03619* (2015).