RISC-V Day Tokyo 2024 Winter - Jan 16, 2024

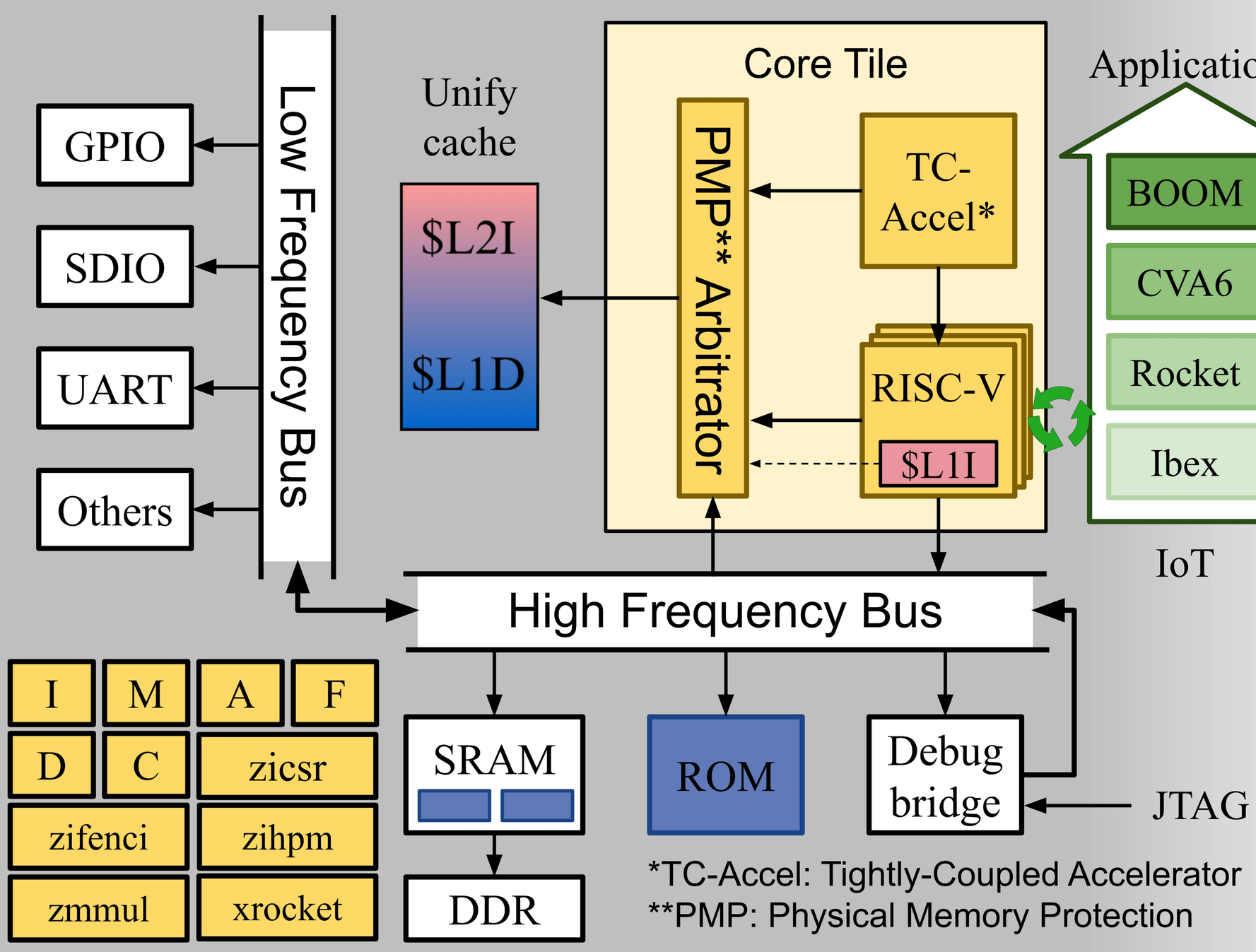
A Multi-purpose RISC-V Framework

Binh Kieu-Do-Nguyen, Cong-Kha Pham, and Trong-Thuc Hoang
University of Electro-Communications (UEC), Tokyo, Japan

I. INTRODUCTION

This poster presents our motivated RISC-V framework. The open-source RISC-V Instruction Set Architecture (ISA) is increasingly gaining popularity and acceptance, both within the industry and academic communities. The modularity of the RISC-V ISA allows the processors to adapt to a wide range of requirements, from Internet of Things (IoT) devices to application-class processors. In addition, openness facilitates the accessibility of diverse open-source and commercial implementations. These elements could encompass several areas such as security, artificial intelligence, low-power design, and multi-core. Because the RISC-V system designs are subject to distinct requirements, our primary motivation for this work is the provision of an efficient RISC-V framework that offers flexibility to accommodate varying requirements, ranging from "IoT-class" to "application class" capabilities. The foundation of our RISC-V framework is the widely recognized Chipyard framework. This framework enables us to provide a high-level description of the RISC-V system using Scala, a powerful functional programming language. Using Chipyard as a foundation, we enhance the existing configuration and incorporate additional modules to expand the scope of the RISC-V framework. We focus on the interaction between the necessary functionality for small IoT devices, the execution of application-class tasks (such as virtual memory, caches, and several modes of privileged operation), the modularity of integrated components and the energy consumption. The ultimate objective of this effort is to provide an abstract-to-tapeout framework that can be readily managed by RISC-V researchers.

II. The RISC-V framework



The RISC-V Framework

- Reconfigurable from IoT to Application-class.
- Support multi-core.
- Customizable peripherals.
- Support tightly-coupled accelerator.
- Support debugging/ programming through JTAG (OpenOCD, GDB).

Bus sytem

- Hierarchical bus system.
- Based on TileLink.
- Deadlock-free.
- Burst + Atomic access.
- Convertible with AXI-4.

Memory system

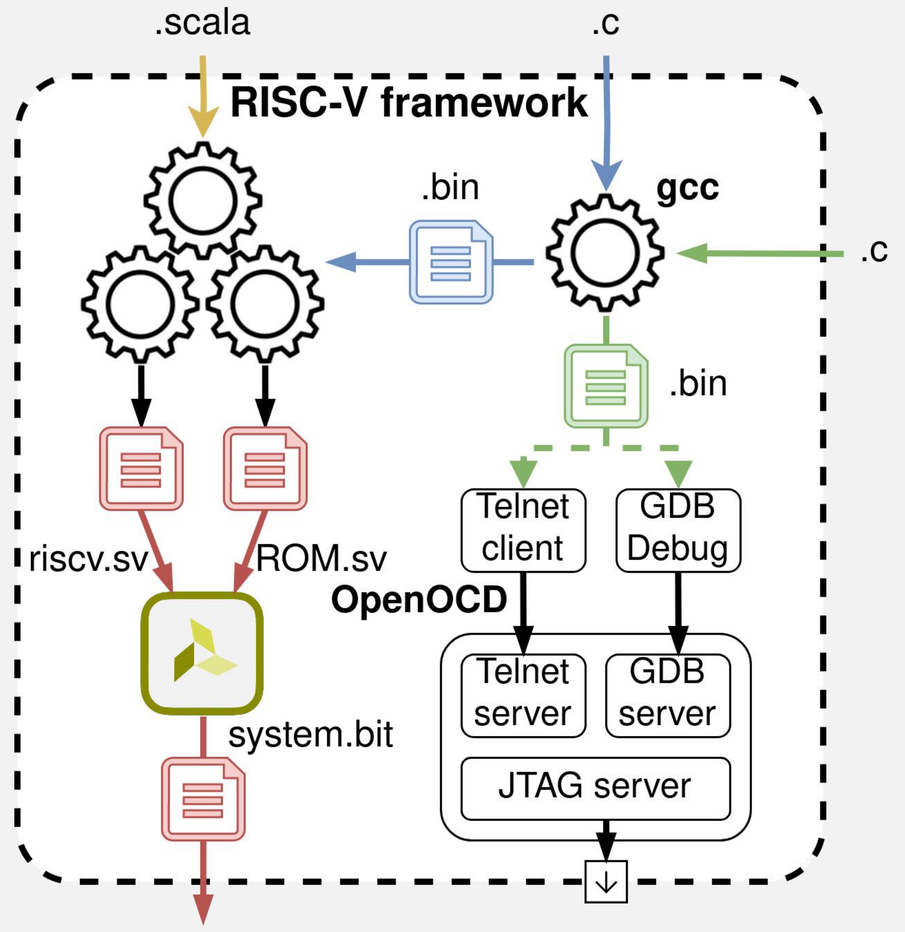
- ROM stores BootROM.
- Two-level instruction cache ($L1I, $L2I).
- Unify cache:
  - L2 Instruction cache.
  - L1 Data cache.
  - Programmable.
  - Direct access.
- Scratchpad SRAM cache for DDR.

III. IMPLEMENTATION & RESULTS

- **Platform** : Arty A7-100T (XC7A100TCSG324-1)
- **RISC-V Core** : Rocket - RV32imac
- **Cache** : 4-KB (I-Cache) 8-KB (Unify D-Cache)

Config	LUTs	Registers	BRAMs	DSP	Freq (MHz)
Tiny	10,198	5,241	6	4	119
DDR	18,718	12,940	6	4	107
DDR + SRAM	22,234	7,068	141	4	104

IV. WORKS FLOW



REFERENCES

[1] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 11, pp. 2629-2640, Nov. 2019.
[2] D. Rossi et al., "PULP: A parallel ultra low power platform for next generation IoT applications," 2015 IEEE Hot Chips 27 Symposium (HCS), Cupertino, CA, USA, Aug. 2015, pp. 1-39.
[3] D ̈orflinger et al., “A Comparative Survey of Open-Source Application-Class RISC-V Processor Implementations,” in ACM International Conference on Computing Frontiers, May 2021, p.12–20.
[4] P. D. Schiavone et al., “Slow and steady wins the race? a comparison of ultra-low-power risc-v cores for internet-of-things applications,” in PATMOS, Sept 2017, pp. 1–8.
[5] A. Amid et al., "Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs," in IEEE Micro, vol. 40, no. 4, pp. 10-21, 1 July-Aug. 2020.

Corresponding author: Binh Kieu-Do-Nguyen (binh@vlsilab.ee.uec.ac.jp)