Entregable E3.2: Desarrollo de un simulador de la arquitectura

Información del Documento

Código del proyecto	001-P-001723
Página web del proyecto	https://drac.bsc.es/es
Plazo contractual	01/06/2019 - 31/05/2022
Nivel de difusión	Público
Naturaleza	Entregable
Autor(es)	Santiago Marco, Julián Pavón, Adrián Cristal, Miquel Moretó
Contribuyente(s)	BSC, UPC y UAB
Revisor(es)	César Hernández (BSC) y Antonio Espinosa (UAB)
Palabras clave	Simulador de microarquitectura cycle-accurate, gem5, procesador fuera de orden, jerarquía de memoria

El proyecto DRAC con número de expediente 001-P-001723 ha sido cofinanciado en un 50% con 2.000.000€ por el Fondo Europeo de Desarrollo Regional de la Unión Europea en el marco del Programa Operativo FEDER de Cataluña 2014-2020, con el soporte de la Generalitat de Cataluña.

Control de cambios

Versión	Fecha	Autor	Comentario
01	15/09/2021	Santiago Marco	Descripción del simulador gem5 usado en el proyecto para modelar el procesador Lagarto Ka y su acelerador de genòmica.
02	30/09/2021	Santiago Marco	Incorporación de los cambios sugeridos por los revisores internos. Entregable enviado.
03	15/11/2021	Miquel Moretó	Revisión final y cierre del documento

Content

1. Introduction

This deliverable aims to present the methodology and tools employed to simulate the general purpose RISC-V CPU design and genomics accelerator in DRAC. Our evaluation framework is based on Gem5, a modular discrete event driven computer system simulator platform widely used in the community [1, 2]. Gem5 is an execution-driven multi-core full system simulator that can do cycle accurate execution of a complete system with devices and an operating system. Gem5's components can be rearranged, parameterized, extended or replaced easily to suit researcher's needs. Gem5 supports multiple Instruction Set Architectures (ISA), and provides different CPU models and a detailed model of the cache hierarchy.

The gem5 simulator developed in the project can be found in an internal repository in the BSC gitlab (https://repo.hca.bsc.es/gitlab/gem5-bsc/gem5). This repository is shared with the different partners in the DRAC project.

2. General Purpose Processor Design

The DRAC project is following a hardware-software co-design approach to ensure the final hardware design meets real-world High-Performance Computing (HPC) and bioinformatics application requirements. As described in deliverable E3.1 "Informe sobre el rendimiento de las aplicaciones y pipelines de análisis en sistemas de computación", the applications selected for the project are a set of basic building blocks widely-used across many genomics applications: Wavefront Alignment Algorithm (WFA), Genomic Sequence FM-Inxex, and Genome Mapper. Moreover, initial analysis of these applications indicate the key hardware resources requirements for the general purpose processor design:

- **CPU performance requirements**: HPC processors usually rely on a large-scale multicore composed of many aggressive out-of-order cores. In this project, we want to analyze if the applications of interest really benefit from having such a power hungry design or, instead, have enough performance with a moderate out-of-order design (like the Lagarto Ka processor) or even a high performance in-order design (like the Sargantana processor).
- **Requirements for Single Instruction, Multiple Data (SIMD) vector requirements**: A key aspect will be the analysis of the trade-offs for the different vector lengths permitted in the RISC-V vector ISA. The simulation results from the selected applications will enable the exploration of different vector lengths via manual code vectorization, auto-vectorization, and the use of generic performance libraries.
- **Memory hierarchy design requirements**: We will analyze the memory performance requirements of the selected applications to understand their sensitivity to memory bandwidth and memory latency, which requires a careful design of the memory hierarchy, both on-chip and off-chip.

Since there is no available hardware with RISC-V vector support and the envisioned ISA extensions, we utilized the full-system Gem5 simulator [1,2] for our architectural exploration. Gem5 is an execution-driven multi-core full system simulator that can do cycle accurate execution of a complete operating system. Gem5's components can be rearranged, parameterized, extended or easily replaced to suit researchers' needs. It simulates the passing of time as a series of discrete events and its intended use is to simulate one or more cores in a computer system in various ways.

Gem5 is more than just a simulator; it is a simulation platform that lets researchers use as many of its premade components as they want to build up their own simulation system. Gem5 can simulate a complete system with devices and an operating system in full system mode (FS mode), or user space only programs where system services are provided directly by the simulator in syscall emulation mode. It supports several ISAs such as Alpha, Arm, MIPS, x86 and RISC-V. It also provides different CPU models: a simple one-CPI (clock cycle per instruction) CPU; a detailed model of an in-order CPU, and a detailed model of an out-of-order CPU. The in-order and out-of-order CPUs model an execution pipeline in detail with multiple stages (i.e. fetch, decode, rename, dispatch, issue, execute, writeback, commit) and a parameterized width. The CPU models also include different branch predictors and hardware prefetchers for improved performance. Memory instructions are modeled in detail, accessing a multi-level cache hierarchy and an off-chip main memory that are also modeled in detail. In fact, Gem5 CPU and memory models have been validated against existing processors in the past years with very good accuracy and simulation speed.

Gem5 supports check-pointing and KVM emulation to accelerate system and benchmark initialization using less detailed CPU and memory models. In this project, we employ the check-pointing capabilities so that simulations start right at the parallel sections. All the experiments run with the most detailed configurations available for each architecture trying to resemble a real system. Moreover, simulations are run in full system mode and the simulated system runs an Ubuntu v16.04 OS.

3. Simulated General Purpose Processor Design in Gem5

We have configured our simulation infrastructure to resemble the architecture envisioned in the project. As shown in Table 1, we evaluate three different core models: a high performance core that resembles an aggressive high-performance out-of-order core (similar to Arm's A72 processor), a moderate out-of-order core (similar to Lagarto Ka), and a simple in-order core (similar to Sargantana). Each core has private L1 instruction and data caches, and a shared last-level L2 cache. The off-chip memory technologies can then be modeled on top of this multi-core setup and we plan to use both pin-based technologies like DRAM (DDR4) as well as High-Bandwidth Memory (HBM) stacks with wider and abundant memory channels that are based on silicon interposer technologies, bypassing the physical limitations of pin-based off-chip memory. Table 1 details the architectural parameters we use.

Table 1: Gem5 simulator configurations

Core (aggressive)	8-wide issue/retire, 92-entry instruction queue, 192-entry ROB, 48 LDQ + 48 STQ, 1 vector processing unit (VPU), 2GHz
Core (moderate)	2-wide issue/retire, 16-entry instruction queue, 64-entry ROB, 16 LDQ + STQ, 1 vector processing unit (VPU), 1.5GHz
Core (in-order)	Single-issue in-order design, 1 vector processing units (VPU), 1.5GHz
Private caches (in-ord/mod/aggr)	L1I/LID: 16KB/32KB/64KB, 4-/4-/8-way, 2 cycle, 2/4/8 MSHRs;
Last-level cache (in-ord/mod/aggr)	128/256/512KB, 8-/8-/16-way, 10 cycle, 4/8/16 MSHRs
Main memory	1) 4 DDR4-2400 channels, 2 ranks/channel, 16 banks/rank, 8KB row-buffer. 128-entry write and 64-entry read buffers per channel 2) 8 HBM2 channels (1 stack), 32GB/s per channel peak bandwidth. 128-entry write and 64-entry read buffers per channel

The gem5 simulator allows us to modify different key parameters in the microarchitecture such as the vector length of the VPU, the number of units (both scalar and vector functional units), the sizes and latencies of all cache levels, or the memory technology (HBM and DDR). Such flexibility enables the DRAC researchers to perform a design space exploration of different designs of the general purpose processor and accelerators.

To understand the performance trade-offs of the WFA algorithm, we will make use of various representative input datasets for different sequence lengths (100 and 1K characters) and different error rates (1% and 10%). These sequence lengths and error rates are characteristic of real sequencing machines; for example, 100 characters long, like those produced by Illumina sequencers, or sequences of 1K characters long, like those produced by long-read sequencer by PacBio or Nanopore. All datasets will be simulated, for a given sequence length and error rate, using a sequence simulator provided on the official WFA repository. Note that these simulated datasets are widely used by the community to evaluate bioinformatics applications and are considered the standard simulating this application. The gem5 simulator can execute the WFA algorithm in full system with the operating system support and model all the software and hardware details of the final system.

4. Conclusions

In this deliverable, we have presented the methodology and tools employed to simulate the general purpose CPU design and accelerators in DRAC. Our evaluation framework is based on Gem5, a modular discrete event driven computer system simulator platform. In the gem5 simulator, we model the design of the Lagarto Ka processor, a 2-way 64-bit out-of-order RISC-V general purpose processor capable of booting the Linux operating system. Coupled with the Lagarto Ka processor, a set of specialized accelerators are designed in the DRAC project to facilitate the execution of specific applications. Such accelerators are also modeled in the gem5 simulator.

5. References

[1] N. Binkert, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, D. A. Wood, B. Beckmann, and G. Black, S. K. Reinhardt, A. Saidi, A. Basu, and J. Hestness, D. R. Hower, T. Krishna. The Gem5 simulator, August 2011.

[2] J. Lowe-Power, A. Mutaal Ahmad, A. Akram, M. Alian, R. Amslinger, M. Andreozzi, A. Armejach, N. Asmussen, B. Beckmann, S. Bharadwaj, G. Black, G. Bloom, B. R. Bruce, D. Rodrigues Carvalho, J. Castrillon, L. Chen, N. Derumigny, S. Diestelhorst, W. Elsasser, C. Escuin, M. Fariborz, A. Farmahini-Farahani, P. Fotouhi, R. Gambord, J. Gandhi, D. Gope, T. Grass, A. Gutierrez, B. Hanindhito, A. Hansson, S. Haria, A. Harris, T. Hayes, A. Herrera, M. Horsnell, S. Ali Raza Jafri, R. Jagtap, H. Jang, R. Jeyapaul, T. M. Jones, M. Jung, S. Kannoth, H. Khaleghzadeh, Y. Kodama, T. Krishna, T. Marinelli, C. Menard, A. Mondelli, M. Moreto, T. Mück, O. Naji, K. Nathella, H. Nguyen, N. Nikoleris, L. E. Olson, M. Orr, B. Pham, P. Prieto, T. Reddy, A. Roelke, M. Samani, A. Sandberg, J. Setoain, B. Shingarov, M. D. Sinclair, T. Ta, R. Thakur, G. Travaglini, M. Upton, N. Vaish, I. Vougioukas, W. Wang, Z. Wang, N. Wehn, C. Weis, D. A. Wood, H. Yoon, E. F. Zulian. The Gem5 Simulator: Version 20.0+. arXiv:2007.03152, September 2020.