D3.1: FPGA Emulation Platform

Document Properties

Contract Number	101092993
Contractual Deadline	M16 (April 30, 2024)
Dissemination Level	Public
Nature	Report
Edited by :	Jordi Cortina, Semidynamics
Authors	Jordi Cortina, Semidynamics Fabien Chaix, FORTH Bastian Lidner, EXTOLL Iakovos Mavroidis, EXAPSYS Filippo Mantovani, BSC
Reviewers	Olivier Déprez, SiPearl Mondrian Nüssle, EXTOLL Stelios Louloudakis, FORTH
Date	26/04/2024
Keywords	FPGA, Emulation Platform, EPAC
Status	Complete
Release	1.0

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement no. 101092993. The project is funded under the call on "Digital and emerging technologies for competitiveness and fit for the green deal"

History of Changes

Release	Date	Author, Organization	Description of Changes
0.1	07/02/2023	J.Cortina, Semidynamics	Initial Draft
0.2	17/04/2023	J. Cortina, Semidynamics	First internal version
0.3	18/04/2023	J. Cortina, Semidynamics	Bulk of content in place
0.4	19/04/2024	J. Cortina, Semidynamics	Added Interconnection diagram
0.5	24/04/2024	S. Louloudakis, FORTH	Incorporation of internal review comments
0.8	25/04/2024	J. Cortina, Semidynamics	Release candidate RC1
0.9	29/04/2024	S. Louloudakis, FORTH	Additional round of internal review
0.95	29/04/2024	M. Marazakis, FORTH	Release candidate RC2
1.0	30/04/2024	M. Marazakis, FORTH	Version for formal submission

Table of Contents

Table of Figures

List of Tables

1 Executive Summary

This report describes describes the results of Task 3.1. Specifically, the focus is on the FPGA Emulation platform that is leveraged by project partners to progress while the PCIe Accelerator and Microserver Platforms are being implemented. This is especially important for partners involved in the software-focused activities of WP4 and WP5. This emulation platform serves as a Software Development Vehicle (SDV). The core of the document resides in Section 3 (FPGA Emulation Platform). It is composed of three sub-sections:

- Section 3.2 (Hardware Setup) presents the hardware settings for the RISER FPGA platform
- Section 3.3 (FPGA Project) gives details of the platform firmware that emulates a RISC-V System-on-Chip
- Section 3.4 (Software) provides insights into the platform software stack from boot to applications.

Many of the partners involved in the development of the RISER FPGA Emulation platform are also partners in the EPI-SGA2 and EUPILOT projects and the RISER FPGA Emulation platforms use IP blocks that are developed from these partners. In order to provide a complete picture, this document highlights the elements that are common among projects and the novel aspects that have been developed in RISER T3.1.

The FPGA development platform used by other projects is the AMD/Xilinx VCU128 board. One of the main contributions of RISER has been to port the full SDV emulation infrastructure and software toolchain to Alveo U55C. This migration requires changes to the design to match the U55C board specifications (fewer logic resources, no external DDR4 RAM, no Gigabit Ethernet) but will allow for larger-scale deployment in the FPGA cluster available at BSC. Consequently, we anticipate expanding our FPGA emulation platform to tens of nodes in the latter period of the project.

In addition, the RISC-V core has been upgraded to Atrevido 323 in the RISER SDV. This core version is a significant improvement over the version present in previous projects:

- Out-of-Order core vs In-order
- Incorporates Semidynamics's new Gazzillion technology, supporting up to 128 outstanding misses.
- New branch predictor.
- Addition of Cache Management Operations (CMO) support.
- Upgraded floating-point unit, with half-precision arithmetic support.

Altogether, we expect that these changes to cover the needs of RISER partners with respect to emulation and prototyping in an effective manner.

2 Context

This document provides a description of the FPGA-based emulation platform featuring the RISER system, designed primarily to streamline hardware and software debugging processes. The system comprises of various key components, including the out-of-order RISC-V core sourced from Semidynamics, an DRAM memory controller tailored for the FPGA's memory, as well as UART and Ethernet connectivity for networking capabilities. These integral hardware elements enable the booting of Linux, thereby enhancing the platform's versatility. Moreover, the design incorporates embedded logic analysers within the FPGA logic, along with other essential IP blocks, to further facilitate efficient hardware and software debugging tasks. This testing design has been shared with RISER partners, serving as the designated Software Development Vehicle (SDV) for the advancement of system software within WP4, and for the development and exploration of application use cases within WP5.

3 FPGA Emulation Platform

3.1 Overview of Emulation Platform

This section describes the infrastructure developed for the RISER project to provide a test and validation platform, to be used as a vehicle allowing early software development and verification on a RISC-V runtime system environment. The intent of this early-generation platform is to resemble as much as possible the final RISER system boards so that the software developed in WP4 and WP5 can reach a high degree of maturity as early as possible, before the production of fully-featured platforms. The RISC-V ecosystem in the FPGA emulation platform leverages the RTL implementation of the chip that will be used in the final accelerator boards. The design includes several IP's developed within the EPI consortium by different partners interconnected in a manner that resembles as much as possible the final RISC-V chip that will be utilised on the RISER platforms. The emulation platform also includes additional logic supporting evaluation of performance characteristics as well as tracing capabilities for software debugging and optimisation.

3.2 Hardware Setup

3.2.1 Overview

The FPGA Emulation Platform shown in Figure 1 is integrated by an FPGA compute-based platform and an x86 Server. The host server is a commodity x86 server featuring an AMD Ryzen 5 5600 Central Processing Unit (CPU) with 128GB of DDR4-3200 memory, both mounted on a Mini-ITX motherboard. It runs with a Linux OS runtime environment based on the Red Hat mainstream distribution (Rocky Linux 9) with local storage and a mounted Network file-system. In terms of software, the server uses the Xilinx Vivado 2022.1 design suite installed to manage the programming and debugging of the FPGA evaluation board.

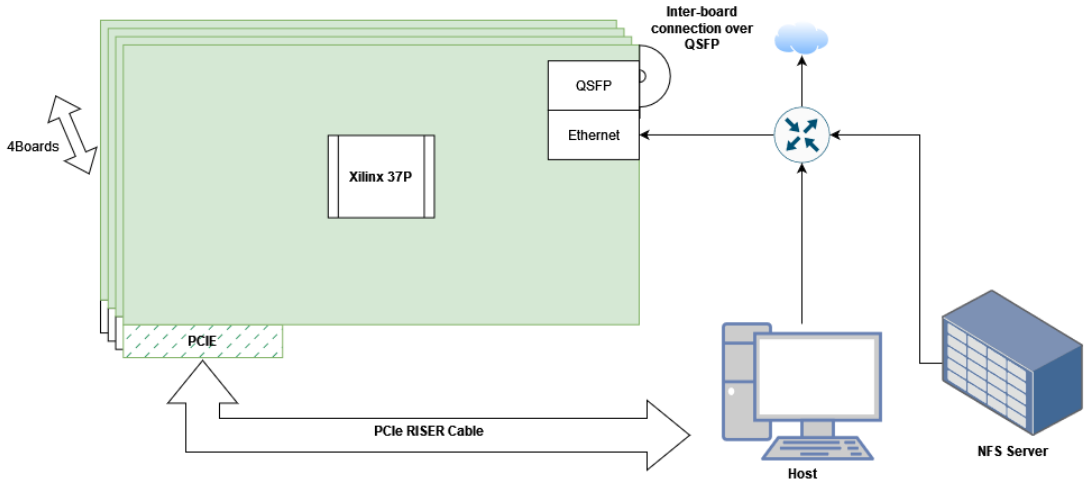

FIGURE 1. EMULATION PLATFORM HARDWARE SETUP DIAGRAM

3.2.2 FPGA Board

The FPGA boards used for the RISER emulation are the AMD VCU128 and the U55C, both featuring an AMD Virtex UltraScale+ VU37P HBM2 MPSoC and VU47P HBM2 MPSoC respectively. The UltraScale+ MPSoC integrates 8GB of HBM DRAM adjacent to an FPGA die, enabling substantial memory bandwidth and a significantly reduced PCB footprint. Virtex UltraScale+ HBM FPGAs mitigate bandwidth bottlenecks and power consumption associated with the use of parallel memories, such as DDR4, in various applications including compute, database, and network acceleration. The VCU128 board is specifically designed for prototyping applications utilizing Virtex UltraScale+ HBM FPGAs. The resources incorporated in the VCU128 and U55C are detailed in Table 1.

Resources	AMD VCU128	AMD U55C
Memory capacity	8 GB HBM2, 4.5 GB DDR4	16 GB HBM2
Memory throughput	460 GB/s	460 GB/s
LUTs	1,304K	1,304K
Registers	2,607K	2,607K
DSP Slices	9,024	9,024
High speed Communication	4 QSFP28, 1-Gbit Ethernet port, PCIe Gen3x16 or Gen4x8	2 QSFP28, PCIe Gen3x16 or Gen4x8

TABLE 1. FPGA SPECIFICATIONS

3.2.3 FPGA Resources availability

The Emulation Platform is based on FPGA devices, which are both expensive and difficult to exploit. Fortunately, members of the RISER consortium possess both a sufficient amount of those devices and the skill set to perform developments on the Emulation Platform and improve it. In particular, BSC provides remote access to a small prototyping infrastructure consisting of five VCU128 boards. In the forthcoming months of the project, BSC will extend access to a larger FPGA cluster for the RISER partners, enabling further developments on the FPGA Emulation platform. Other partners may also leverage local resources to support specific tasks. The number of FPGA boards available to RISER partners is presented in \hTable 2. As can be seen, there is an significant number of devices for RISER partners to work on, although those resources are shared with other projects.

Partner	Access level	Boards count [1]	
		Alveo U55C	AMD/Xilinx VCU128
BSC	Remote/All partners	96	5
Semidynamics	Local only	5	5
FORTH	Local only	4	3

TABLE 2: FPGA BOARDS AVAILABLE TO RISER PARTNERS

3.3 FPGA Project

3.3.1 Architecture Overview

The FPGA project is composed of two main blocks well differentiated by their origin: the RISC-V Compute Ecosystem and the 3rd Party AMD/Xilinx support IPs that make up the block design. The

[1] *This is the total number of boards hosted by partners at the time of writing, which is shared with other projects.*

RISC-V Compute ecosystem leverages the RTL from EPAC 1.5 (v1.5 of the RISC-V accelerator chip, from the EPI-SGA2 project) which incorporates several IP blocks developed by some of the partners present in the RISER project:

- Atrevido RISC-V Core, from **Semidynamics**.
- Vector Processing Unit (VPU) accelerator, from **BSC**.
- Network on Chip (NoC), from **EXTOLL**.
- Last Level Cache, developed by **FORTH**.
- Open source IPs: Debug Module, Platform-Level Interrupt Controller (PLIC) and Core-Level Interrupt Controller (CLINT) – originally developed for the Pulp platform[2].

The 3rd party IPs from AMD/Xilinx instantiated in the Block Diagram (BD) are a set of freely available designs tailored to the specific FPGA platform used, which provides the RTL design with connectivity peripherals such as PCIe, Ethernet, UART, and memory resources in the form of a controller, and an in-package HBM stack. In addition, the bitstream includes the required hardware to trace the execution of instructions (Tracer) and provide performance information during Atrevido cores' execution.

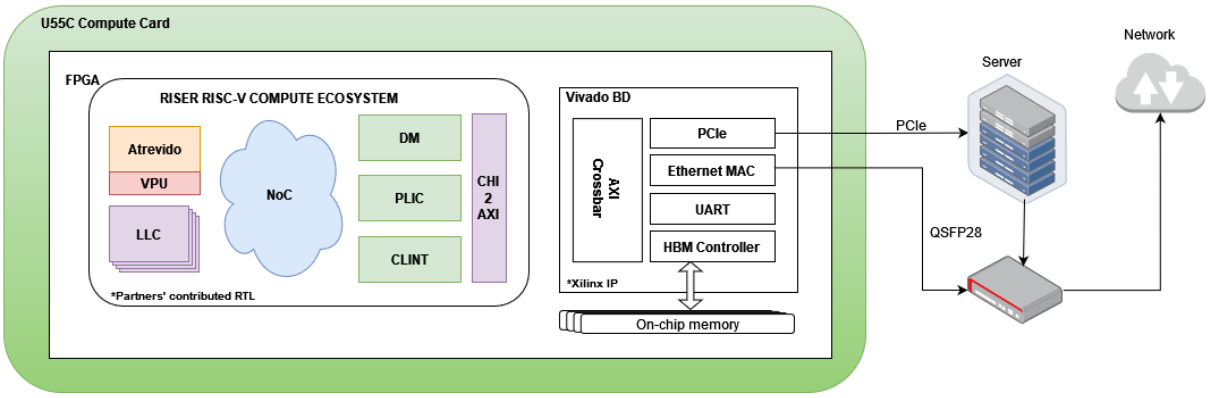

FIGURE 2. SDV ARCHITECTURE DIAGRAM

Figure 2 shows the connection scheme between the host server and the U55C device. The host and the FPGA are connected through three different interfaces:

- UART/JTAG interface: used to program the FPGA and also to access the UART terminal,once a Linux image is running on the RISC-V ecosystem in the FPGA.
- QSFP28 connector: establishes a point-to-point IP network between the host and the FPGA board. It is used to access the FPGA via SSH and give access to a network filesystem (NFS) to the Atrevido RISC-V core once Linux has booted.
- PCIe Interface: The host uses this interface to write the Linux image to the on-chip HBM memory. The configuration of the RISC-V ecosystem is performed via this interface.

3.3.2 RISC-V Core

The Atrevido core is presented in Figure 3, and is developed by Semidynamics. This is a 12-stage pipeline, 3-way-decode, 5-way-issue, 3-way-commit, 64-bit RISC-V core. Capable of targeting demanding HPC and AI tasks, Atrevido can also boot modern fully-featured operating systems such as Linux. ATV323 supports an optional native Vector Unit from Semidynamics, or can interface a 3rd

[2] https://pulp-platform.org/: *PULP is a silicon-proven platform organized in clusters of RISC-V cores. It consists of a set of IPs described in SystemVerilog, together with the related simulation and synthesis scripts, as well as the necessary runtime software written in C and RISC-V assembly to enable using the platform.*

party VPU over Semidynamics's OVI 2.0 (Open Vector Interface Specification). The later is the current implementation used in the RISER project, where the core interfaces the VPU from BSC. More information on the Vector Processing Unit can be found in the Section 3.3.3.

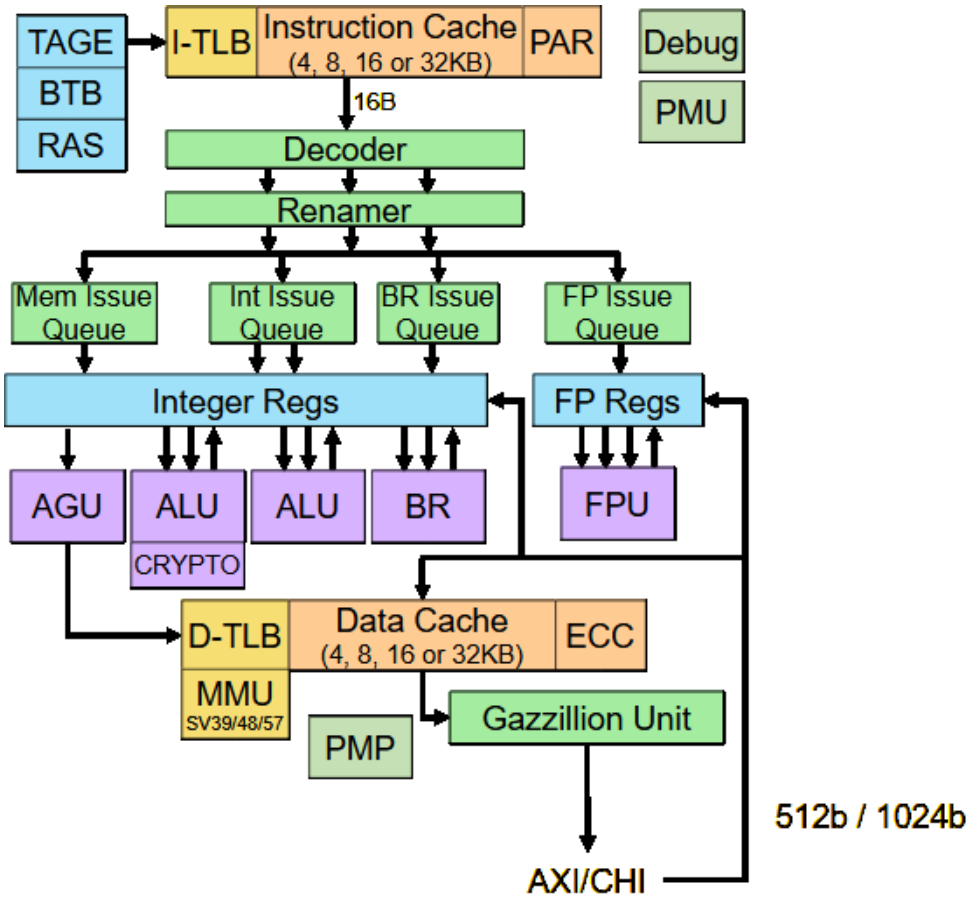

FIGURE 3. ATREVIDO 323 BLOCK DIAGRAM

The Atrevido 323 core supports three privilege levels: machine (M), user (U), and supervisor (S). Furthermore, both SV39 and SV48 virtual memory are supported. Table 3 also presents the list of RISC-V extensions supported by the core. Finally, the Atrevido 323 core supports the Semidynamics Gazzillion Misses ™ technology, allowing to support up to 128 outstanding misses; and adheres to the following specifications:

- RISC-V Instruction Set Manual, Volume I: Unprivileged ISA, version 20200125
- RISC-V Instruction Set Manual, Volume II: Privileged Architecture, v1.12
- RISC-V Debug Support v0.13

Extension name	Description of the extension
A	Atomic instructions
B	Bit Manipulation instructions (Zba, Zbb, Zbc, Zbs)
C	Compressed instructions
D	Double-Precision Floating Point instructions
F	Single-Precision Floating Point instructions
I	Base integer instruction set
M	Integer Multiplication and Division instructions

V	Vector instructions
Zicsr	Control and Status Register instructions
Zifencei	fence.i instruction
CMO	Cache management instructions (Zicbom, Zicbop, Zicboz)
G=IMAFD	Support for Linux OS

TABLE 3: RISC-V EXTENSIONS SUPPORTED BY ATREVIDO 323

3.3.3 Vector Processing Unit (VPU)

The EVA (Enhanced Vitruvius Architecture) VPU is an advanced design of the Vitruvius vector unit featured in EPAC-1.0 and EPAC-1.5. Compared to the previous implementation, it adopts the latest RISC-V V-extension (version 1.0). It also connects with the Atrevido scalar core via an updated Open Vector Interface (OVI), now at version 2.0. Atrevido manages memory access for vector memory operations. EVA is designed for HPC acceleration, featuring long vector registers (16384 bits) and a flexible number of functional units (8 in the current implementation), all linked by a dual ring interconnect. The EVA Vector Processing Unit (VPU) has the following general characteristics:

- Maximum vector length (VLEN): 16,384 bits, seen as 256 elements of 64 bits, 512 elements of 32 bits, 1024 elements of 16 bits, or 2048 elements of 8 bits.
- 1 Fused Multiply-Add (FMA) unit per lane capable of 2 DP FLOP/cycle.
- Support for 64- and 32-bit FP operations per lane capable of 4 SP FLOP/cycle.
- Support for 64, 32, 16, 8-bit integer and fixed-point operations, signed and unsigned.
- Support for vector register renaming with 40 physical vector registers per core.
- Limited out of order capability in vector memory operations.
- Overlapped execution of arithmetic, memory, and data movement operations.
- Support for any number of vector loads as provided by the OVI-2.0 interface.
- Support for masked operations, according to the mask layout in RVV-1.0.
- Support for Vector Register Group Multiplier (aka LMUL) LMUL>1 and fractional LMUL.

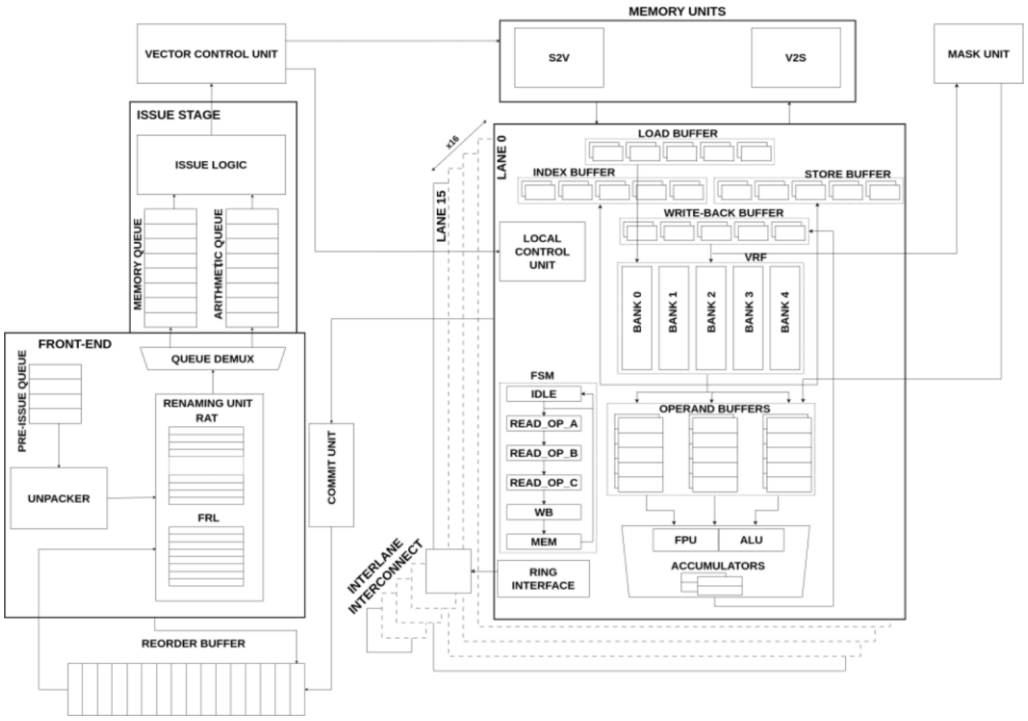

FIGURE 4. EVA ARCHITECTURE BLOCK DIAGRAM

Figure 4 illustrates the comprehensive block diagram for the EVA VPU. This RISC-V vector accelerator operates on a decoupled design principle, wherein the Atrevido out-of-order scalar core dispatches instructions to the VPU. This set-up enables the VPU to process instructions with a degree of out-of-order execution capability. The architecture of EVA is organized around lanes, each of which is uniformly configured. Every lane houses a segment of the vector register file and is paired with a Fused Multiply Accumulate (FMA) functional unit. Additionally, each lane is equipped with its own Finite State Machine (FSM), granting it the autonomy to operate independently of the others. This independence is maintained except in scenarios where executing specific instructions—such as those involving permutation and data movement—requires access to vector elements from different lanes. In the RISER Emulation platform implementation, the VPU incorporates a total of 8 lanes.

Finally, the VPU is supported by the LLVM C compiler (originating from the EPI-SGA2 project) which grants users a convenient way to generate vectorized code that leverages this high-performance computing IP.

3.3.4 Last-Level Cache (LLC)

The Last Level Cache (LLC) developed by FORTH originates from the EPAC-1.5 design and has been adapted for the RISER FPGA emulation platform. The LLC consists of multiple banks distributed on different cross-points of the NoC. Each bank of the LLC is 256 KBytes, 8-way set associative and has an optimized seven-stage pipeline. In this RISER system, this is the second level cache which is also the last level of cache memory, hence the name Last Level Cache (LLC). The pipeline of the Last Level Cache is shown in Figure 5 and has the following characteristics:

- AMBA5 CHI and AXI Protocol
- Multiple distributed banks (up-to 4 banks).
- 256KB 8-way set-associative per bank with Pseudo-LRU replacement
- Fully pipelined and non-blocking design: 1 line (64-bytes) per cycle per bank.
- Supports cache allocation hints by core and vector unit (non-temporal loads and stores).
- Programmable address interleaving: 64-bytes / 2KBytes / 4KBytes (OS page).
- Point-of-Serialisation (PoS) and support for atomic operations
- Cache Management Operations (CMOs)
- 64 outstanding misses per bank.
- 64 outstanding write-backs or non-temporal stores per bank.

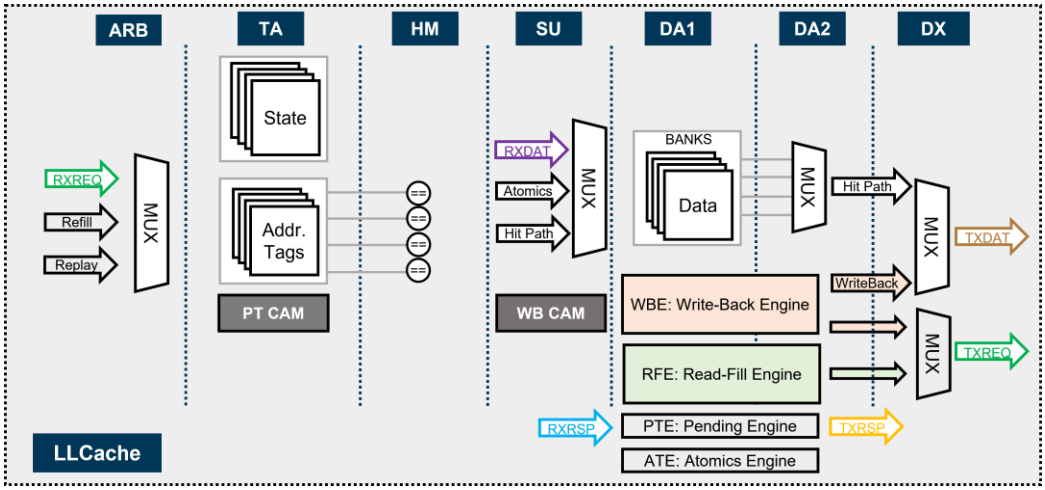

FIGURE 5. THE LAST LEVEL CACHE PIPELINE

3.3.5 Network on Chip (NoC)

The NoC used in the Emulation Platform has been re-used from the EPAC-1.5 test chip and adapted for the particular requirements of the SDV. Two different kind of NoCs are utilized: (1) a CHI based cache coherent high-performance NoC enables accessing data from main memory and data exchange between the accelerator cores and the LLC, and (2) a NoC based on AXI lite, for configuration and debugging purposes. The topologies for both NoCs are shown in Figure 6.

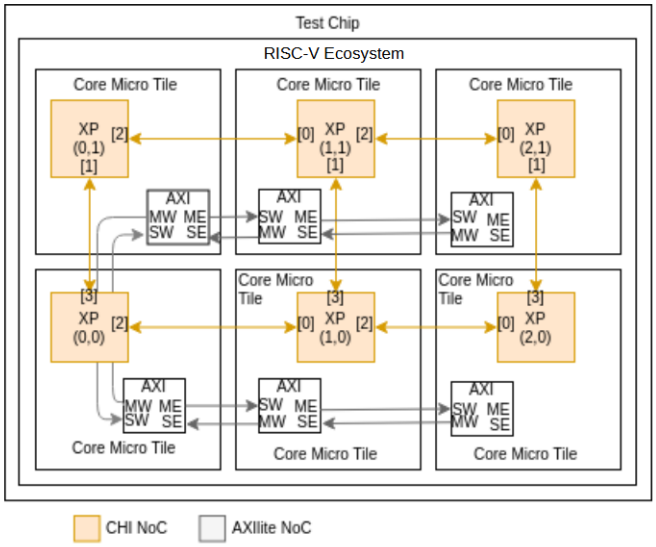

FIGURE 6. RISC-V ECOSYSTEM NOC

CHI NoC

The CHI high-performance NoC developed by EXTOLL is the main on-chip interconnection network. It transfers data from main memory into the last level caches, from the last level caches into the cores and between the cores. The cache-coherent network supports 4 different channels: request, response, data and snoop. These channels are realized as distinguished channels to provide the needed bandwidth and throughput. Each endpoint like cores or LLCs can insert one request on each channel in each clock cycle. A complete cache line of 512 bits can be injected into the network every clock cycle. For example, running at a frequency of 1GHz, the NoC provides an injecting bandwidth of 64 GB/s per port on the data channel.

The basic building block of the NoC is the Cross Point (XP). Each XP includes four links to connect the XPs with each other and two ports to provide access for devices like the different cores or the LLCs to the network. The XPs themselves can be connected in any topology that can be built with four links. For EPAC-1.5, a 2D mesh configuration was chosen, which is a known scalable topology. It also fits for the physical implementation of the test chip, as the micro tiles can be laid out in a grid. The EPAC-1.5 test chip consists of 6 micro tiles. Therefore, the mesh uses a 3x2 configuration, which is also reflected in the physical implementation. The topology is shown in Figure 6. RISC-V Ecosystem NoC . Dimension order is used as routing algorithm for this network. It provides a deadlock-free routing in a 2D mesh. It also retains the order of injected packets on each channel, which is mandatory requirement by the CHI specification. All channels besides the snoop channel support only unicast routing. For the snoop channel multicast routing is supported. The same snoop request can be sent to more than one destination node. If the injecting device sends this request multiple times for each destination, then the network utilization gets increased. Therefore, the NoC implements multicast support for the snoop channel. Each XP has a special snoop routing table to replicate requests in a XP when they need to be sent to different egress links of the XP to reach their destinations. A credit-based flow control is used between the XPs.

AXIlite configuration NoC

A separate NoC is used for low bandwidth access to initialize and configure the components of the test chip. That second NoC adheres to the AMBA AXI4-Lite standard and is based on open-source HW IP modules from the Pulp platform. Each tile contains a crossbar switch with three master and three slave ports. Two master and two slave ports connect to other tiles; one master and one slave port connect into the tile. The bottom three tiles and the top three tiles are connected with each other, and the bottom half and the top half are connected between the IO and the upper cross-point tile. External accesses to components on the config NoC are possible through the RISC-V debug module in the IO tile, which is also attached to the config NoC.

3.3.6 Other IPs

Core-Local Interrupt Controller (CLINT)

This IP is responsible for handling interrupts at the core level. In particular, it deals with timer and software interrupts, and maintains the memory-mapped control and status registers which are associated with those functions. The RISER FPGA emulation platform uses the IP from the open-source PULP platform.

Platform-Level Interrupt Controller (PLIC)

This IP is responsible for handling hardware interrupt sources that are shared amongst the hardware threads (harts) of the RISC-V processor. Typically, it exposes an interface for I/O devices to raise interrupt signals and for harts to divert from the normal program flow to execute interruption handling code. The RISER FPGA emulation platform uses this IP from the open-source PULP platform.

Debug Module

In order to bring up a System-on-Chip and develop applications effectively, it is essential that processors provide means to the user to pause the execution of programs and probe the internal state of the system and the application memory. Together with an adequate software stack, the Debug Module allows developer to perform those actions. The RISER FPGA emulation platform uses this IP from the open-source PULP platform.

CHI to AXI Interface

An IP block is needed to interface and bridge the CHI NOC presented in Section 3.3.5 and AXI peripherals such as the Ethernet and the PCIe controllers. This IP has been developed by FORTH.

PCIe end-point controller

Both the AMD/Xilinx VCU128 and the Alveo U55C can be plugged into a PCIe port of a host PC. This is very useful to implement high-speed communications between applications running on the host machine and the RISC-V SoC. In particular, we use it to load the OS image and other binaries, as described in the Section 3.4 (Software). On the side of the FPGA, we use a PCIe controller IP provided by AMD/Xilinx configured as XDMA with 8 lanes of PCIe Gen.2. This setup is meant to perform system initialization and not high-performance transfers.

Ethernet controller

Ethernet connectivity is implemented by means of an Ethernet controller IP in the FPGA. Due to specifics of the FPGA boards, we use several solutions. For AMD/Xilinx VCU128 boards, a 1 Gigabit Ethernet controller IP from AMD/Xilinx is used. For Alveo U55C, a 10 Gigabit Ethernet controller IP is used instead.

External memory latency and throughput control

To assess the impact of increased memory latency, a Latency Controller IP block developed by FORTH is integrated on the memory just before the DRAM memory controller(s), being HBM or DDR4. It exposes configuration registers through an AXI4-lite interface, allowing users to configure the desired additional latency. In addition, a Bandwidth Limiter IP block developed by FORTH is used to throttle bandwidth and throughput, enabling the study of system behaviour and performance under different memory system bandwidth thresholds.

GPIO

The FPGA Emulation platform contains a GPIO IP with several outputs connected to the Atrevido reset, kick core and tracer signals. This device is mapped to the platform memory, and the outputs can be controlled by plain writes to the GPIO slave over XDMA.

3.3.7 Emulation Platform changes related to EPAC-1.5 TC

Focusing on creating an FPGA Emulation Platform that can be used for system software development, as well as vector code development and overall system benchmarking, we have identified the relevant modules. The EPAC-1.5 RTL design, has been patched and modified accordingly for FPGA porting and optimization. Some key changes are listed below:

- Porting of Atrevido, LLCs and VPU's data structures from SRAMs to AMD/Xilinx-specific primitives such as URAMs and BRAMs to save FPGA resources.
- Upgraded Atrevido core.
- Upgraded VPU.
- Adapted the L2Cache to function as LLC.
- Porting of some of VPU's arithmetic operations to DSPs hardware block.
- Bridging of the EPAC-1.5 code with the 8GB HBM memory of the FPGA, using custom glue logic.
- Addition of tracing mechanism developed by Semidynamics that enables co-emulation using Hardware-In-the-Loop (HIL) techniques
- Addition of auxiliary signals that can be monitored via Integrated Logic Analyzers (ILA) to enable the system's monitoring/benchmarking under real workloads.
- The FPGA designs and projects have been upgraded to the newer AMD/Xilinx Vivado 2022.1.
- The Linux kernel and OpenSBI of the RISER SDV have been upgraded to Linux kernel 6.x series in order to closely track the mainline Linux kernel to have the most up-to-date RISC-V support that improves system stability – the current RISER SDV Linux kernel version is 6.6 LTS. Moreover, the RISER SDV now supports two full-scale Linux distributions such as Ubuntu 22.04 LTS and Fedora that help the users have access to the latest packages via their package manager and easily install the dependent libraries for their applications.The RISER SDV also supports a minimal testing-oriented Linux environment, based on BusyBox.

3.4 Software

3.4.1 Platform initialization sequence

The sequence of steps required to initialize the Emulation Platform (cf. Figure 7) are as follows:

1. Programming the FPGA with the platform bitstream over USB.
2. Load the XDMA driver to enable PCIe communication with the FPGA.
3. Release system reset over PCIe.
4. Load executable binaries to the FPGA's DDR4 or HBM memory over PCIe.
5. Configure the control registers of the RISC-V ecosystem design over PCIe.
6. Enable the kick config bit to start the core fetch from memory.

7. Mount NFS folder with applications.

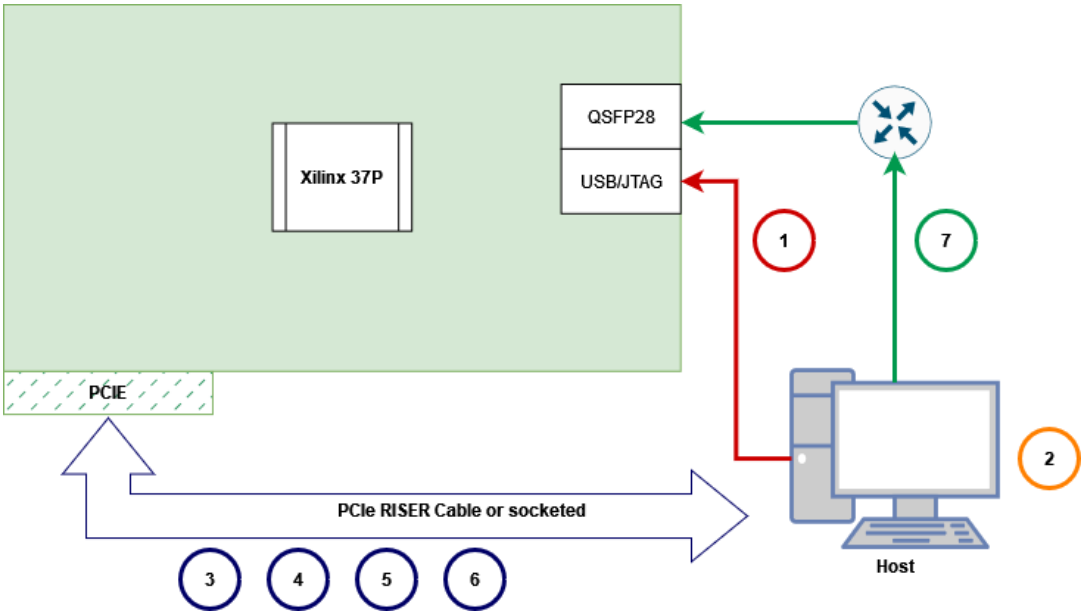

FIGURE 7. SYSTEM INTERCONNECTION DIAGRAM

3.4.2 Linux Boot

Once the platform is out of reset, the boot of the Operating System begins. As is typical for the Linux OS, the RISER FPGA emulation platform uses the following artefacts:

- A **BootROM** binary performs minimal low-level initialization of the machine.
- **OpenSBI** first configures the hardware as a First Stage Boot Loader, and then prepares Linux binaries for execution.
- A **device tree** binary blob provides platform-specific information to the OS.
- The **Linux kernel** then takes over and performs the initial configuration of the OS
- A **root file system** is then loaded by the kernel. It contains additional OS features (i.e. kernel modules and system services) as well as user applications.

Creating those artefacts for the FPGA Emulation platform from scratch consists of multiple complex steps, in particular since the multiple toolchains required to build them are usually not present in the user's machine. It is also inconvenient that those artefacts may only be validated on the FPGA emulation platform, and not directly on the user's PC.

To that end, yet another RISC-V Tool (YARVT)[3] has been developed. This tool will be described in detail in the RISER Deliverable 2.3, to be delivered at M17. In a nutshell, this tool automates most of the artefacts generation. First, compilation toolchains for RISC-V for both bare-metal and Linux are fetched and built. Second, the artefacts described above are built (except the BootROM and device tree). Third, those artefacts are aggregated into a single binary image, which simplifies the handling of the system. Finally, users may also use YARVT to build a QEMU binary that mimics the FPGA Emulation platform. Equipped with those binaries, users may boot the platform using the procedure described in

[3] *Open-source build tool, available at https://github.com/mickflemm/yarvt*

Section 3.4.1 (Platform initialization sequence); or emulate this boot directly on their machines, using the corresponding QEMU image.

The device tree and the BootROM are generated independently by firmware designers and system software developers for different reasons. The BootROM does not evolve much and is different depending on the use case (e.g. QEMU or FPGA). The device tree is modified very often and it would therefore be inconvenient to regenerate the whole boot image each time.

3.4.3 Running Applications

Job scheduling

There are currently a limited amount of SDV nodes, each one composed of one x86 server (host) and one FPGA development board. Figure 8 represents the network infrastructure presented to the user: first a login to a "login node" is required and then, through SLURM commands (operating on the blue area of the network), the users can allocate FPGA resources.

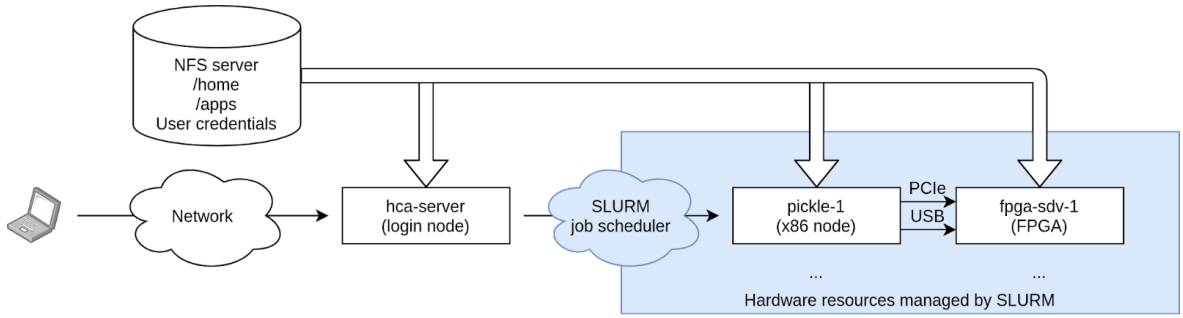

FIGURE 8. JOB SCHEDULING INFRASTRUCTURE DIAGRAM

Due to the limited hardware resources, access to SDV nodes is given through the SLURM job scheduler. Both interactive sessions, via the salloc command, and jobscripts, via the sbatch command, are supported. Additionally, users may optionally select a version of the RTL by passing an extra SLURM parameter known as constraint. A custom script called *prologue* runs at the beginning and end of a SLURM job.

```
$ salloc -p fpga-sdv -N1 -t1-00:00:00
salloc: Granted job allocation <job_id>
salloc: Waiting for resource configuration
// Wait for ~3min
salloc: Nodes pickle-1 are ready for job
$ ssh fpga-sdv-1
fpga-sdv-1$
```

```
$ cat jobscript.sh
#SBATCH --partition=fpga-sdv
#SBATCH --nodes=1
#SBATCH --time=1:00:00


ssh fpga-sdv-1 "<myscript>"
$ sbatch jobscript.sh
```

The prologue script starts by parsing the constraint given by the user to select the corresponding bitstream. This bitstream is then programmed into the FPGA using the Vivado toolchain. Afterwards, the script offloads a Linux kernel image to the FPGA through the PCIe bus and triggers the Linux boot as explained in the previous sections. Lastly, the prologue script performs some functional checks before handing the node to the user: if the Linux image running inside the FPGA has reached the login prompt, if it responds to ping and SSH connections, etc.

For security and IP protection reasons, the prologue script also removes access to the PCIe and USB buses. In this manner, malicious users may not reprogram the FPGA with a different bitstream nor offload a non-curated Linux image.

The whole prologue script takes up to three minutes to execute. To minimize waiting time, if the constraint requested by the user is the same as the one from the previous job, and the Linux inside the FPGA is still alive (i.e., responds to ping), there is no need to reprogram the FPGA and boot the OS.

The Linux image that is booted inside the FPGA is a standard Ubuntu 22.04 LTS distribution, which mounts its root filesystem and /home from an NFS server (Fedora support has also been developed by Semidynamics). This allows users to have access to the SDV node using the same credentials as in the login node and have an implicit file sharing mechanism. Furthermore, system software such as compilers and mathematics libraries are also mounted from the NFS server, and are shared with other commercial RISC-V platforms of the datacenter. In this manner, users have a binary-compatible environment in whichever RISC-V partition they are running on.

3.4.4 Tools and Metrics

In addition to execution of RISC-V binaries, the FPGA Emulation platform offers several capabilities that are important for software partners to assess the merits of the considered hardware/software combination.

Memory latency & throughput modifications

Interactions between executed software and external memory are critical for high performance applications. In the RISER FPGA emulation platform, users have the possibility to manipulate both the maximum throughput and the minimum latency to the external memory. In effect, IPs presented in Section 3.3.6 (Other IPs) practically introduce a fixed latency to each request and throttle them, if users configure them so. In practice, users may set those parameters from the PC that hosts the FPGA, before booting the SDV. Additional latency can be set at the clock cycle granularity. Memory bandwidth throttling is determined as the fraction of cycles during a given time window, in which memory accesses are completed. The resulting bandwidth can be calculated by multiplying this fraction with the data width and the FPGA frequency.

Tracing capabilities

The SDV also provides a co-emulation functionality, incorporating a SemiDynamics implemented tracing mechanism, capable of capturing vital information about the code executed by the Atrevido core in the FPGA. This tracing mechanism is integrated into the RTL code through a patch, and during code execution, it transfers the captured trace to the FPGA's HBM. To facilitate this process, a host application is deployed, responsible for copying the captured trace from the FPGA to the host memory and regulating the FPGA data flow to ensure lossless copies of the trace.

Concurrently, Spike, a RISC-V Instruction Set Simulator (ISS), runs on the host machine, and equivalence checking is performed between the traces generated by Spike and Atrevido. Upon detecting a mismatch, an extended report detailing the discrepancy, along with information about the executed code leading up to the deviation, is generated. The tracing infrastructure is depicted in Figure 9.

Leveraging this infrastructure, hardware developers can formulate comprehensive lists of RTL signals and triggers, aiding in the comprehension and resolution of RTL issues.

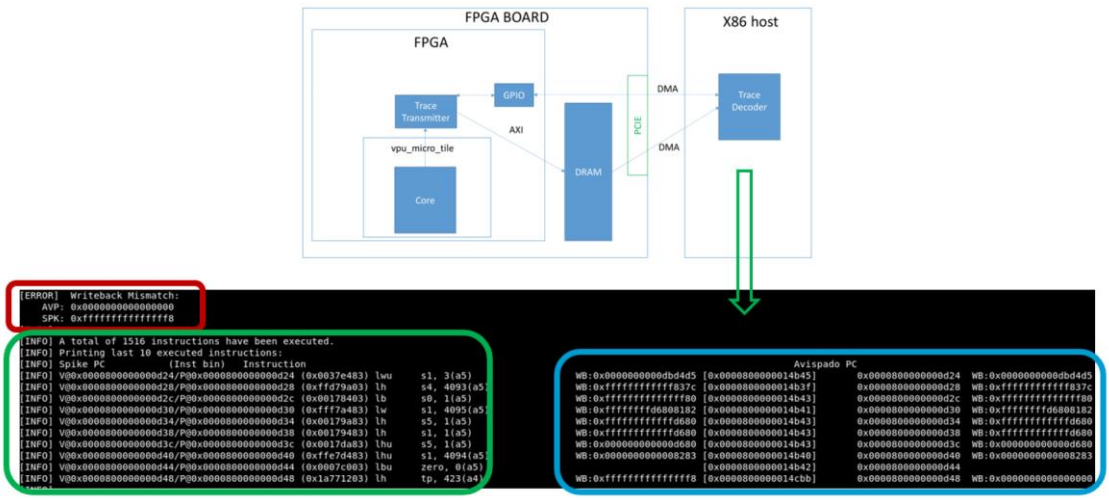

FIGURE 9. TRACER OUTPUT CAPTURE

4 Concluding Remarks

The work described in this document has led to the development of an FPGA-based emulation infrastructure for RISER system platforms (Accelerator and Microserver). This infrastructure enables activities in the software-focused packages (WP4 and WP5) to proceed with development and testing of software stacks and use cases well before the RISER system boards become available. The Emulation Platform (SDV – Software Development Vehicle) builds upon the foundation created earlier by the ongoing EPI-SGA2 and EUPILOT projects. This collaborative effort across projects is crucial as it expands the prototyping infrastructure beyond what a single project could achieve alone. Additionally, it allows for sharing and reuse of expensive FPGA boards across projects.

Within WP3, significant improvements have been made to the FPGA emulation infrastructure by upgrading some of its key components, such as the RISC-V Core and the VPU, to their latest available versions, thereby enhancing the platform's performance. Furthermore, enhancements have been made to the supported FPGA boards: the current SDV is now compatible with the Xilinx U55C platform, in addition to the already supported VCU128. Additionally, the OS support has been expanded to include two fully-featured Linux distributions, Fedora and Ubuntu, for deployment in Cloud/HPC environments.

Future work for this SDV, beyond ongoing limited-scope effort focused on maintenance and usability enhancements, includes expansion to a larger cluster of U55C boards hosted at BSC, which will enable providing more resources to the activities focused on system software and use cases. This expansion of the emulation infrastructure is expected to further stress-test the robustness of RISER system designs.

This deliverable precedes D2.3 (due by M17, i.e. end of May 2024), which will present the suite of tests and tools used in RISER for platform verification and evaluation. Taken together, D3.1 and D2.3 provide the basis for upcoming deliverable D3.2 (due by M22, i.e. end of October 2024), which is the planned initial release of the RISER Acceleration and Microserver Platforms. The FPGA-based emulation platform described in the present document is an essential pre-requisite for evolving the designs for the upcoming RISER platforms. With the delivery of D2.3, the RISER project will have reached Milestone MS2 (“FPGA emulation completed and verified against specifications”).

5 Appendix: Acronyms and Abbreviations

Term	Definition
AMBA	Advanced Microcontroller Bus Architecture
AMBA-CHI	AMBA-Coherent Hub Interface
AXI	Advanced eXtensible Interface
BRAM	Block RAM
C2C	Chip-to-Chip
CLINT	Core-Local Interrupt Controller
CMO	Cache Management Operations
CPU	Central Processing Unit
DDR	Double Data-Rate
DRAM	Dynamic Random Access Memory
EPAC	EPI Accelerator (collection of IP blocks, incl. Core, Cache, NoC, VPU)
EPI	European Processor Initiative
FLOP	Floating Point Operations per Second
FMA	Fused Multiply Accumulate (FMA)
FPGA	Field Programmable Gate Array
HBM	High Bandwidth Memory
HIL	Hardware-In-the-Loop
HPC	High Performance Computing
ILA	Integrated Logic Analysers
IP	Intellectual Property block (hardware and/or software artifact)
ISA	Instructions Set Architecture
JTAG	Joint Test Action Group
LLC	Last-Level Cache
LMUL	Vector Register Group Multiplier
LPDDR	Low Power Double Data Rate
LRU	Least Recently Used
MPSoC	Multi-Processor System-on-Chip
NFS	Network File System
NoC	Network on Chip
NVMe	Non-Volatile Memory express
OpenSBI	Open Supervisor Binary Interface
OS	Operating System
OVI	Open Vector Interface
PCIe	Peripheral Component Interconnect express
PLIC	Platform-Level Interrupt Controller
RTL	Register Transfer Level
SDV	Software Development Vehicle
SRAM	Static Random Access Memory
SSH	Secure Shell Protocol
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
VPU	Vector Processing Unit
XP	Cross-Point