# Enabling Design Space Exploration of RISC-V Accelerator-rich Computing Systems on gem5

Odysseas Chatzopoulos*, George Papadimitriou, Vasileios Karakostas, Dimitris Gizopoulos
Dept. of Informatics & Telecommunications, University of Athens
{od.chatzopoulos, georgepap, vkarakos, dgizop}@di.uoa.gr

**Abstract**

*The slowdown of CMOS scaling has led to the widespread use of heterogeneous SoCs that combine general-purpose processor cores with customized accelerators. The immense growth of the RISC-V community means that more and more people are interested in using RISC-V cores in their SoCs. It is thus imperative to obtain tools that allow for fast design space exploration and modeling of these designs. In this paper we describe our effort to port gem5-SALAM, a cutting-edge pre-RTL heterogeneous SoC simulator, to incorporate general-purpose RISC-V cores. In such a way, we provide to the RISC-V community a powerful tool to design, evaluate and optimize such systems.*

## Introduction

The end of Dennard scaling and the slowdown of Moore's law [1] have forced a paradigm shift in computer architecture. Single and multicore microprocessors have been succeeded by heterogeneous systems-on-chip (SoC) that are widespread across computing domains, ranging from embedded devices to large datacenters [1]. Such designs incorporate general-purpose processor cores with multiple domain-specific accelerators of varying size and complexity to meet certain power and performance constraints. The RISC-V ecosystem has seen tremendous growth in the past few years, making SoCs that include RISC-V cores extremely attractive. Successfully designing and optimizing such systems depends on the ability to quickly and accurately model and simulate the complex interactions between their components, performing design space exploration across a wide range of design parameters [1]. While RTL simulation is very accurate [1], the relatively low throughput makes simulating the entire SoC excessively time-consuming, thus limiting exploration of vast design spaces to a few hand-picked design points. On the other hand, early (i.e., pre-RTL) microarchitecture-level modeling based on detailed functional and performance models, provides high throughput, ease of use, and high flexibility. To this end, several attempts have been made to fill the gap between low-throughput and accuracy of modeling. gem5-SALAM [2] is a pre-RTL heterogeneous SoC simulator based on the widely used gem5 [3] microarchitecture-level simulator that has managed to significantly bridge this gap.

*Corresponding author: od.chatzopoulos@di.uoa.gr
This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101070238 and No. 101097224. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

gem5-SALAM utilizes a dynamic LLVM-based accelerator model that enables scalable modeling of multiple interacting accelerators with full-system simulation support. In contrast to other similar SoC simulators, such as gem5-Aladdin [4], gem5-SALAM offers a much more accurate representation of runtime-dependent accelerators and enables the study of the interaction between multiple accelerators and other system components, such as the memory system and general-purpose processor cores. Since it harnesses the power and maturity of gem5, gem5-SALAM is highly customizable to the user's needs. Different accelerator architectures, ranging from kernel-level accelerators tied to the system cache, to application-level accelerators attached to the IO bus, can be quickly evaluated and compared.

In its current state, gem5-SALAM supports only the Arm ISA when it comes to the general-purpose cores in the modeled SoC. Even though the accelerator system would be largely identical, this lack of support for the RISC-V ISA poses a significant barrier to researchers and engineers that want to evaluate and deploy SoCs with RISC-V cores. Hence, adding RISC-V support to gem5-SALAM would allow the RISC-V community to take advantage of all the powerful features that this simulation framework has to offer. In this paper, we summarize the main changes and enhancements that are necessary to allow for gem5-SALAM to use the RISC-V ISA for the simulated processor cores.

## Porting gem5-SALAM to RISC-V

Porting gem5-SALAM to RISC-V has been facilitated by the recent introduction of RISC-V full-system execution support into gem5 [5]. The main challenge, however, is to identify the Arm specific components and translate them into the corresponding RISC-V ones. We summarize below the major components that had strong dependency on the Arm platform. Specifically:

1. The interrupt system used by gem5-SALAM hardware components that employs the Arm General Interrupt Controller (GIC) for posting interrupts to the host CPU.
2. The bare-metal software infrastructure that includes the Arm specific boot code and interrupt handler support.
3. The automatic gem5 configuration script generator that uses an Arm gem5 configuration script as a template.

### From (Arm) GIC to (RISC-V) PLIC

gem5-SALAM hardware components, such as the main Communications Interface (responsible for the interface between the host and the accelerator) and the DMA (direct memory access) devices, use the Arm GIC to send and receive interrupts to and from the CPU. These functions have been translated to the Platform Level Interrupt Controller (PLIC) that is present in the current gem5 RISC-V model. After a lengthy debugging process, we managed to identify a bug in the gem5 PLIC implementation that resulted in the incorrect memory mapping of the interrupt claim space. After rectifying this issue, we were able to use interrupts to properly synchronize accelerator functions with the host CPU.

### RISC-V Bare-Metal Application Support

gem5-SALAM uses bare-metal applications running on the host CPU core, to simplify driver development for each new accelerator that is being studied, without limiting the potential for OS support. To run bare-metal applications on RISC-V cores, we use SiFive's Freedom-E SDK since the HiFive platform that gem5 supports is based on the SiFive U54 SoC. By using this SDK we minimize the manual effort required for deploying bare-metal applications and have full bare-metal C library support in the form of Newlib. Moreover, the Freedom-E SDK provides a flexible system for interrupt handling, which, in conjunction with the transition from GIC to PLIC, provides full interrupt support.

### Automatic Configuration Script Generator

The latest version of gem5-SALAM uses an automatic gem5 configuration script generator to simplify the development of accelerator-rich SoCs. This allows for complex configuration scripts to be generated by parsing a single YAML file that contains a description of the simulated system. To port the generator to RISC-V, (1) we swapped the Arm-specific script template to an already existing RISC-V full-system configuration script, (2) made modifications to initialize the gem5-SALAM components, and (3) added the accelerator memory mapped addresses to the address ranges of the RISC-V platform.

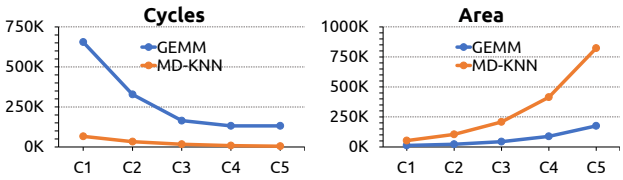


**Figure 1:** *Performance & Area.*

## Proof of Concept

As a proof of concept, we explore the design space of two accelerators for two kernels from MachSuite [6], GEMM (General Matrix Multiply) and MD-KNN (Molecular Dynamics using K-Nearest Neighbors), for five different architectural configurations (i.e., C1 – C5, where higher numbers represent more parallel functional units). Figure 1 shows the results for the execution time (left graph) and the area overhead (right graph). We observe the different trade-offs that gem5-SALAM allows us to study.

## Conclusion & Future Work

By applying these modifications to gem5-SALAM, we managed to enable all the powerful features of this simulation framework for exploring different SoC architectures that include RISC-V general-purpose CPU cores, customized accelerators and memory hierarchies, as well as studying the complex interactions between these components. As future work, we plan to introduce a full-fledged Linux operating system to the simulated machine and develop appropriate drivers for the accelerators, modeling the entire system stack. We also plan to use gem5-SALAM to perform design space exploration of various DNN (Deep Neural Network) hardware accelerators and augment the proposed RISC-V based framework for performance, power/energy consumption, and resilience studies.

## References


[1] Y.S. Shao and D. Brooks. *Research Infrastructures for Hardware Accelerators*. Morgan & Claypool Publishers, 2015.

[2] Samuel Rogers et al. "gem5-SALAM: A System Architecture for LLVM-based Accelerator Modeling". In: *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (2020), pp. 471–482.

[3] Nathan Binkert et al. "The gem5 simulator". In: *ACM SIGARCH computer architecture news* 39.2 (2011), pp. 1–7.

[4] Yakun Sophia Shao et al. "Co-designing accelerators and SoC interfaces using gem5-Aladdin". In: *IEEE/ACM International Symposium on Microarchitecture* (2016).

[5] Peter Yuen Ho Hin et al. "Supporting RISC-V full system simulation in gem5". In: *Proc. Workshop Comput. Architect. Res. RISC-V* (2021).

[6] Brandon Reagen et al. "Machsuite: Benchmarks for accelerator design and customized architectures". In: *2014 IEEE International Symposium on Workload Characterization* (2014).