Simply-V: A RISC-V Reconfigurable Playground Soft-SoC for Open Hardware Research and Fast Prototyping

Vincenzo Maisto, Stefano Mercogliano, Manuel Maddaluno, Alessandro Cilardo
Department of Information Technology and Electrical Engineering
University of Naples Federico II, Naples, Italy
{vincenzo.maisto2, stefano.mercogliano, manuel.maddaluno, acilardo}@unina.it

Abstract—**The recent rise of open hardware, mainly driven by the momentum of the RISC-V ecosystem, has sparked significant innovation in the development of open-source CPUs and SoCs. This movement has enabled broad exploration across academia and industry, fostering collaboration and reuse. However, the diversity and openness that empower this space also introduce challenges: academic projects often fall short of industry-grade robustness, lack of standardization, and simulation limitations. To ease the work of researchers some key challenges must be faced in open hardware development: platforms' reconfigurability, ease of integration of third-party IPs, and support for technological heterogeneity. To address these issues, we present Simply-V, a flexible, FPGA-based soft-SoC platform designed for rapid prototyping and open hardware research. Simply-V enables plug-and-play support for multiple CPUs, IPs and accelerators, offers structured configurability across embedded and highperformance profiles, and supports the integration of both RTL and HLS-based components. We demonstrate the SoC's capabilities through platform-fair CPU benchmarking and the iterative development of HLS-designed convolutional accelerators, showcasing simplified fast prototyping, configurability, and heterogeneous IP support on real hardware. Simply-V is openly available at https://github.com/HiSA-Team/Simply-V.git.**

Index Terms—RISC-V, FPGA, Fast-Prototyping, Experimental Research.

I. INTRODUCTION

In recent years, open hardware has experienced a remarkable surge, largely fueled by the RISC-V open ISA, which has become a catalyst for research into open-source CPUs and Systems-on-Chip (SoCs) across both academia and industry. On one hand, this rich, diverse, and open environment fosters knowledge sharing and promotes the reusability of hardware solutions. On the other hand, academic projects often fall short of industry-grade standards in areas such as documentation, usability, and long-term maintainability. As a result, open hardware researchers frequently encounter significant challenges, not only in reproducing experimental results, but also in building upon existing work. Most setups are often hard to reproduce, and the inherent heterogeneity can result in inconsistent or non-comparable performance figures. As a mitigation to these challenges, one would wish for a simple, verified and hardware-ready playground platform for open hardware research that is reconfigurable, easy to use and reuse in larger systems. Such an achievement, however, is nontrivial for several reasons. First, validating CPUs in realistic scenarios, such as running full operating systems or benchmarking memory hierarchies, goes beyond basic testbenches. Second, while reusable IP blocks like accelerators, peripherals and protocol bridges are widely available, they often lack consistent interface standardization and toolchain compatibility, shifting focus from open hardware research to low-level troubleshooting. Lastly, seamless configurability remains a major roadblock. Tasks like address mapping, dependencies management, memory and bus resizing, or clock domain crossing (CDC) are often hardcoded or require manual rework, limiting scalability and slowing down design iterations.

To address these challenges, we present Simply-V (pronounced "simplify-ve"), an easy-to-deploy, flexible, and extensible soft-SoC platform for rapid prototyping, open hardware research and development. Simply-V provides a simple, FPGA-based, hardware-ready playground platform for full-stack evaluation on real physical devices in open hardware research. The platform offers a structured reconfiguration flow, is portable across a wide range of FPGA devices, supporting both embedded and high-performance profiles. It enables drop-in integration of multiple CPUs and accelerators, along with a configurable interconnect managed through a high-level flow. We demonstrate the capabilities of Simply-V with a platform-fair benchmarking of set open-source CPUs, across the embedded and high-performance computing (HPC) profiles, in both 32 and 64 bits. Additionally, we validate the integration of custom IPs, support for technological heterogeneity and fast design iterations by deploying a set of incrementally-designed high-level synthesis (HLS) convolutional accelerators.

II. BACKGROUND AND MOTIVTION

A. A Flexible SoC for Open Research and Fast Prototyping

While many reconfigurable RISC-V-based platform designs have been proposed, a significant number of them are either proprietary or not publicly accessible. Conversely, most opensource RISC-V CPUs are distributed with minimal testing environments, valuable for evaluating the processor's functionalities but not meant to serve as hardware-ready SoCs.

Manuscript received April 30, 2025; revised July 23, 2025; accepted July 25, 2025. Published September 2, 2025.
Issue category: Special Issue on DSD/SEAA 2025 on Works in Progress (WiP) Session, Salerno, Italy, Sept. 2025.
Paper category: Short
DOI: doi.org/10.64552/wipiec.v11i1.86

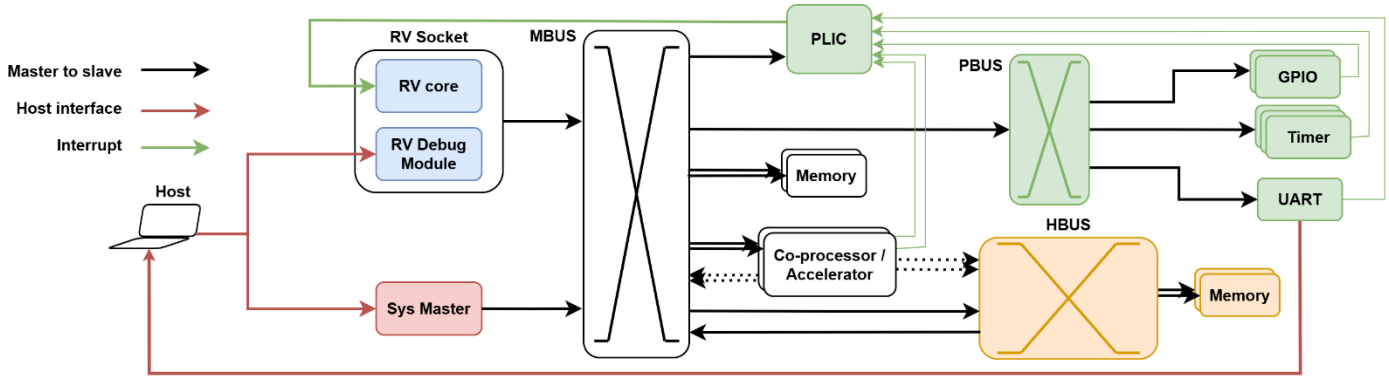

Fig. 1. General architecture and on-chip interconnect of Simply-V. It features a main bus (MBUS), a peripheral bus (PBUS) for low-speed devices and a high-performance bus (HBUS) for high-bandwidth memory accesses, suitable for accelerators and co-processors. On the MBUS, the RV Socket hosts a RISC-V processor and debug module. Finally, the SysMaster grants the host full control of the platform and master access to the SoC interconnect.

More sophisticated open-source SoCs do exist, such as lowRISC's OpenTitan [1], however, these systems lack the right reconfigurability across a wide spectrum of technology needed for open-hardware research. In contrast to fixed-functionalities SoCs, frameworks for SoC generation can offer more flexibility, such as ESP [2] and Chipyard [3]. Similarly to Simply-V, using a configuration-driven design flow, they can generate RTL for a complete SoC, including the CPUs, caches, interconnects, and co-processors. On the other hand, Chipyard specifically focuses on verification of ASIC tape-outs, and ESP is mostly oriented at tiled architectures, integrating third party SoCs and IP cores. Instead, Simply-V is explicitly targets FPGA platforms and SoC architecture, with fast prototyping for academic research as its primary aim.

B. A Practical Alternative to Simulation

Simulation frameworks have long been the cornerstone in digital design validation. System emulation platforms such as QEMU, and instruction set simulator like Spike, are clearly limited to functional validation, and cannot be used for performance evaluation. More advanced tools, such as gem5 [4] and event-driven simulators, like GVSoC [5], offer reasonable trade-offs between timing accuracy and simulation time, but tend to be platform-specific and require a reimplementation of the simulated modules. On the other hand, cycle-accurate RTL simulators are often prohibitively slow, namely when simulating long-running programs like booting an OS. Hybrid hardware/software co-simulation approaches attempt to mitigate these issues, but they rely on custom intermediate representations, or emulation on expensive FPGA platforms [6].

Simply-V addresses these challenges by providing the fidelity of hardware execution without the burden of RTL design and platform integration. Such flexibility allows rapid prototyping, fast design iterations, and real-system validation beyond cycle-accurate simulations or FPGA-based emulations.

III. ARCHITECTURE

This section describes the design principles of our Simply-V and design challenges it addresses. To ease the work of researchers and practitioners, we focus on (1) system-wide reconfigurability, (2) ease of integration of custom IPs and (3) technological heterogeneity as pivotal requirements.

MBUS: the Simply-V architecture, depicted in Figure 1, is based on fully parametric and reconfigurable, yet simple, main bus (MBUS) interconnect, based on AMBA AXI4. Most memories in Simply-V, generically encompassing ROMs, onchip SRAMs and external DRAMs, are mapped on the MBUS.

PBUS: Low-end and low-frequency peripherals, which commonly require a limited range of addresses for register file data and control, are collected in the peripheral bus (PBUS), as an additional slave to the MBUS. By design, the PBUS is meant for non-performance-critical bus traffic, hence we opt for an AXI4-lite interconnect.

HBUS: Our platform is also designed of accelerator development and HPC configurations, hence a high-bandwidth bus (HBUS) interconnect is exposed as a further slave of the MBUS. The HBUS offers streamlined, long-word access to more performant memories, such as DDR banks or HBM channels. The HBUS is suitable for high-performance accelerators, which require high-bandwidth to external DRAM memories, such as AI engines or vector co-processors.

PLIC and interrupts: Interrupts are managed by an implementation of the RISC-V standard platform-level interrupt controller, namely PLIC, integrated as a custom unit leveraging our custom IP flow, detailed in Section III-C.

SysMaster: Since Simply-V is primarily designed for research and development, host-side debug is a fundamental feature. We explicitly expose on the MBUS a host-side connection, the SysMaster. It allows the user to directly interact with the SoC, e.g. to inject faults or read-back data over a high-speed link, e.g. PCIe, rather than low-speed JTAG.

Cross-profile UART: A UART peripheral is hosted in the PBUS for both embedded and HPC profiles. For embedded, we leverage a physical UART IP over a PMOD connector. In case of HPC deployment, such as on PCIe acceleration cards, a PMOD connection is typically not available. Therefore, we design a virtual UART module to emulate the same behaviour of its physical counterpart over the PCIe link.

A. SoC Configuration Flow

Simply-V provides a lightweight, parameter-based configuration flow to re-shape the platform at build time and restructure the whole SoC to adapt it to their experimental needs.

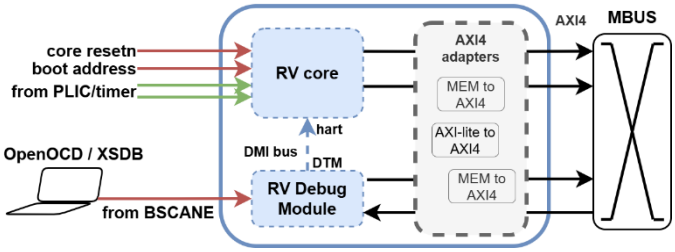

Fig. 3 . Architecture of the RV Socket, with CPU-specific logic enclosed in dashed lines.

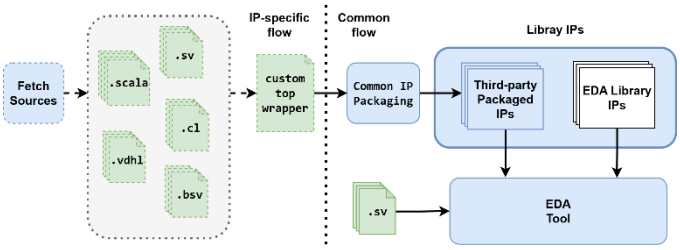

Fig. 2. Custom IP packaging flow. On the left, in dashed lines the IP-specific steps. On the right, the common steps managing packaged IPs as library elements in the target EDA tool.

Four modular input tables are required, namely a systemlevel configuration, e.g. target CPU, RISC-V XLEN, and three configuration files for MBUS, PBUS and HBUS, respectively, for interconnect, address map and CDC management.

Buses are reconfigured in a transparent, automated and verified process. The necessary RTL modifications to remap addresses and interconnections are automated and hidden to the user. Peripheral IPs, such as timers, GPIOs or accelerators, can be optionally instantiated in one or multiple instances and their clock frequency, alongside the CPU clock, can be controlled from configuration files.

B. *RV Socket and Debug Support*

Simply-V aims at providing a fast deployment flow that targets different processors with a vendor-agnostic plug-in framework. To address such needs, we introduce the RV Socket, a modular CPU wrapper offering a unified interface for the RISC-V cores towards the MBUS for operation and the host for debugging. Figure 2 depicts the architecture of the RV Socket. In the following, we present motivation and detail our design choices.

1) Unifying CPU Interfaces: In order to provide a vendor-independent interface for CPUs, we leverage our custom IP flow to provide a packaging framework for compatible and re-usable adapters. Adapters can be either imported or implemented from scratch, and deployed alongside the RISCV CPU to provide a unified interface for all CPUs, allowing plug-in CPU support.

2) RISC-V External Debugging: RISC-V CPUs can support a Debug Transport Module (DTM) for external debugging. The RISC-V debug specification defines a Debug Module Interface bus (DMI), but the implementation is left to the designer. Consequently, each RISC-V core comes with a tightly-couples DTM. With our simple configuration flow, the transition between CPUs remains seamless, which transparently enables the right DTM and compatible DMI interconnect.

C. *Custom IP Packaging*

For Simply-V, we design a custom IP packaging methodology to ease the integration of custom and third-party IPs. We allow users to package RTL or other HDL sources into a selfcontained IP, namely a blunt out-of-context netlist, with no remaining references to its source code. Such a flow is depicted in Figure 3. The first step of the packaging is to provide the necessary sources, resolve internal dependencies, and is IPspecific. Simply-V poses no constraints on this step, other than providing a custom top wrapper module for all IP sources. The second step is unified for all IPs and automatically builds such top module in a library IP element. Consequently, this strategy enables Simply-V to integrate third-party IPs with potential non-compatible code bases, effectively turning third-party IPs in simple and off-the-shelf library elements.

D. *Managing Clock Domain Crossing*

Managing clock domains can be difficult and deploying a whole SoC in the same domain might be inefficient in performance and power. Our configuration flow allows for the slaves of the main bus to be clocked at different frequencies, with automated and verified CDC bridges deployment. A main clock domain is shared by RV Socket, MBUS, PLIC and BRAM memories, as such modules typically show no advantage in a dedicated domain. The PBUS hosts all of its low-speed peripherals in a single domain, clocked at lower frequency. All MBUS additional peripherals, such as programmable co-processors or specialized accelerators, can be placed in a dedicated domain, allowing fast integration at their natural frequency, or use the MBUS domain. Moreover, the HBUS maximizes integration with DDR and HBM channels by deploying in their high-speed clock domain. Such a domain is available for accelerator deployment for the best performance and integration with the high-speed interconnect.

IV. EXPERIMENTAL VALIDATION

In this section, we empirically validate the capabilities of Simply-V for fast prototyping and research. We configure our SoC CDC with PBUS, HBUS and DRAM memories in dedicated clock domains and build our FPGA designs with Vivado v2024.2. We deploy Simply-V embedded profile on Digilent Nexys A7 Artix-7 board. For validation in HPC profile, we use an AMD Xilinx Alveo PCIe Card.

A. *Cross-vendor CPU Benchmarking*

We demonstrate a platform-fair comparison of RISC-V CPUs and the plug-and-play support of multiple processors from a diverse pool of vendors, namely CV32E40P from OpenHW, Ibex from lowRISC, MicroblazeV from AMD Xilinx, and finally, we demonstrate RV64 support with CVA6. Leveraging our configuration flow, for given a Simply-V setup we seamlessly plug in and out different CPUs. Additionally, we showcase the transparent profile transition from embedded to

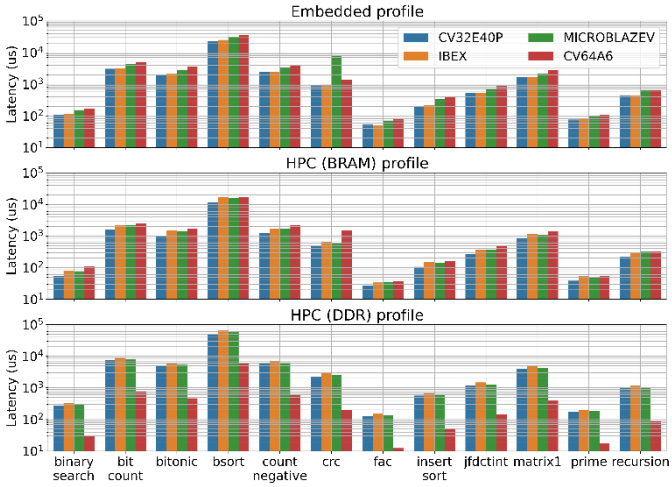

Fig. 4. Latency results of tacle-bench CPU benchmarking across Simply-V profiles and memory devices.

HPC setups, with no overhead from the practitioner. We run the tacle-bench1 benchmark on all CPUs and Simply-V profiles. For the embedded profile, we run software from onchip BRAM memory, in the main clock domain at 50MHz, alongside the CPU. For the HPC profile, we run the software both from BRAM memory and an external DDR bank.

Figure 4 shows the tacle-bench[1] latency results, averaged over 10 iterations. CPUs can be quickly evaluated and compared in the embedded profile, for its fast turn-around time, obtaining a fast, baseline indication with the sophisticated CVA6 core performing worse than simpler cores. Transitioning to a HPC configuration, namely and keeping code in local BRAMs or off-chip DDR, and increasing the target frequency in the main clock domain, a researcher can easily evaluate the differences in performance for the various cores.

B. *Fast Prototyping an HLS-based Convolutional Core*

In this section, we demonstrate the use of Simply-V as a hardware-ready platform for fast prototyping custom IPs, including support for technological heterogeneity with HLS technology. We target an 8-bit 2D convolutional engine, namely CONV2D, as a representative example of modern workloads. We implement a pool of CONV2D engines and integrate each one in Simply-V as an accelerator IP, demonstrating fastprototyping capabilities, both from the IP design and SoC integration perspective. Figure 5 reports the HLS engine's performance across design iterations: (1) *Naive loop-nest*: the baseline prototype of our core is a basic HLS-compatible Ccode, with a single AXI master port for memory access. (2) *AXI bursts*: activating memory coalescing; (3) *Double buffering*: implementing double buffering; (4) *Frequency boost*: leveraging our configuration-based CDC, we boost the IP clock frequency; Such approach improves performance by a only limited amount, suggesting the IP core is bottle-necked by memory accesses; (5) *Split interfaces*: maximizing data-access parallelism with three parallel read and write AXI ports; (6) *HBUS access*: alternatively, leveraging the HBUS interconnect for wider memory accesses, showcasing the best performance.

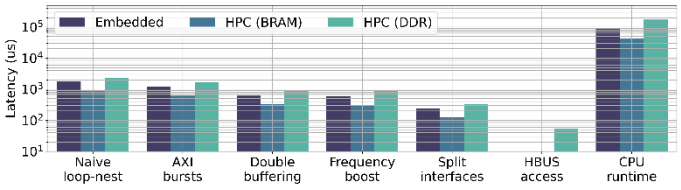

Fig. 5. Demonstrating fast prototyping with the latency evaluation of the multiple design iteration of HLS-CONV2D IP across Simply-V profiles.

V. Conclusions

In this work, we presented Simply-V, a reconfigurable, hardware-ready soft-SoC platform for fast prototyping and open hardware research. We demonstrated the capabilities of our platform by simplifying platform-fair CPU benchmarking and fast prototyping a HLS-based convolutional engine. Moving forward, we plan support for additional CPUs and IPs for RISC-V extensions and heterogeneous technologies, such as Chisel. We plan to soon boot Linux on Simply-V and deliver a full-fledged platform for experimental and applied research.

ACKNOWLEDGMENTS

This work has been partially supported by the Spoke 1 "FutureHPC & BigData" of ICSC - Centro Nazionale di Ricerca in High-Performance-Computing, Big Data and Quantum Computing, funded by European Union - NextGenerationEU.

REFERENCES

[1] M. Ciani et al., "Unleashing opentitan's potential: a silicon-ready embedded secure element for root of trust and cryptographic offloading," ACM Transactions on Embedded Computing Systems, 2024.

[2] P. Mantovani et al., "Agile SoC development with open ESP," in Proceedings of the 39th International Conference on Computer-Aided Design, ICCAD '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[3] A. Amid et al., "Chipyard: Integrated design, simulation, and implementation framework for custom socs," Ieee Micro, vol. 40, no. 4, pp. 10–21, 2020.

[4] N. Binkert et al., "The gem5 simulator," ACM SIGARCH computer architecture news, vol. 39, no. 2, pp. 1–7, 2011.

[5] N. Bruschi et al., "GVSoC: A Highly Configurable, Fast and Accurate Full-Platform Simulator for RISC-V based IoT Processors," in 2021 IEEE 39th International Conference on Computer Design (ICCD), pp. 409–416, 2021.

[6] S. Karandikar et al., "FireSim: FPGA-Accelerated Cycle-Exact Scale-Out System Simulation in the Public Cloud," in 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), pp. 29–42, IEEE, 2018.

[1] https://github.com/tacle/tacle-bench/tree/V1.9