Extending OpenPiton framework towards the HPC domain: first steps

Xabier Abancens[1], Mohsin Shahbaz[1], Teresa Cervero[1], John Davis[1], Guillem Lopez[1,2], Miquel Moretó[1,2] *

[1]Barcelona Supercomputing Center (BSC), Barcelona,
[2]Universitat Politecnica de Catalunya (UPC), Barcelona

Abstract

HPC applications are demanding more specialized accelerators to tackle with their increasing complexity, and requirements. This context, together with the fact that Europe is promoting the use of the open source RISC-V ISA, brings the opportunity of exploring the design space for specific working scenarios. For research and exploration purposes, OpenPiton could be a well-suited framework for future manycore accelerators designs, although there are several areas that should be improved to achieve HPC performance. This paper presents an extension of the OpenPiton framework towards the HPC domain, starting from modifying the computational elements, the Tiles. In addition, this paper extends the SystemVerilog verification environment, but also the FPGA implementation.

Introduction

Europe is focused on digital autonomy, especially for High Performance Computing (HPC). As a consequence, an undeniable effort is being doing in the adoption of the open-source RISC-V ISA for the development of new architectures. In this scenario, OpenPiton [1] could play a significant role; at least for research and exploration purposes.

OpenPiton is a manycore research framework that comprises different tools and modules to build, test and implement RTL designs. Originally, it was developed for the SPARC v9 architectures (OpenSPARC T1). Later on, the framework has been adapted for RISC-V architectures, being the one with Ariane core [2] a reference design. Taking this design as a golden reference, it is clear that is far from being HPC-friendly. To correct this situation, several modules should be incrementally improved: 1) the Tile, 2) the Memory hierarchy (L1, L2 and L3 levels), 3) the NoC. The targeted baseline accelerator is shown in Figure 1, composed by the OpenPiton framework plus a new Tile, based on a RISC-V design Drac Vector IN-Order (DVINO) processor [3].

DVINO as a Tile in OpenPiton

DVINO is a system composed by a Scalar Core (SC) with a coupled accelerator, connected through the Open Vector Interface (OVI) [4].

*The MEEP project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 946002. The JU receives support from the European Union's Horizon 2020 research and innovation program and Spain, Croatia, and Turkey

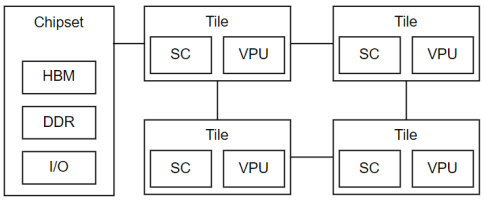

Figure 1: *Block diagram of a multicore system based in OpenPiton with DVINO as a Tile*

Compared to the Ariane Tile, DVINO brings the possibility for: 1) a straightforward compatibility with all RISC-V cores within the Lagarto family; including in-order and out-of-order cores, and 2) support for one or multiple accelerators; in this paper we focus on the vector processing unit (VPU).

Enhancements at Tile-level

DVINO includes a Lagarto Hun SC. A single-issue in-order RISC-V core that implements the 64-bit RV64IMA scalar RISC-V ISA v2.2 and privileged ISA v1.11, with a 5-stage pipeline datapath [5]. In addition, as an accelerator, it supports a VPU based on a design developed for the European Processor Initiative (EPI) project [6].

Scalar core modifications At core level, the enhancement were applied on the pipeline; moving from a basic Lagarto Hun RV64IMA(V) with 5 stages to a Lagarto Hun RV64GC(V) with 6 stages.

Firstly, support for single and double precision floating point operations (F/D extension version 2.2) was incorporated. Secondly, support for the compress instruction extension (C version v2.0) was added. As a result, the core has the capability of booting Linux distributions; currently Fedora.

Vector Processing Unit modifications The original VPU was adapted in multiple ways, being the most relevant for this paper: 1) fusing vector lanes in pairs, 2) adding the capability of configuring the amount of active lanes (2, 4, 8 or 16 lanes).

DVINO integration in OpenPiton To guarantee a smooth integration of DVINO, as a Tile in OpenPiton, we reused the Ariane L1 cache subsystem. Consequently, we adapted the cache interfaces on the SC side to ensure a correct communication protocol with L1 caches. A Performance Monitoring Unit (PMU) is added per core, which includes CSRs for getting metrics about the Tile behavior, including scalar and vector instructions.

Verification

The reference design OpenPiton+Ariane provides a self-checking simulation environment to run a regression and continuous integration bundles. The setup is able to compile and run tests on the simulation model from an available test suite [7] containing RISC-V tests and benchmarks written in assembly or C language.

Verification environment enhancements

Implementation of ISA Scoreboard The major component added to the framework is the Verification Scoreboard, connected to the commit stage of the scalar core. It performs the ISA state checking at each instruction commit. To achieve this per-commit comparison, the open source Spike ISS [8] is modified to step the simulation by one instruction and integrated with the testbench with DPI functions.

Implementation of Memory Scoreboard A memory scoreboard is added to monitor any memory transaction going out of the core. It contains a store checker which compares the store address and data being stored after each instruction with that of Spike. It also monitors any intrinsic stores done by the core such as updates of A and D bits of page tables.

Addition of an ISA Coverage model A coverage model is developed in System Verilog, consisting of two major parts: 1) Unprivileged and, 2) Privileged RISC-V ISA. The Unprivileged coverage model targets the ISA instructions for each implemented RISC-V extension in the scalar core. The coverpoints covers the fields of each instruction and then taking the cross coverage to cover every possible combination. The Privileged ISA coverage model is made on the grounds of RISC-V Privileged ISA spec, focusing on the topics of Control Status Registers, Interrupts and Exceptions, Virtual Memory and their interrelated behaviours. These coverage models have enabled the testbench to cover and verify RTL functionalities not simply visible through code coverage.

Continuous Integration (CI) on Gitlab

A Gitlab CI has been configured with two pipelines: sanity regression and nightly regression. The sanity regression is run for every push to the repository and consists of Compliance tests and some random tests targeting the available extensions in the core i.e. RV64GC(V). The nightly regression is a periodic pipeline which consists of a set of 200 random tests generated with riscv-dv [9].

FPGA results

Extending FPGAs support One of our contributions to the OpenPiton project is the additional support for the Alveo U280 and U55C FPGAs, from Xilinx. These cards offer higher density in resources and memory, among other features.

Supporting a custom FPGA Shell In addition, the framework has been adapted to be compatible with a flexible, adaptable and configurable FPGA Shell [10], which provides a seamless communication between the host and the accelerator implemented in the FPGA.

References

[1] Jonathan Balkind et al. *OpenPiton: An Open Source Manycore Research Framework.* URL: http://doi.acm.org/10.1145/2872362.2872414.

[2] Jonathan Balkind, et al. *OpenPiton+Ariane: The First SMP Linux-booting RISC-V System Scaling from One to Many Cores.* https://carrv.github.io/2019/papers/carrv2019_paper_12.pdf.

[3] Guillem Cabo et al. *DVINO: A RISC-V Vector Processor Implemented in 65nm Technology.* DOI: 10.1109/DCIS55711.2022.9970128.

[4] Roger Espasa et al. *AVISPADO - VPU Interface.* URL: https://github.com/semidynamics/OpenVectorInterface.

[5] Jaume Abella et al. *An Academic RISC-V Silicon Implementation Based on Open-Source Components.* DOI: 10.1109/DCIS51330.2020.9268664.

[6] *The European Processor Initiative.* https://www.european-processor-initiative.eu/accelerator/.

[7] RISC-V Software. *riscv-tests.* URL: https://github.com/riscv/riscv-tests.git.

[8] RISCV. *The Spike RISC-V ISA Simulator.* https://github.com/riscv/riscv-isa-sim. Sept. 2020.

[9] Google, Inc. *Random instruction generator for RISC-V processor verification.* URL: https://github.com/google/riscv-dv.

[10] Daniel J. Mazure, et al. *Enabling RISC-V in Large Scale FPGA Platforms.* https://open-src-soc.org/2022-05/media/posters/4th-RISC-V-Meeting-2022-05-03-Daniel-Mazure-poster-abstract.pdf.