IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 21, 2022, accepted April 30, 2022, date of publication May 10, 2022, date of current version May 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3174125

A Survey of the RISC-V Architecture Software Support

BENJAMIN W. MEZGER[1], (Student Member, IEEE), DOUGLAS A. SANTOS[1,2], (Student Member, IEEE), LUIGI DILILLO[2], (Member, IEEE), CESAR A. ZEFERINO[1], (Member, IEEE), AND DOUGLAS R. MELO[1], (Member, IEEE)

[1]Laboratory of Embedded and Distributed Systems, University of Vale do Itajaí, Itajaí 88302-901, Brazil
[2]Laboratoire d'Informatique, de Robotique et de Microelectronique de Montpellier, University of Montpellier, 34095 Montpellier, France

Corresponding authors: Benjamin W. Mezger (ben@edu.univali.br) and Douglas R. Melo (drm@univali.br)

This work was supported in part by the Foundation for Support of Research and Innovation, Santa Catarina, under Grant FAPESC-2021TR001907; in part by the Brazilian National Council for Scientific and Technological Development under Contract 313.513/2021-0; and in part by the University of Montpellier and La Région Occitanie under Contract 20007368/ALDOCT-000932.

ABSTRACT RISC-V is a novel open instruction set architecture that supports multiple platforms while maintaining simplicity and reliability. Despite its novelty, the software support for RISC-V has been increasing in the last years, given that popular toolchains and operating systems already have support for RISC-V. However, although many works have been exploring the RISC-V software ecosystem, no work that raised the current state of software support for RISC-V is available. In this context, this survey reviews the contributions introduced in the last years to understand the RISC-V's software ecosystem and its usage in both academic and industrial environments. We classified and evaluated the works into four main categories: application fields, RISC-V implementations, software architecture, and deployment features. The primary goal of this research is to provide the community with a comprehensive overview of the current state of RISC-V software support and identify and highlight the main contributions from recent work.

INDEX TERMS RISC-V, software support, operating systems.

I. INTRODUCTION

With the increasing number of instructions of popular Instruction Set Architectures (Instruction Set Architecture (ISA)s) and the requirements of backward compatibility of older extensions, researchers from the University of California at Berkeley developed an open ISA based on Reduced Instruction Set Computer (RISC) principles. This architecture was named RISC-V [1] and seeks to provide a base ISA and optional application-specific extensions to support software engineers with a small and robust ISA [2]. In other words, it offers a stable ISA for compiler and operating system designers, enabling them to work with hardware architects to provide additional resources to meet application requirements and participate in the decisions and implementations of the RISC-V ISA specifications [2]. Nowadays, RISC-V is maintained by RISC-V International, a nonprofit organization [3].

RISC-V aims at becoming a universal ISA by supporting numerous processor sizes, from embedded controllers to high-performance computers, and a wide variety of software stacks and programming languages [2]. Moreover, given the future computing landscape, multiple platforms such as Internet-of-Things (IoT) devices and personal mobiles will likely dominate the market, requiring ISAs to support these systems [4].

Over the last decade, several studies have focused on the different issues related to the RISC-V architecture, and part of these contributions is summarized and analyzed in survey papers focused on security [5]–[8] and open-sourcing [9], [10] aspects. Although many studies have addressed the software stack for RISC-V, there is no report on the literature analyzing these studies from a unified perspective. Driven by the growth of the RISC-V ecosystem and the lack of studies concerning the current state of software for this ISA, this work seeks to fill this gap and explore the state-of-the-art software support for RISC-V architectures. This survey analyzes works carried out in the last five years to understand the RISC-V's software ecosystem and its usage in academic and industrial environments. The main contributions of this research are: (i) providing the community with a

The associate editor coordinating the review of this manuscript and approving it for publication was Liang-Bi Chen.

comprehensive overview of the current state of the RISC-V software ecosystem, and (ii) identifying and highlighting the main contributions of recent work.

The remainder of this work is organized as follows. Section II summarizes the RISC-V ISA and Section III describes the methods applied to conduct this review. Next, Sections IV and V cover application fields and the RISC-V implementations, respectively. Finally, Sections VI and VII cover a set of software support characteristics and deployment features. Concluding, Section VIII presents the final remarks.

II. RISC-V

Unlike prior ISAs, RISC-V relies on a modular design by providing a frozen base ISA core (RV32I), along with extensions that provide additional functionalities. Moreover, it supports a full software stack, providing a stable target to compiler writers, operating system developers, and assembly language programmers [2]. A modular design enables optional standard extensions that the hardware can include if specified by the system's requirements, facilitating the development of small or large-scale applications and compilers to generate more reliable code. Besides, when RISC-V needs to include new instructions to the ISA, they are kept optional and non-required for all future RISC-V implementations, contrary to incremental ISAs [2].

A. THE BASE ISA

The RISC-V ISA provides four primary base integer (I) variants: RV32I (for 32-bit), RV32E (for 32-bit with 16 registers), RV64I (for 64-bit), and RV128I (for 128-bit). The base instruction sets provide the minimum requirements for running a processor and can run an operating system. It uses a two's-complement representation for signed integer values, and data is stored in memory using the little-endian system but allows non-standard alternatives to provide a big-endian memory system [1].

Besides the base ISAs, the RISC-V specification provides further functionalities through extension modules, such as M, for multiplication and division instructions; A, for atomic instructions; F, for floating-point instructions; D, for double-precision floating-point instructions; G, comprising the group of previously mentioned extensions; Q, for quad-precision floating-point instructions; L, for decimal floating-point instructions; C, for compressed instructions; B, for bit manipulation instructions; J, for dynamically translated languages support; T, for transactional memory support; P, for packed-Single Instruction Multiple Data (SIMD) (Single Instruction/Multiple Data) instructions; V, for vector operation instructions; N, for user-level interrupt support; H, for hypervisor support; and S, for supervisor-level instructions [3].

B. INSTRUCTION FORMAT

The RV32I base ISA has four instruction formats (R/I/S/U) and two variants (B/J) based on the handling of immediate operands. All of them have a fixed length of 32 bits and must be aligned on a four-byte boundary in memory [1]. Thus, the six instruction formats are:

- R-type, for register-register instructions.
- I-type, for register-immediate and load instructions.
- S-type, for store instructions.
- U-type, for instructions with a large upper immediate.
- B-type, for conditional branch instructions.
- J-type, for unconditional jump instructions.

Given that the six instruction formats have regular encoding, the decoding of instructions is much more straightforward than ARM or x86 architectures. For example, RISC-V provides three register operands at the same position in all formats, simplifying the decoding process. In addition, the specified registers to be read or written are always in the same position in all instructions, enabling register access to start before the instruction decoding phase [1], [2].

C. PRIVILEGE LEVEL

The hardware must provide the Operating System (OS) mechanisms that enable the processor to change its execution privilege status, such as going from the user mode to the supervisor mode, and provide protection across different software components [11]. RISC-V supports three privilege levels of execution to protect different software stacks, enabling multiple software stacks to run with different privileges levels. Attempts to perform operations not permitted by the current privilege level will result in exceptions and require handling by an underlying execution environment. The highest and mandatory privilege level of the RISC-V hardware platform is the Machine-Mode (M-Mode), which is inherently trusted and has all low-level access to the system. Besides M-Mode, RISC-V supports Supervisor-Mode (S-Mode) intended for OS usage, and User-Mode (U-Mode) for conventional applications. In addition, each privilege level has a set of privileged ISA extensions with support for optional extensions. This extension enables, for instance, having the S-Mode extended to support a hypervisor execution environment [12].

D. REGISTERS

RISC-V has 32 registers (`x0-x31`), whose names are determined by the RISC-V's Application Binary Interface (ABI). Register `zero` is hardwired and always holds the value zero, mainly to simplify the ISA. The `ra` and `sp` registers hold the return address and stack pointer, respectively. Registers `t0-t6` hold temporary values that are not guaranteed to persist after a function call, and `s0-s11` hold persistent values across function calls. Finally, registers `a0-a1` hold the first two arguments of a function and return value, and `a2-a7` hold any remaining arguments [2].

E. CONTROL AND STATUS REGISTER

The Control and Status Registers (CSRs) are system registers to control and monitor the machine's current state.

CSRs can be read or written through specific Control and Status Register (CSR) instructions, and have restrictions for low-privilege levels. A RISC-V implementation may contain additional CSRs, accessible to a subset of the privilege levels. Any attempt to either access a CSR without the appropriate privilege level or to write into a read-only CSR raises an illegal instruction exception [12].

F. EXCEPTIONS AND INTERRUPTS

Both exceptions and interrupts on RISC-V processors are considered traps. As exceptions and interrupts happen during runtime, the processor provides mechanisms to make an unscheduled procedure call to an arbitrary address [11]. RISC-V classifies traps into two main categories: synchronous and asynchronous. Synchronous traps are exceptions resulting from an instruction execution, such as accessing an invalid memory address or executing an invalid instruction. Asynchronous traps are interrupts and are external events that occur asynchronously to the instruction stream. RISC-V sets the most significant bit of the `mcause` control status registers to identify whether the trap is synchronous or asynchronous, and the least significant bits for identifying which interrupt or exception occurred [2].

RISC-V has three sources of interrupt: software, timer, and external interrupts. Software interrupts enable programmers to interrupt a given Central Process Unit (CPU), allowing efficient Inter-Process Communication, while timer interrupt is triggered when a hart (i.e., hardware thread) time comparator exceeds or equals the global `timebase` register. External interrupts, in turn, are asserted by a platform-level interrupt controller [12]. The mechanisms for raising and clearing interrupts can vary according to the hardware platform, given that they can use different memory maps and demand divergent features from their interrupt controller [2]. However, all RISC-V systems handle exceptions and mask interrupts in the same way.

G. ADOPTION AND MOTIVATION

RISC-V International is a nonprofit organization and counts with more than 40 sponsoring companies [3]. For the past decade, NVIDIA has been shipping their Graphics Processing Units (GPUs) with proprietary microcontrollers called Falcon, but since 2016, NVIDIA is evaluating RISC-V for their GPUs [14]. Western Digital open-sourced the SweRV Core, an industry-qualified processor featuring a 32-bit in-order, 2-way superscalar design with a 9-stage pipeline core [15]. In 2019, Alibaba revealed its first embedded high-performance 64-bit RV64GCV RISC-V-based processor, which includes a set of custom extensions [16], [17].

Due to the recent US restrictions on its ARM design, Huawei HiSilicon released their first RISC-V-based board together with the Harmony OS in May 2021 [18]. With LEON SPARCv8 being the dominant architecture in European avionics and facing difficulties in leveraging software from the commercial domain, the Dependable Real-time Infrastructure for Safety-critical Computer (De-RISC) project introduces the RISC-V architecture for aviation and space environments by using fault-tolerant techniques and supporting compute-intensive applications [19]. Further, apart from Apple's recent switch to ARM-based System-On-Chip (SoC), it has recently demonstrated an interest in exploring the RISC-V architecture [20].

Apart from industrial motivation and adoption, academic development is also in progress. For example, the PULPino processor from the Swiss Federal Institute of Technology (ETH Zurich) is ready for industrial standards. PULPino is optimized for low-power consumption, concentrating on providing IoT solutions [21], [22]. Given this background, the institution of technology Semico Research Corp. estimates that, in 2025, the market will have around 62.4 billion RISC-V cores worldwide [13], [23], represented in Figure 1.

III. MATERIALS AND METHODS

In order to carry out this survey and identify the current state of the RISC-V software support, we conducted a bibliographic survey using the following materials and methods. First, we performed a search on the IEEE, ACM, Springer, and Usenix digital libraries using the query: `("RISC-V" AND ("Software" OR "Operating System" OR "OS"))`. We limited the search to retrieve works published in the last five years. Furthermore, we applied the impact of the publication channel and the normalized number of citations received by each work to select the ones to be analyzed.

After the search and selection phase, we categorized the RISC-V software ecosystem as shown in Figure 2. From top to bottom, we first explore the variety of environments RISC-V is applied and then continue to explore available implementations that have been proposed. We then further explore the different software architectures that have been ported or implemented on RISC-V and the deployment characteristics, such as security, reliability, and power features.

This work will further explore these categories in the following sections: application fields (Section IV); RISC-V implementations (Section V); and software support, including software architecture, OS support, file systems, network stack, and uncategorized features (Section VI). In addition, we analyzed the additional features, including the support for security, reliability, and low-power operation (Section VII).

IV. APPLICATION FIELDS

The space industry has had difficulties leveraging software from the commercial domain and is now considering alternative architectures in a larger commercial market [19]. Furthermore, with the introduction of SoC and multi-core processors, this industry now seeks to migrate components to a higher level of integration, introducing a new set of issues that needs to be solved. A modular architecture, such as RISC-V, enables designers to extend the processor and implement functionalities tailored to their application requirements, making RISC-V highly adaptable to various environments.

The De-RISC project introduces a novel RISC-V hardware/software platform meeting the requirements and

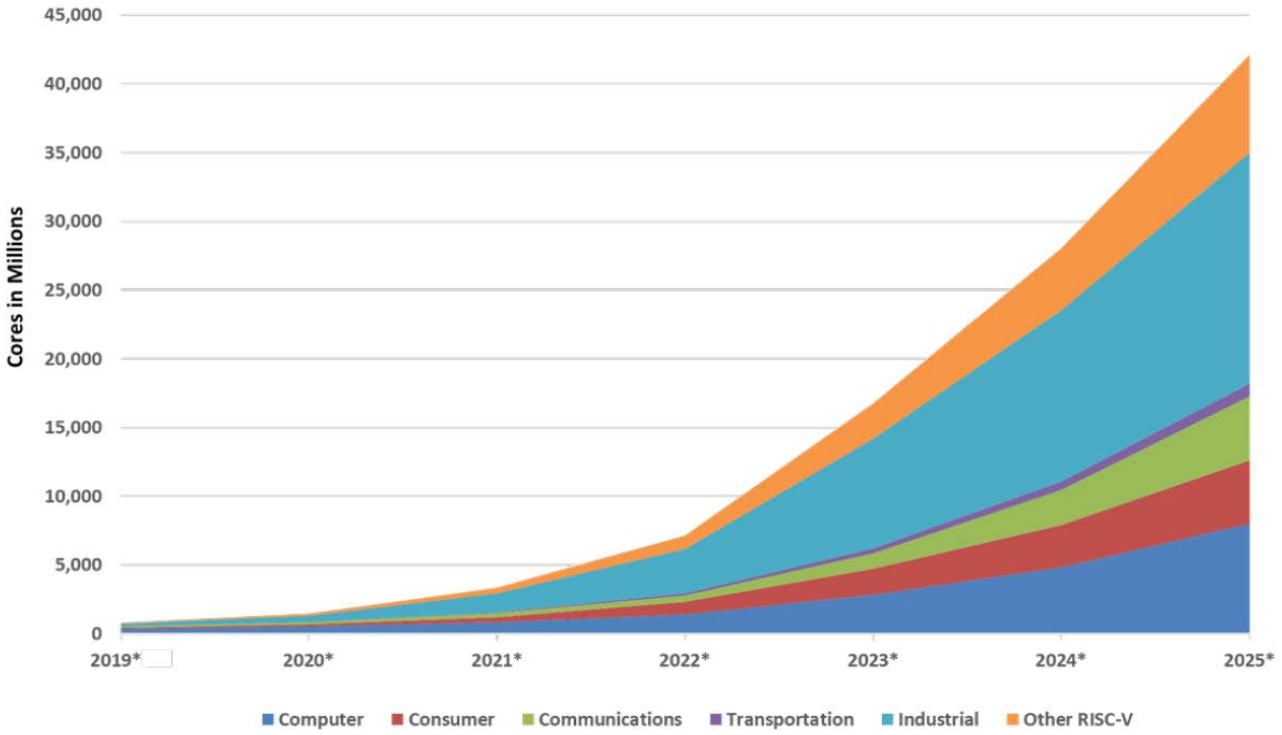

FIGURE 1. Market forecasting for RISC-V [13].

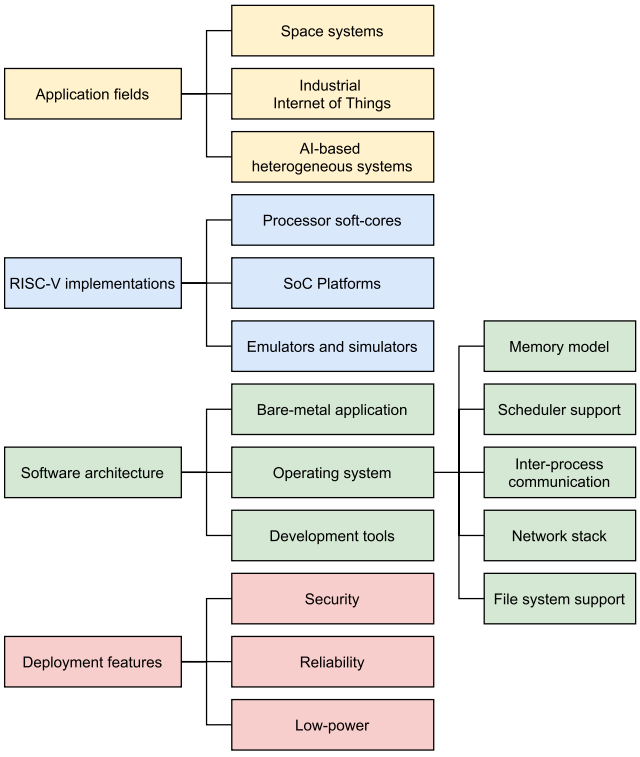

FIGURE 2. RISC-V ecosystem overview.

functionalities imposed by the space environment. De-RISC meets reliability by designing its Multiprocessor System-on-Chip (MPSoC) with fault-tolerance techniques to provide a correct operation in the presence of faults [24].

The SELENE project, depicted in Figure 3, proposes a reliable platform for safety-critical computing. It is built on open-source technology and enables designers to adapt the system to a specific requirement domain, integrating applications of different criticalities and demands, achieving a diverse set of redundancy and performance [25]. Furthermore, the openness of RISC-V improves component reuse and prevents the need for developing projects from scratch, increasing productivity and reducing cost [26].

With the increase of embedded device deployments and the expected growth of the Industrial Internet-of-Things (IIoT) market to reach 1.11 trillion US dollars by 2028 [27], [28], IoT devices have become a fundamental part of the lives of billions of people around the world. Securing these devices without loss of performance or increasing power usage has become a matter of discussion with the accelerating growth of connected devices. In this regard, the RISC-V community has investigated several security issues for constrained devices. Solutions addressing high-level goals identified by NIST IR 8228 [29], authenticated secure-boot images while verifying updates during runtime [30], hardware enforcement and memory isolation mechanisms [31], and post-quantum solutions [32] are all available for the RISC-V architecture.

Apart from security and reliable applications, RISC-V has a set of research fields concerning AI-based applications. In [33], the authors seek to provide a heterogeneous processor design for Convolutional Neural Network (CNN)-based applications by proposing a domain-specific architecture design and an accelerator for the inference of CNNs, adhering to low-power characteristics. The SiFive Intelligence

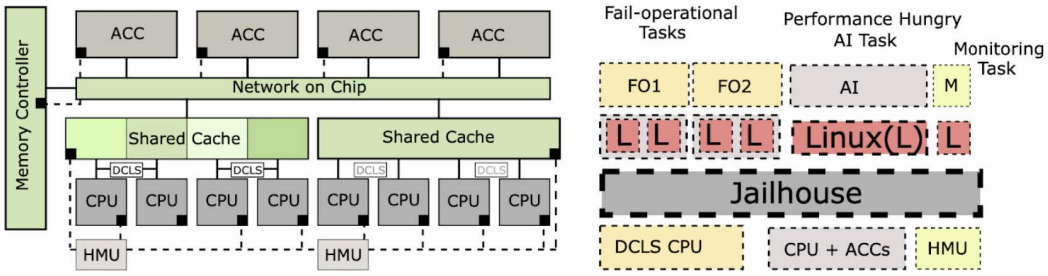

FIGURE 3. Architecture of the SELENE project [25].

X280 is a multi-core RISC-V compliant processor that optimizes Artificial Intelligence and Machine Learning (AI/ML) inferencing computing. X280 uses the vector extension and SiFive Intelligence extensions, making the core suitable for high-throughput single-threaded and power-constrained applications [34]. Finally, the work [35] uses a RISC-V-based architecture to provide a platform for Robot Operating System (ROS)-based applications.

The RISC-V architecture is also being applied in approximate computing solutions. The work [37] proposes a methodology to help designers to select the most optimized Floating-Point Unit (FPU) configuration that meets a given threshold for an specific application. Similarly, the work [38] presents an automated quality-driven methodology that combines different approximate computing techniques to explore and apply error-tolerant sections to applications.

RISC-V is highly adaptable for various contexts, given the open specification, modularity, and the community. It enables designers to deploy from small, low-cost embedded devices to a large-scale system with custom extension support meeting the requirements imposed by the system.

V. RISC-V IMPLEMENTATIONS

Despite the requirements of a general-purpose processor, RISC-V enables domain-specific implementations to exist, either by implementing the RISC-V base ISA with custom characteristics or implementing a new extension.

A. PROCESSOR CORES

The PULPino project provides three RISC-V core implementations: (i) a 32-bit 2-stage pipeline named Ibex (formerly named Zero-riscy), (ii) a 4-stage pipeline core named CV32E40P (formerly named RI5CY), and (iii) a 64-bit 6-stage core named CVA6 (formerly named Ariane). Ibex, maintained by lowRISC and illustrated in Figure 4, is well suited for embedded control applications and supports the Integer (I) or Embedded (E), Integer Multiplication and Division (M), Compressed (C), and Bit Manipulation (B) extensions, and is available as a SystemVerilog project [36], [40]. The CV32E40P core, depicted in Figure 5, implements the RV32IMFC ISA and the Xpulp extension for higher code density, performance, and energy efficiency characteristics [36], [41]. Finally, the CVA6 core, illustrated in Figure 6, implements the RV64IMAC ISA, focuses on reducing the critical path, and supports a configurable size and separate Translation Lookaside Buffer (TLB) [39].

SonicBOOM (Figure 7) is a Register-Transfer Level (RTL) implementation of an RV64GC superscalar out-of-order core written in Chisel3. SonicBOOM improves the architectural characteristics of BOOMv2 and seeks to provide a state-of-the-art platform for high-performance research. The SonicBOOM core optimizes the execution path and redesigns the instruction fetch unit with a hardware Tagged Geometric (TAGE) branch predictor algorithm. Further, SonicBOOM's load-store unit provides multiple loads per clock cycle and achieves 6.2 CoreMark/MHz [42], [43].

NOEL-V is a synthesizable VHSIC Hardware Description Language (VHDL) processor with support to the 32- and 64-bit RV{IM,IMAC,GCH} ISA, with single- or dual-issue features. The core is available as part of a subsystem that includes system peripherals and is configurable to use RISC-V extensions. The NOEL-V dual-issue processor allows two instructions per clock cycle and implements advanced branch prediction capabilities [19], [44].

In [33], the authors proposed a CPU based on the low-power 2-stage pipeline Hummingbird E200 RISC-V core, which implements the RV32IMAC ISA.

The authors of [46] presented the HARV processor, with support to the RV32I ISA and focus on reliability. This processor core employs fault tolerance techniques in its internal structures, i.e., error-correcting code in all registers and Triple Modular Redundancy (TMR) in the controller and the Arithmetic Logic Unit (ALU), to provide reliability.

In June 2021, SiFive announced the P270 and the P550 cores to their Performance family. P270 is an 8-stage, dual-issue, efficient in-order pipeline core compatible with RV64GCV ISA. P550 features 13-stage, triple-issue, out-of-order pipeline core implementing the RV64GC ISA and delivering a SPECInt 2006 score of 8.65/GHz, the highest performance RISC-V processor available to date [47]–[49].

B. SOC PLATFORMS

The PULPino project provides a set of complete system platforms: (i) PULPino (Figure 8) and PULPissimo (Figure 9),

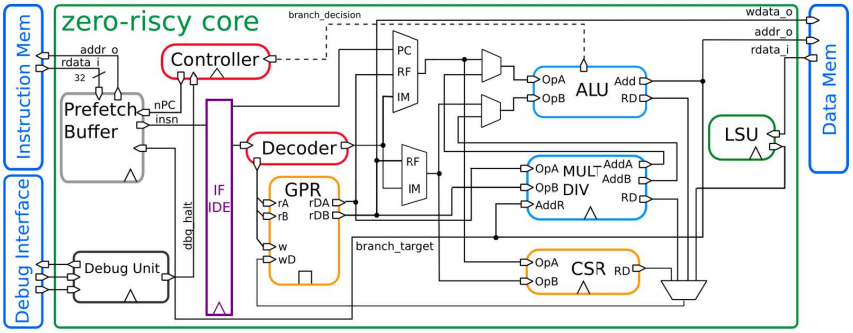

FIGURE 4. Block diagram of the Ibex core [36].

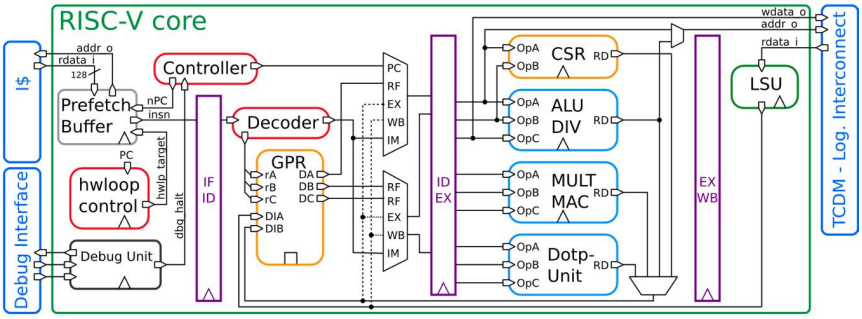

FIGURE 5. Block diagram of the CV32E40P core [36].

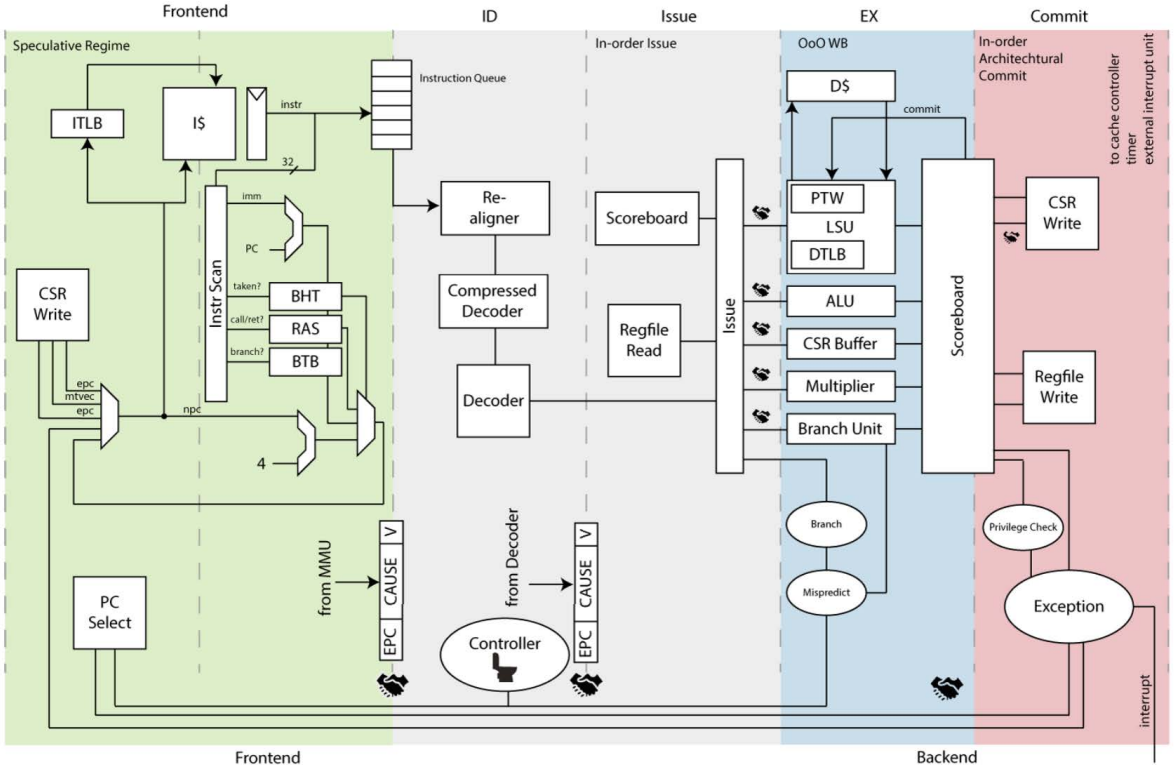

FIGURE 6. Block diagram of the CVA6 core [39].

two platforms based on a single-core microcontroller, (ii) OpenPULP (Figure 10), a multi-core IoT processor, and (iii) Hero (Figure 11), a multi-cluster heterogenous accelerator, which combines a parallel many-core accelerator

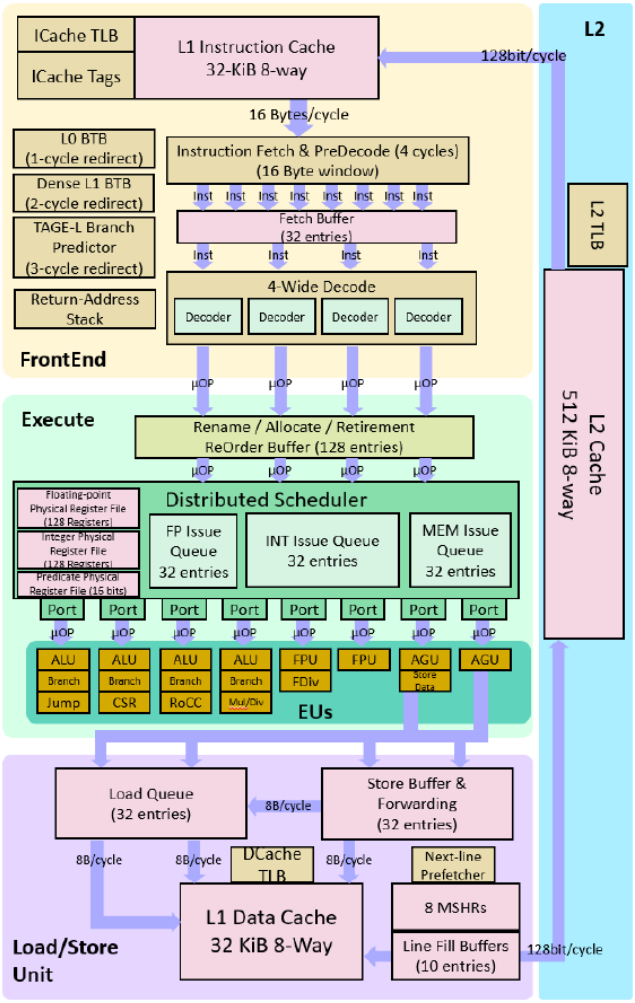

FIGURE 7. Block diagram of the SonicBOOM core [42].

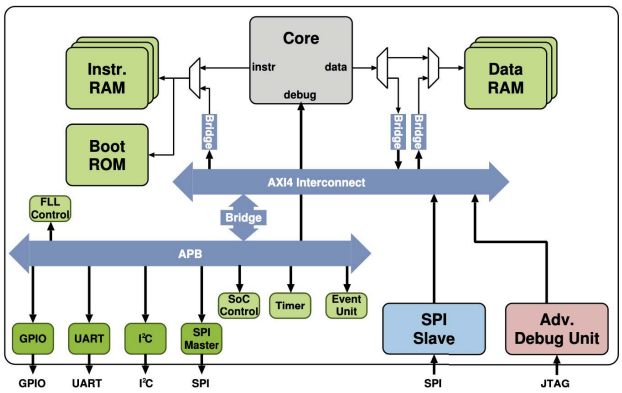

FIGURE 8. Block diagram of the PULPino SoC [36].

on an Field-Programmable Gate Array (FPGA) with a hard ARM Cortex-A multi-core host processor.

The FE310-G000 SoC by SiFive, illustrated in Figure 12, has an E31 core with a single-issue in-order pipeline and implements the RV32IMAC ISA. The E31 core peaks a

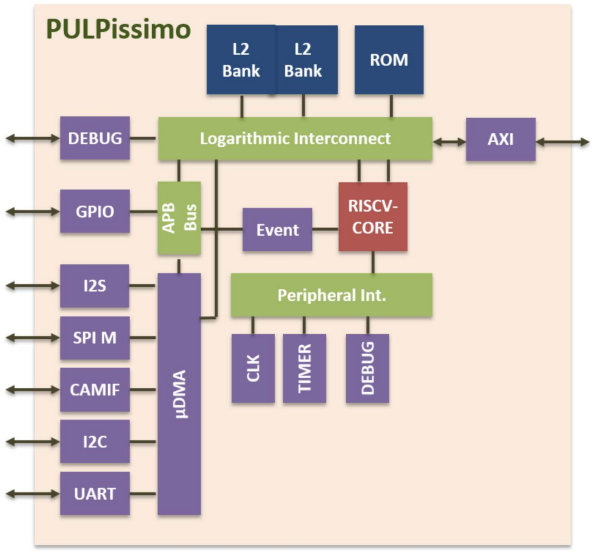

FIGURE 9. Block diagram of the PULPissimo SoC [45].

sustainable execution rate of one instruction per clock cycle. The FE-3100-G000 features two Universal Asynchronous Receiver-Transmitter (UART) devices for serial communication, three Quad Serial Peripheral Interfaces (QSPIs), three Pulse-Width Modulation (PWM) ports, and support to low-power operations [50]. The HiFive1 is one of the first commercial available RISC-V boards, which features the FE310-G000 SoC. The work [51] proposes a set of novel approaches that enhance Virtual Prototype (VP) design flow using the HiFive1. Similarly, the work [38] proposes an automated quality-driven methodology which supports the HiFive1.

The PolarFire SoC FPGA by Microchip is a low-power, thermal efficient, and defense-grade security device for intelligent, connected systems. The SoC has a 5-stage single-issue in-order pipeline RISC-V and does not suffer from Meltdown and Spectre exploits of common architectures. PolarFire cores are deterministic and coherent with the memory subsystem, enabling Linux and real-time capable applications to execute. The SoC implements the RV64GC and RV64IMAC ISAs [54].

The work [30] uses a Universal Sensor Platform (USeP) SoC to meet IoT requirements, which features a RISC-V processor and integrates peripherals with a scalable subsystem as a Three Dimensional System-in-Package (3D-SiP).

The Sipeed MAix-GO development board, employed by [55], features a Kendryte K210 SoC, with a dual-core RV64GC processor, including support to a single and double-precision FPU.

C. EMULATORS AND SIMULATORS

The open-source QEMU project is a generic machine emulator and virtualizer, providing a virtual model of an entire machine to run a guest OS. QEMU supports 32- and 64-bit RISC-V implementation and several different

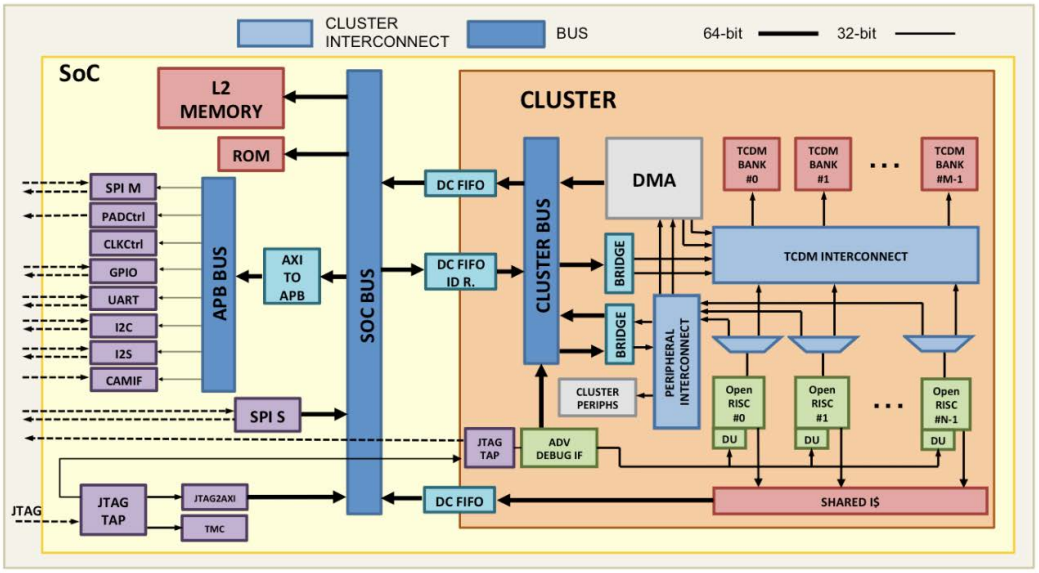

FIGURE 10. Block diagram of the OpenPULP SoC [52].

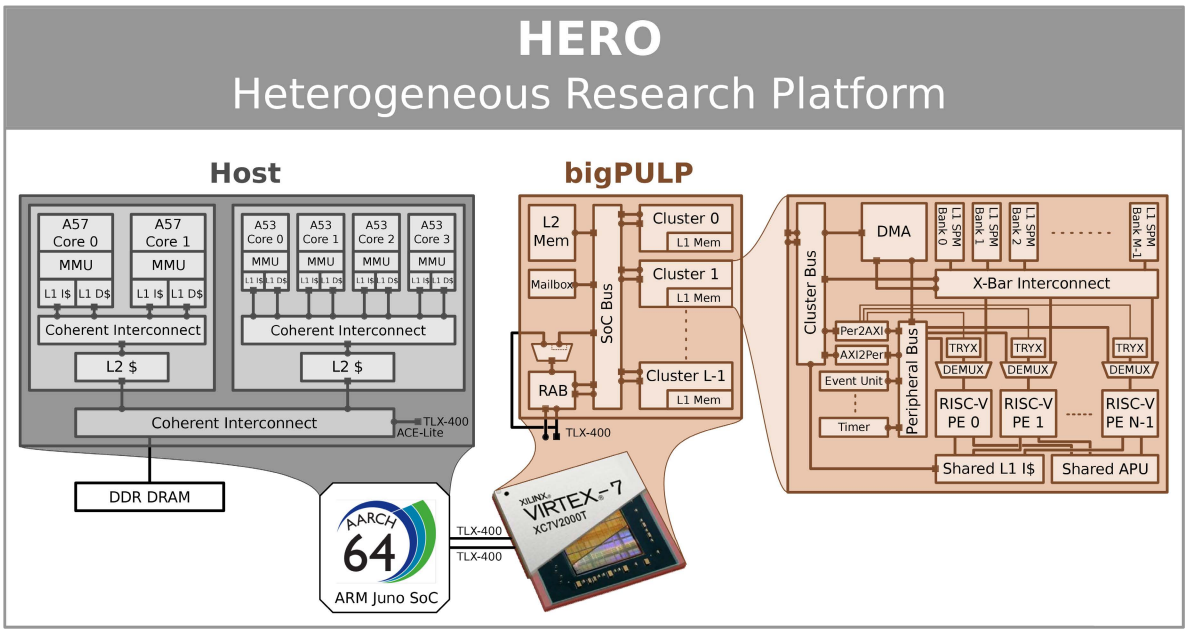

FIGURE 11. Block diagram of the Hero SoC [53].

machines, including Microchip's PolarFire SoC, SiFive's HiFive Unleashed, and Shakti C platform [56].

Gem5 is an open-source modular simulator for computer architecture research, including system-level architecture and processor microarchitecture. Gem5 support to RISC-V privileged ISA specification is still in development [57].

Spike is a RISC-V ISA simulator with a functional model of one or more RISC-V hardware threads. It supports a large set of RISC-V extensions and eases simulating new instructions by letting the user describe the functional behavior and add the opcode and opcode masks [58].

RISC-V Assembler and Runtime Simulator (RARS) is built on top of MIPS Assembler and Runtime Simulator (MARS) and extends the software to enable features such as instructions hot-load of RISC-V extensions [59]. Similarly, WebRISC-V [60] provides a web-based server-side simulation of a 5-stage pipeline RISC-V implementation. The application enables writing simple assembly programs and visualizing data in registers, memory, and the internal state of the pipeline. Ripes is an open-source visual computer architecture simulator built around the RISC-V ISA [61], providing multiple microarchitectural models to explore a typical

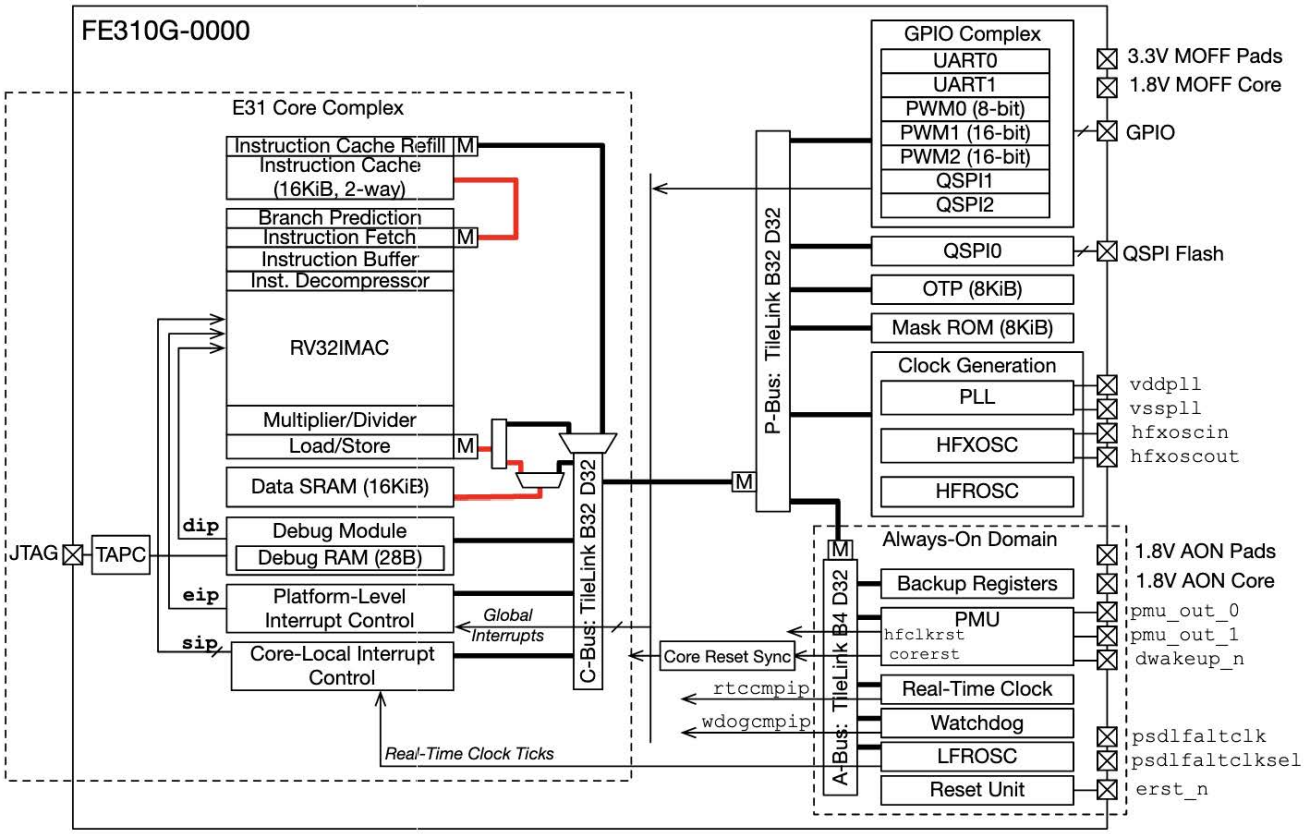

FIGURE 12. Block diagram of the FE3100-G00 SoC [50].

pipeline processor. Similar simulators include Jupiter [62] with support to RV32IMF ISA, Vulcan [63] with support to RV32IMAF ISA, and the emulsiV simulator [64], intended for teaching basics of computer architecture.

D. DISCUSSION

This section reviewed different RISC-V implementations to comprehend the RISC-V ecosystem and the available research towards the open ISA. Given RISC-V's open specification, multiple implementations tackle specific problems for a given application field, making RISC-V highly adaptable. Besides the presented RISC-V cores and SoCs, there are others not mentioned in this work that has shown to be production-ready, supporting low to high-performance computing requirements.

VI. SOFTWARE SUPPORT

The ISA interface encompasses the machine language instructions that a computer can run, acting as a boundary between the hardware and software layer [65], represented in Figure 13. In RISC-V's privilege layer, both the user application and the OS can access the ISA directly, as RISC-V provides a subset of instruction repertoire per layer.

The ABI defines a standard for binary portability across programs by defining the system call interfaces to the OS, and hardware resources available in the system through the ISA [65]. This allows binaries compiled to a specific ABI to run without modifications in a different system with the same ISA and OS.

The following subsections seek to evaluate and review works that address different types of software architecture to understand the scope of end-user application and OS support.

A. SOFTWARE ARCHITECTURE

The software architecture relates to its structure, how these components are separated, and their interrelationships. Software engineers seek to structure software to meet current and future demands, making the system reliable, manageable, adaptable, cost-effective, and scalable [67].

1) BARE-METAL APPLICATIONS

Applications running on bare-metal systems have direct access to the processor and peripherals. These applications are not managed by an OS layer, enabling applications that require runtime guarantees along with constraint hardware to function as expected.

The work [66] extends their previous work on COAST [68], which explores Commercial Off-the-Shelf (COTS) systems to provide an automated compiler modification, bringing support to several new processing platforms, such as

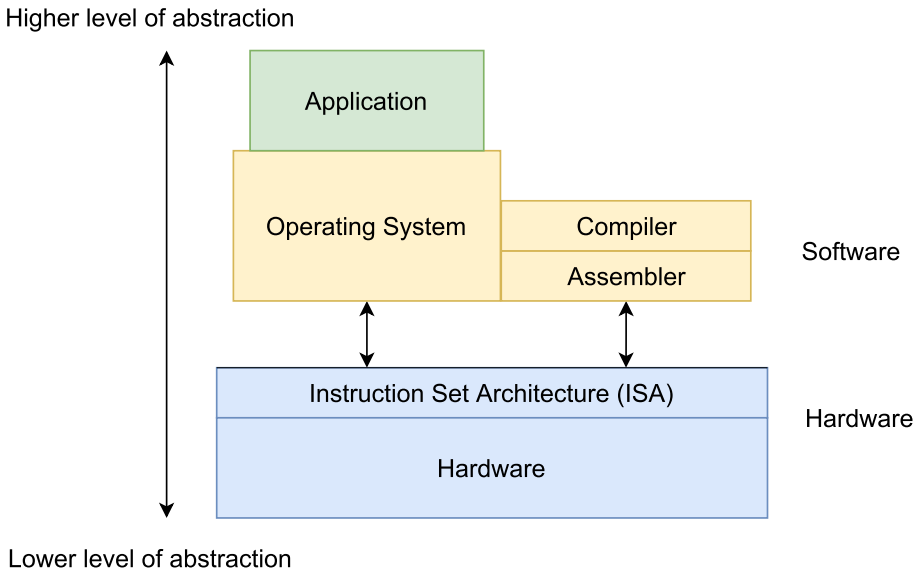

FIGURE 13. Computer system abstraction.

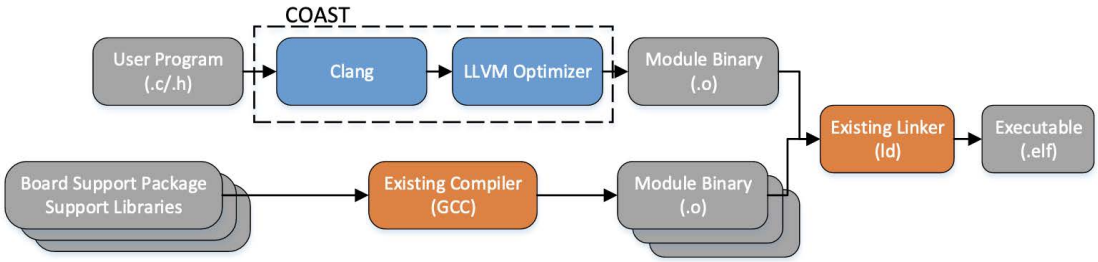

FIGURE 14. COAST architecture [66].

RISC-V and Xilinx SoC-based products. The compiler inserts dual- or triple-modular redundancy during the compilation phase, enabling the software to correct errors during runtime. The proposal is attractive for applications requiring tolerance against Single Event Effect (SEE) and is well-suited for processing in a high radiation environment. Although the tool provides both Duplicate With Compare (DWC) and TMR mechanisms, the default configuration uses Variables 3 (VAR3) protection mechanism. Figure 14 illustrates the COAST architecture.

Stahl *et al.* [69] propose a driver development flow to ease the definition of hardware/software interfaces without a fixed register layout and reduce the development effort and memory usage of an Microcontroller Unit (MCU). The proposal uses a Domain-Specific Language (DSL) to describe the behavior using features such as bit field arrays and hierarchy. In addition, the work proposes a heuristic, code analysis, and generation technique to find an optimized register layout that exploits its performance and memory footprint. The proposal uses PULPino General-Purpose Input/Output (GPIO) and Serial Peripheral Interface (SPI) drivers and has reduced the estimated run time by 52%, 32% reduction of memory accesses, while the driver code size is reduced by 22%.

2) OPERATING SYSTEMS

Zhang *et al.* [70] propose an automatic kernel code synthesis and verification technique framework, which seeks to build a verifiable OS kernel with a high degree of proof automation. The proposed framework enables a software developer to write the required specification of the kernel and translates the corresponding specification into C code. The authors provided a kernel named iv6, which combines exokernel architectural aspects with characteristics of a mathematically proven and trustworthy microkernel. The kernel initializes in RISC-V's M-Mode, runs in a separate address space from userspace, and uses identity mapping techniques for the kernel along with additional characteristics.

In addition to the OS architecture, the work [71] explores the building blocks of the development cycle of an open-source OS for the 64-bit little-endian RISC-V architecture, proposing a custom Linux distribution from Linux From Scratch (LFS) with independent userspace and a package

manager written in Lua programming language. The proposal seeks to provide a Linux-based distribution for open-source hardware and is the first OS targetting RISC-V.

Implementations of a RISC-V architecture with hardware/software co-design are also available, such as the work proposed by [72], which captures possible application-kernel interaction as a Finite-State Machine (FSM) and integrates the Real-Time Operating System (RTOS) semantics directly into the processor pipeline. The proposal significantly improves event latencies, interrupt lock times, and memory footprint at a moderate cost of FPGA resources.

Malenko and Baunach [73] employ a microkernel-based operating system named SmartOS, which seeks to provide basic functionality in privilege-mode while other functionalities run in user-mode. Further, to evaluate the reliability of a RISC-V-based SoC, the work [74] compares five algorithms against a Linux-based OS and a bare-metal implementation by using a lowRISC implementation on a Xilinx FPGA.

The work [75] aims at providing a basic microkernel for the RV32I ISA. The microkernel has support to trap handling and multiple RISC-V privilege modes. The work is focused on enabling students and lecturers to enrich further their comprehension of computer architecture and OS development.

3) DEVELOPMENT TOOLS

The work [76], illustrated in Figure 15, proposes a design method to generate domain-specific many-core architectures with provided frameworks and automated steps using software tools. The solution facilitates engineering and creates many-core architectures with different configurations, including core augmentation through instruction extensions and custom accelerators.

Torres-Sanchez *et al.* [55] evaluate the development environment toolchains and debugging process concerning the Sipeed MAix-GO development board and Tiny YOLO v2 support by deploying a low-power IoT edge application, achieving reasonable cost and performance characteristics.

Herdt and Drechsler's proposal [51] provides an automated formal verification tailored for SystemC-based virtual prototype on top of a RISC-V ISA and significantly improves verification quality, reducing overall verification effort.

Furthermore, the work [77] provides a library to run lightweight and energy-efficient neural networks. The framework automates deployments on MCUs with and without an FPU. Finally, the work [78] proposal enables ISA designers to iteratively refine and evaluate ISA specifications, allowing the improvement upon each result.

B. OPERATING SYSTEM ARCHITECTURE

With the growing complexity of computer hardware, the operating system provides an abstraction layer to the user, acting as an interface between applications and computer hardware, enabling programmers to develop software without knowing much of the underlying details and executing software efficiently. The ISA defines the OS capabilities, as it defines the available instructions to the application. RISC-V has a defined subset of OS support as well as user and machine instructions [65], [79].

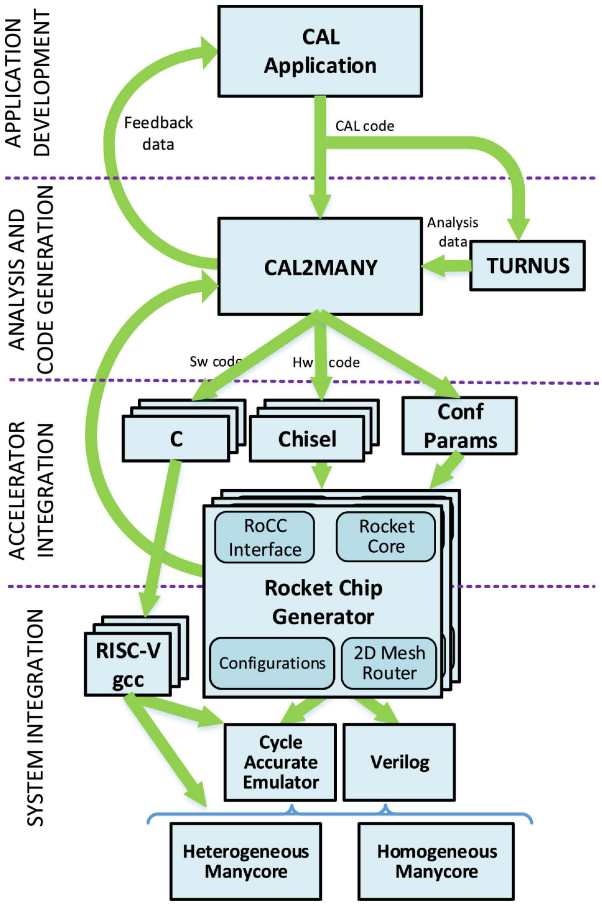

FIGURE 15. System generator framework overview of [76].

1) MEMORY MODEL

The work [76] uses scratchpad memory to avoid having to deal with cache coherence issues. Further, all cores and components in the architecture share the same address space. The memory module routes memory accesses to the data cache or another component through a crossbar network.

The SmartOS employed by [73] organizes the memory with the linker script to structure the Task Control Blocks (TCB), Resource Control Blocks (RCB), and Event Control Blocks (ECB) arrangements and the regions for task stack, data, and entry function.

The cache controller of NOEL-V supports a store buffer First In, First Out (FIFO) with one cycle per store and a wide Advanced High-performance Bus (AHB) data width support to enable start stores and fast cache refill [19].

Trippel *et al.* [78] propose a memory verification model to check for bugs on hardware and software memory models by providing a tool capable of verifying High-Level Language (HLL), compilers, and ISAs. Further, the authors present potential inefficiencies of the RISC-V ISA specification and identify possible solutions to mitigate these inefficiencies.

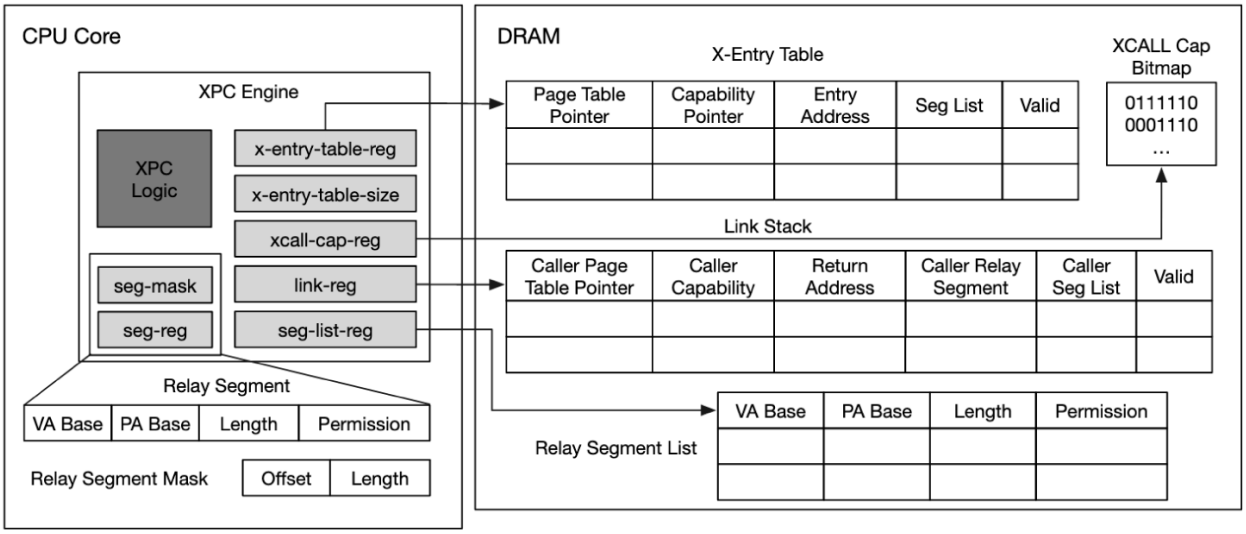

FIGURE 16. XPC architecture [80].

2) SCHEDULER SUPPORT

A processor scheduler assigns processes to be executed by the processor at a given time, meeting system response time, throughput, and processor efficiency objectives and allowing multiple processes to exist concurrently in a multiprogramming system. Furthermore, schedulers can also have real-time features, which guarantee that high-priority tasks execute within a specific time constraint.

The work [81] implements an improved version of a deterministic coprocessor task scheduler based on the Earliest-Deadline First (EDF) algorithm. In addition, the work adds support for CPUs to run two or four real-time tasks in parallel, improving real-time system performance and reducing resource costs using the Heap Queue sorting architecture for the Ready Queue implementation. The improved coprocessor and the task scheduler were described in SystemVerilog and verified by simulations and a simpler version of the Universal Verification Methodology (UVM).

Naylor *et al.* [82] explore the potential of a distributed soft-processor overlay to deliver appropriate performance for FPGA clusters. In addition, the work uses an FPGA-optimized hyperthreaded RISC-V soft-core, named Tinsel, and implements a barrel-scheduled multithreaded core that uses a large subset of the RV32IMF ISA. Tinsel tolerates inherent latencies of a floating-point operation and off-chip memory access. Further, it provides a programming environment through an interface on top of the Tinsel Application Programming Interface (API), abstracting architectural details and handling graph mapping onto the overlay.

Tasks in SmartOS, employed by [73], are preemptive with unique static priorities defined at compile-time, and active priorities dynamically modified by the resource manager and used by the scheduler. Further, SmartOS's kernel uses a TCB structure for managing tasks.

3) INTER-PROCESS COMMUNICATION

Processes may need to communicate with one another, requiring the OS to provide a communication mechanism, preventing race conditions and ensuring the proper sequence of communications. The communication should be well-structured and preferably without using interrupts. The communication the OS provides is known as inter-process communication [65], [79].

Du *et al.* [80] use the SiFive U500 SoC with the RV64IMAFD ISA to add support for a secure and efficient cross-process call architecture. The proposed architecture is illustrated in Figure 16. The work proposes a hardware-assisted OS primitive named XPC, which uses an asynchronous Inter-Process Communication (IPC) across different address spaces and enables a direct switch between IPC caller and callee without trapping into the kernel. XPC improves throughput using a new address-space mapping mechanism and provides a multithreading API with the migration thread model. The work supports split thread state, per-invocation C-Stack, and Android's Ashmem subsystem. The authors assessed performance, achieving a 0.3ms latency for 4KB data, with a 1.6x improvement, mainly from the secure zero-copying message transfer. XPC is compatible with traditional address-space isolation and can easily integrate with existing OS kernels.

The work [83] proposes in-process isolation based on dynamic memory protection domains. Software components can rely on enforced security by using a set of provided protection keys and associate memory, supporting usage rights and entry points.

4) ADDITIONAL FEATURES

The Sipeed MAix-I development board employed by [55] features a highly integrated Espressif ESP8285 SoC with complete self-contained Wi-Fi networking capabilities.

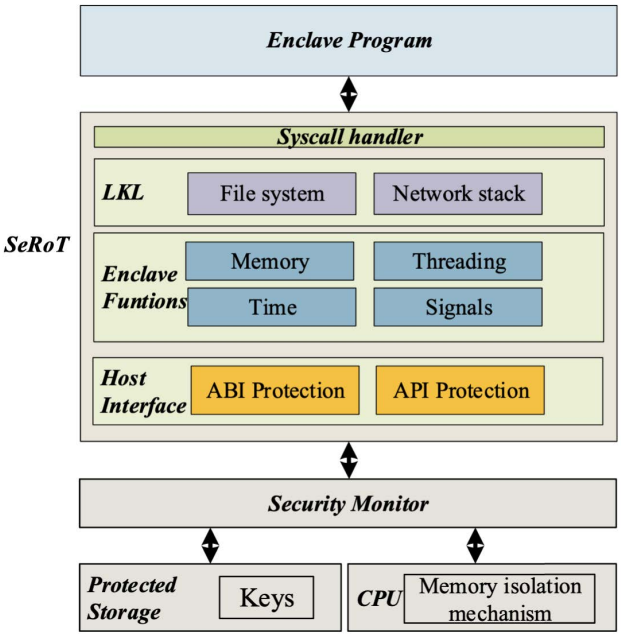

FIGURE 17. SeRot architecture [84].

Du *et al.* [80] use lwIP, a small independent implementation of the TCP/IP protocol suite, as a network stack server for the microkernel and a loopback device driver to pack and send to the network device server.

Liu *et al.* [84] propose a secure runtime system named SeRot, illustrated in Figure 17, which addresses primitives to support unmodified applications. SeRot uses the Linux Kernel Library (LKL) network stack to provide network functionalities to the application. It also enables applications to access a file system by using LKL for accessing and managing file system calls.

C. DISCUSSION

Even with little time in the market, RISC-V's software ecosystem includes compilers, state-of-the-art OSs, and system generation tools to ease hardware/software interfacing. In addition, the open specification of RISC-V has made it possible for one to support and implement robust software architectures to meet a range of requirements.

With the increasing demand for computing power, the simplicity of the RISC-V ISA enables OS engineers to support the architecture and contribute to its specification. On the other hand, the growing complexity of architectures such as x86 makes it difficult for developers to keep up and contribute to the development of the ISA.

RISC-V's openness has led to a wide range of RISC-V applications, including academic and industrial usage, helping researchers to explore the benefits of an open ISA for software development. Overall, the works presented have shown how RISC-V can support software and operating systems with sophisticated memory models, inter-process communication, compiler techniques, and toolchains.

VII. DEPLOYMENT FEATURES

To further investigate the RISC-V software ecosystem, this section reviews the works that tackle security, reliability, and power management characteristics of RISC-V in order to improve the executing software.

A. SECURITY

Hunt *et al.* [86] argue that hardware enforcement isolation mechanisms have historically been the basis for a secure system. However, providing redundant protection for isolation by combining software and hardware techniques can improve system security. Zhang *et al.* [70] implement a Kernel Page-Table Isolation (KPTI) with a channel for accessing userspace from the kernel space by a block of a shared memory region. Further, the framework reduces the impact of human error during the development phase. Similarly, the work [73] explore memory isolation techniques and implement a hardware/software co-design approach for memory isolation by providing two hardware components: Device Driver Isolation Module (DDIM) and System Call Tracing Module (SCTM).

Hwang *et al.* [87] provide a kernel hardware-based monitoring platform and a hardware interface architecture to overcome the semantic gap problem in a RISC-V core. The work introduces a platform called RiskiM to ensure kernel integrity, and an interface architecture named Program Execution Monitor Interface (PEMI). In order to overcome the semantic gap issue, PEMI provides all internal states of the host system to RiskiM to fulfill its monitoring task and protect the kernel in the presence of attacks. Furthermore, Sisejkovic *et al.* [88] implement a hardware/software solution using the 64-bit 6-stage in-order, single-issue Ariane core. The work proposes a hardware design protection against hardware Trojans inserted during the production phase through netlist obfuscation provided by logic locking.

In [89], Fischer *et al.* exploit hardware misconfigured access control of the Read-Only Memory (ROM) as well as incorrect implementations of SoC firmware. The work explores memory overwrites of peripheral regions and has found no code that implements the Counter (CTR) block cipher mode.

Auer *et al.* [30] address three high-level goals, identified by the NISTIR 8228 report, to provide a secure architecture on IoT devices. The work uses a secure-boot mechanism with a certificate chain to authenticate boot images, and the update verification happens during runtime. Furthermore, in order to explore and present a compatible RISC-V with Trusted Execution Environment (TEE) featuring security algorithm accelerators, the work [90] uses a 64-bit RISC-V with the IMAFDC ISA implementation, along with a Rocket chip generator as a hardware platform. The work features SHA-3 and Ed25519 accelerators and provides a procedure for the software to authenticate a Linux bootloader.

Similarly, Siddiqi *et al.* [91] use a lowRISC FPGA implementation to propose a hardware/software-based solution

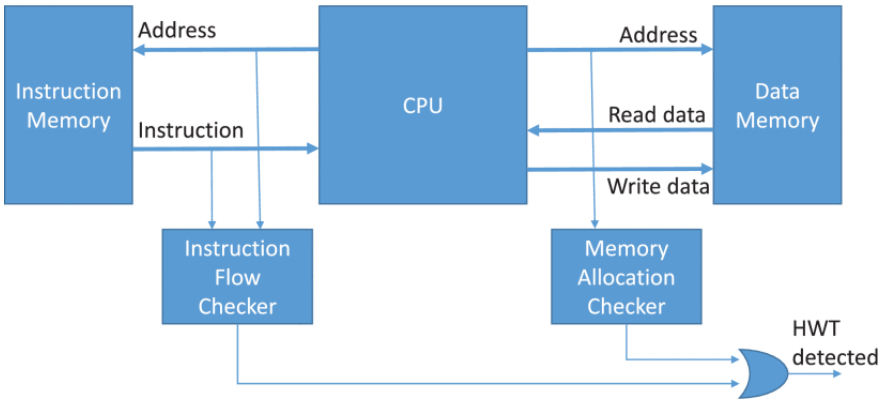

FIGURE 18. Architectural protection mechanism of [85].

to secure system integrity. In addition, the work provides a solution for securing the processor by implementing a self-authenticated secure boot and provide Information Flow Tracking (IFT) to detect and stop memory corruption attacks.

Dessouky *et al.* [92] provide a testbed of real-world software-exploitable RTL bugs based on RISC-V SoCs, exploring TLB, cache, and memory attacks to identify specific vulnerabilities classes. To further explore microarchitectural security, the authors conducts two case studies on the Ariane and PULPissimo SoC. Furthermore, to encourage SoC security research, Fischer *et al.* [89] employs the Ariane SoC RTL to investigate hardware vulnerabilities in SoC designs.

With difficulty in ensuring the trustworthiness of the fabrication process of silicon devices, Bolat *et al.* [85] propose an architectural protection mechanism, shown in Figure 18. The mechanism detects hardware Trojans infesting the instruction and data memories of the system by shielding the communication of the processor and memory.

The work [93] investigates undocumented instructions on RISC-V and ARM ISAs and propose two methods to look for undocumented instructions. Both methods execute a single instruction in a controlled manner, allowing the processor to determine if the instruction is valid by comparing the results to the ISA specification.

Fell *et al.* [94] explore side-channel attacks and investigate the impact of source-code obfuscation techniques to verify what information leakage is exploitable. The work proposes a compiler-based technique to mitigate leakage by employing an extension to the LLVM compiler.

Nyman *et al.* [95] extend the RISC-V ISA with seven new Storage Region Stack (SRS) management instructions to provide context-specific enforcement. The authors modify the instruction decode stage of the processor pipeline to interpret the new instructions. Similarly, the work [91] modifies the execute stage of the processor pipeline to add support to tag propagation and tag check features.

Finally, the work [95] proposes a Runtime Scope Enforcement (RSE) approach to mitigate all known Data-Oriented Programming (DOP) attacks efficiently on memory-unsafe programming languages. The proposed technique enforces memory safety constraints during compile-time, resulting in a low-performance overhead.

B. RELIABILITY

Fault injections can be applied to measure coverage and latency parameters, explore error propagation, and analyze the system workload and fault handling capabilities. In addition, different fault injection models can be utilized to exploit reliability characteristics of given hardware [96], requiring countermeasures to avoid and protect such exploits.

Laurent *et al.* [97] examine fault injection at the microarchitectural layer and propose a cross-layer approach. The authors inject single-bit faults in the experiments to simulate faulty behaviors in a lowRISC processor. The simulation results show behaviors of forwarding capabilities, speculative execution, and writing in a General-Purpose Register (GPR) during branch instruction. The authors also review software fault models and provide alternatives to bypass the countermeasures set by data- and control-flow integrity.

In [98], the authors inject faults on a 64-bit, 5-stage pipeline, lowRISC processor to highlight the importance of hidden registers in the processor pipeline. They provide countermeasures against hidden registers attacks by exploring the multiplier unit and forwarding characteristics, which are invisible from the software point of view and entirely dependent on the processor implementation. In the context of coverage analysis and fault simulation, the work [99] introduces a metric for RISC-V instructions and registers coverage. The work analyzes and compares three available RISC-V test suites and combines them into a unified tool.

The work [100] seeks to provide an efficient solution for fault effect analysis and simulation by introducing a virtual prototype-based approach. Bit flips are performed into the fetched/executed instructions, the GPRs and CSRs of a RISC-V processor and simulated to analyze the impact during software execution. In contrast, the work [74] found

that Single Event Upsets (SEUs) barely affect the number of silent data corruption in the presence of an OS. Similarly, the work [101] evaluates the reliability of running software applications against SEUs that affect the configuration and processor memories. The work compares the utilization of hardened-by-replication software with the baseline version in a set of benchmarks.

C. LOW-POWER

The IoT is one of the fastest-growing approaches among embedded systems. Devices used in IoT need to meet communication and energy efficiency requirements. In this context, Liu *et al.* [33] propose a domain-specific architecture design for Convolutional Neural Network-based IoT applications, with a heterogeneous processor design and accelerator for the inference of CNNs.

The methodology proposed in [38] seeks to minimize energy consumption and meets the defined threshold for a given application-level metric. Furthermore, the toolkit proposed in [77] takes multi-layer perceptrons trained with Fast Artificial Neural Network (FANN) to generate code for energy-efficient neural networks on microcontrollers.

Imianosky *et al.* [102] evaluate CCSDS 123 Compressor performance and power consumption on RISC-V and ARM architectures by executing the algorithms in FreeRTOS and Zephyr OSs. The authors concluded that RISC-V uses less power when compared to the ARM processor, while ARM offers higher performance. The work also showed that FreeRTOS has a lower overhead to the algorithm execution than Zephyr when executed on a RISC-V processor.

The SoC adopted by [30] has low-power characteristics, enabling designers to perform power-tradeoffs at runtime by adjusting the back-gate bias voltage. In contrast, the work [51] proposes methods to aid decisions that significantly impact the power consumption strategies by validating firmware-based power management implementations.

The proposal presented in [103] is well-suited for low-intensity tasks at low data rates and uses a microcontroller running at 25MHz. Further, the work presents ultra-low-power comparator circuits, which run on low frequencies to improve the power overshoot identification. Similarly, the work [55] keeps the CPU below 350mW when running face-detection routines and 35mW when the CPU is waiting for interrupts.

D. DISCUSSION

This section sought to review works that tackled or have similarities with security, reliability, and low-power characteristics. RISC-V enables designers to implement novel resources to improve the processor architecture features.

Due to the RISC-V nature of supporting a variety of environments, several works took advantage of the open ISA to provide security, reliability, and power efficiency solutions to secure processes as well as to run applications in an environment with low-power requirements.

VIII. CONCLUSION

Despite the novelty of RISC-V, a large variety of works aims at adapting and using RISC-V architectures to explore the capabilities of an open ISA. The different implementations of RISC-V aspire to improve or extend the fundamental specifications to satisfy peculiar computing needs. Low-level programming environments strive for a stable specification to comply with high-level programming environments. RISC-V's specification has primarily fulfilled these environments by providing engineers and researchers with a reliable ISA base. In addition, RISC-V's goal of supporting a broad set of computing environments has enabled enterprises and academia to tackle specific software requirements with low cost and flexibility.

This survey demonstrated how RISC-V is prevalent in works that seek to tackle specific requirements and how the community can benefit from an open architecture specification. Furthermore, given RISC-V's novelty and the rich software ecosystem, the community has well received and contributed to adopting RISC-V architectures into a wide range of systems, from small and constrained devices to large-scale computers.

REFERENCES

[1] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, "The RISC-V instruction set manual, volume I: User-level ISA," Dept. EECS, Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. 22, May 2017. [Online]. Available: https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf

[2] D. Patterson and A. Waterman, *The RISC-V Reader: An Open Architecture Atlas*, 1st ed. Berkeley, CA, USA: Strawberry Canyon, 2017. [Online]. Available: http://riscvbook.com/

[3] *RISC-V Int*. Accessed: Apr. 10, 2022. [Online]. Available: https://riscv.org/

[4] K. Asanovic and D. A. Patterson, "Instruction sets should be free: The case for RISC-V," Dept. EECS, Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2014-146, Aug. 2014. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.html

[5] G. S. Nicholas, Y. Gui, and F. Saqib, "A survey and analysis on SoC platform security in ARM, Intel and RISC-V architecture," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Springfield, MA, USA, Aug. 2020, pp. 718–721, doi: 10.1109/MWSCAS48704.2020.9184573.

[6] T. Lu, "A survey on RISC-V security: Hardware and architecture," 2021, *arXiv:2107.04175*.

[7] R. Newell, J. Xie, and H. Handschuh, "Survey of notable security-enhancing activities in the RISC-V universe," in *Proc. 17th Int. Workshop Cryptograph. Architectures Embedded Log. Devices (CryptArchi)*. Prague, CzechRepublic: CryptArchi, 2019, pp. 1–10. [Online]. Available: https://labh-curien.univ-st-etienne.fr/cryptarchi/workshop19/abstracts/newell.pdf

[8] B. Marshall, G. R. Newell, D. Page, M.-J.-O. Saarinen, and C. Wolf, "The design of scalar AES instruction set extensions for RISC-V," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, no. 1, pp. 109–136, Dec. 2020, doi: 10.46586/tches.v2021.i1.109-136.

[9] A. Dörflinger, M. Albers, B. Kleinbeck, Y. Guan, H. Michalik, R. Klink, C. Blochwitz, A. Nechi, and M. Berekovic, "A comparative survey of open-source application-class RISC-V processor implementations," in *Proc. 18th ACM Int. Conf. Comput. Frontiers*, New York, NY, USA, May 2021, pp. 12–20, doi: 10.1145/3457388.3458657.

[10] I. Elsadek and E. Y. Tawfik, "RISC-V resource-constrained cores: A survey and energy comparison," in *Proc. 19th IEEE Int. New Circuits Syst. Conf. (NEWCAS)*, Toulon, France, Jun. 2021, pp. 1–5, doi: 10.1109/NEWCAS50681.2021.9462781.

[11] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann, 2017. [Online]. Available: https://dl.acm.org/doi/book/10.5555/3153875

[12] A. Waterman, Y. Lee, R. Avizienis, D. A. Patterson, and K. Asanovic, "The RISC-V instruction set manual volume II: Privileged architecture version 1.9," Dept. EECS, Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2016-129, Jul. 2016. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-129.html

[13] Semico Research & Consulting Group. (2019). *RISC-V Market Analysis: The New Kid on the Block*. [Online]. Available: https://semico.com/content/risc-v-market-analysis-new-kid-block

[14] M. Larabel. (2016). NVIDIA is building its next-gen falcon controller using RISC-V. Phoronix. [Online]. Available: https://www.phoronix.com/scan.php?page=news_item&px=NVIDIA-RISC-V-Next-Gen-Falcon

[15] A. Shilov. (2021). Western digital reveals SweRV RISC-V core, cache coherency over Ethernet initiative. AnandTech. [Online]. Available: https://www.anandtech.com/show/13678/western-digital-reveals-swerv-risc-v-core-and-omnixtend-coherency-tech

[16] J. Hsu. (2021). RISC-V star rises among chip developers worldwide. IEEE Spectrum. [Online]. Available: https://spectrum.ieee.org/tech-talk/semiconductors/design/riscv-rises-among-chip-developers-worldwide

[17] C. Chen, X. Xiang, C. Liu, Y. Shang, R. Guo, D. Liu, Y. Lu, Z. Hao, J. Luo, Z. Chen, C. Li, Y. Pu, J. Meng, X. Yan, Y. Xie, and X. Qi, "Xuantie-910: A commercial multi-core 12-stage pipeline out-of-order 64-bit high performance RISC-V processor with vector extension : Industrial product," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit. (ISCA)*, Valencia, Spain, May 2020, pp. 52–64, doi: 10.1109/ISCA45697.2020.00016.

[18] A. Shilov. (2021). Huawei's HiSilicon develops first RISC-V design to overcome ARM restrictions. Tom's Hardware. [Online]. Available: https://www.tomshardware.com/news/huaweis-hisilicon-develops-first-risc-v-design-to-overcome-arm-restrictions

[19] J. Andersson, "Development of a NOEL-V RISC-V SoC targeting space applications," in *Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2020, pp. 66–67, doi: 10.1109/DSN-W50199.2020.00020.

[20] S. Shankland. (2021). Apple shows interest in RISC-V chips, a competitor to iPhones' ARM tech. CNet. [Online]. Available: https://www.cnet.com/tech/mobile/apple-shows-interest-in-risc-v-chips-a-competitor-to-iphones-arm-tech/

[21] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Gurkaynak, and L. Benini, "Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2700–2713, Oct. 2017, doi: 10.1109/TVLSI.2017.2654506.

[22] PULP Platform. (2021). *PULP Users. Pulp Platform*. [Online]. Available: https://www.pulp-platform.org/pulp_users.html

[23] J. O. Mixon. (Nov. 2019). *Semico Forecasts Strong Growth for RISC-V*. [Online]. Available: https://riscv.org/announcements/2019/11/9679/

[24] F. Gomez, M. Masmano, V. Nicolau, J. Andersson, J. Le Rhun, D. Trilla, F. Gallego, G. Cabo, and J. Abella Ferrer, "De-RISC—Dependable Real-Time Infrastructure For Safety-Critical Computer Systems," *Ada User J.*, vol. 41, no. 2, pp. 12–107, Jun. 2020. [Online]. Available: http://hdl.handle.net/2117/341317

[25] C. Hernandez, J. Flieh, R. Paredes, C.-A. Lefebvre, I. Allende, J. Abella, D. Trillin, M. Matschnig, B. Fischer, K. Schwarz, J. Kiszka, M. Ronnback, J. Klockars, N. McGuire, F. Rammerstorfer, C. Schwarzl, F. Wartet, D. Ludemann, and M. Labayen, "SELENE: Self-monitored dependable platform for high-performance safety-critical systems," in *Proc. 23rd Euromicro Conf. Digit. Syst. Design (DSD)*, Kranj, Slovenia, Aug. 2020, pp. 370–377, doi: 10.1109/DSD51259.2020.00066.

[26] S. Di Mascio, A. Menicucci, G. Furano, C. Monteleone, and M. Ottavi, "The case for RISC-V in space," in *Applications in Electronics Pervading Industry, Environment and Society*, S. Saponara and A. De Gloria, Eds. Springer, Cham, Switzerland, 2019, pp. 319–325, doi: 10.1007/978-3-030-11973-7_37.

[27] Grand View Research. (Jul. 2021). *Industrial Internet of Things Market Worth $1.11 Trillion by 2028*. [Online]. Available: https://www.grandviewresearch.com/press-release/global-industrial-internet-of-things-iiot-market

[28] Grand View Research. (Jun. 2021). *Industrial Internet of Things Market Size Report*. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/industrial-internet-of-things-iiot-market

[29] K. Boeckl, K. Boeckl, M. Fagan, W. Fisher, N. Lefkovitz, K. N. Megas, E. Nadeau, D. G. O'Rourke, B. Piccarreta, and K. Scarfone, *Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks*, Standard NISTIR 8228, United States Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2019, doi: 10.6028/NIST.IR.8228.

[30] L. Auer, C. Skubich, and M. Hiller, "A security architecture for RISC-V based IoT devices," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Florence, Italy, 2019, pp. 1154–1159, doi: 10.23919/DATE.2019.8714822.

[31] M. Malenko and M. Baunach, "Hardware/software co-designed peripheral protection in embedded devices," in *Proc. IEEE Int. Conf. Ind. Cyber Phys. Syst. (ICPS)*, Taipei, Taiwan, May 2019, pp. 790–795, doi: 10.1109/ICPHYS.2019.8780325.

[32] V. B. Y. Kumar, N. Gupta, A. Chattopadhyay, M. Kasper, C. Kraus, and R. Niederhagen, "Post-quantum secure boot," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 1582–1585, doi: 10.23919/DATE48585.2020.9116252.

[33] Z. Liu, J. Jiang, G. Lei, K. Chen, B. Qin, and X. Zhao, "A heterogeneous processor design for CNN-based AI applications on IoT devices," *Proc. Comput. Sci.*, vol. 174, pp. 2–8, Jan. 2020, doi: 10.1016/j.procs.2020.06.048.

[34] SiFive. (2021). *SiFive Intelligence*. [Online]. Available: https://www.sifive.com/cores/intelligence

[35] J. Lee, H. Chen, J. Young, and H. Kim, "RISC-V FPGA platform toward ROS-based robotics application," in *Proc. 30th Int. Conf. Field-Program. Log. Appl. (FPL)*, Gothenburg, Sweden, Aug. 2020, p. 370, doi: 10.1109/FPL50879.2020.00075.

[36] PULP Platform. (Jun. 2017). *PULPino: Datasheet*. [Online]. Available: https://pulp-platform.org/docs/pulpino_datasheet.pdf

[37] N. A. Said, M. Benabdenbi, and K. Morin-Allory, "FPU bit-width optimization for approximate computing: A non-intrusive approach," in *Proc. 15th Design Technol. Integr. Syst. Nanosc. Era (DTIS)*, Marrakesh, Morocco, Apr. 2020, pp. 1–6, doi: 10.1109/DTIS48698.2020.9080931.

[38] J. Castro-Godinez, M. Shafique, and J. Henkel, "Towards quality-driven approximate software generation for accurate hardware: Work-in-progress," in *Proc. Int. Conf. Compil., Archit., Synth. Embedded Syst. (CASES)*, Shanghai, China, Sep. 2020, pp. 12–14, doi: 10.1109/CASES51649.2020.9243814.

[39] 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," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 11, pp. 2629–2640, Nov. 2019, doi: 10.1109/TVLSI.2019.2926114.

[40] LowRISC. (2021). *Ibex RISC-V Core*. [Online]. Available: https://github.com/lowRISC/ibex

[41] OpenHW Group. (2021). *OpenHW Group CORE-V CV32E40P RISC-V IP*. [Online]. Available: https://github.com/openhwgroup/cv32e40p

[42] J. Zhao, B. Korpan, A. Gonzalez, and K. Asanovic, "SonicBOOM: The 3rd generation Berkeley out-of-order machine," in *Proc. 4th Workshop Comput. Archit. Res. RISC-V*, vol. 5, 2020.

[43] BOOM. (2021). *The Berkeley Out-of-Order RISC-V Processor*. [Online]. Available: https://github.com/riscv-boom/riscv-boom

[44] Cobham Gaisler. (2021). *NOEL-V Processor*. [Online]. Available: https://www.gaisler.com/index.php/products/processors/noel-v

[45] PULP Platform. (Mar. 2021). *PULPissimo: Datasheet*. [Online]. Available: https://raw.githubusercontent.com/pulp-platform/pulpissimo/master/doc/datasheet/datasheet.pdf

[46] D. A. Santos, L. M. Luza, L. Dilillo, C. A. Zeferino, and D. R. Melo, "Reliability analysis of a fault-tolerant RISC-V system-on-chip," *Microelectron. Rel.*, vol. 125, Oct. 2021, Art. no. 114346, doi: 10.1016/j.microrel.2021.114346.

[47] SiFive. (2021). *SiFive Performance P550 Core Sets New Standard as Highest Performance RISC-V Processor IP*. [Online]. Available: https://www.sifive.com/press/sifive-performance-p550-core-sets-new-standard-as-highest

[48] M. Larabel. (2021). SiFive announces the performance P550 as the fastest RISC-V processor yet. Phoronix. [Online]. Available: https://www.phoronix.com/scan.php?page=news_item&px=SiFive-Performance-P550-P270

[49] Bloomberg. (2021). *SiFive Performance P550 Core Sets New Standard as Highest Performance RISC-V Processor IP*. [Online]. Available: https://www.bloomberg.com/press-releases/2021-06-22/sifive-performance-p550-core-sets-new-standard-as-highest-performance-risc-v-processor-ip

[50] SiFive. (Oct. 2017). *SiFive FE310-G000 Manual*. [Online]. Available: https://static.dev.sifive.com/FE310-G000.pdf

[51] V. Herdt and R. Drechsler, "Efficient techniques to strongly enhance the virtual prototype based design flow," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Limassol, Cyprus, Jul. 2020, pp. 182–187, doi: 10.1109/ISVLSI49217.2020.00041.

[52] PULP Platform. (Feb. 2019). *PULP Hardware Reference Manual*. [Online]. Available: https://raw.githubusercontent.com/pulp-platform/pulp/master/doc/datasheet.pdf

[53] PULP Platform. (2021). *HERO Documentation*. [Online]. Available: https://pulp-platform.org/hero.html

[54] Microsemi. (Dec. 2019). *PolarFire SoC Advanced Datasheet*. [Online]. Available: https://www.microsemi.com/document-portal/doc_download/1244583-polarfire-soc-advance-datasheet

[55] E. Torres-Sanchez, J. Alastruey-Benedé, and E. Torres-Moreno, "Developing an AI IoT application with open software on a RISC-V SoC," in *Proc. XXXV Conf. Design Circuits Integr. Syst. (DCIS)*, Segovia, Spain, 2020, pp. 1–6, doi: 10.1109/DCIS51330.2020.9268645.

[56] QEMU. (2021). *RISC-V System Emulator*. [Online]. Available: https://qemu.readthedocs.io/en/latest/system/target-riscv.html

[57] Gem5. (2021). *About Gem5*. [Online]. Available: https://www.gem5.org/about

[58] RISC-V. (2021). *Spike RISC-V ISA Simulator*. [Online]. Available: https://github.com/riscv/riscv-isa-sim

[59] B. Landers. (2021). *RARS—RISC-V Assembler and Runtime Simulator*. [Online]. Available: https://github.com/TheThirdOne/rars

[60] R. Giorgi and G. Mariotti, "WebRISC-V: A web-based education-oriented RISC-V pipeline simulation environment," in *Proc. Workshop Comput. Archit. Educ.*, New York, NY, USA, 2019, pp. 1–6, doi: 10.1145/3338698.3338894.

[61] M. B. Petersen, "Ripes: A visual computer architecture simulator," in *Proc. ACM/IEEE Workshop Comput. Archit. Educ. (WCAE)*, Jun. 2021, pp. 1–8, doi: 10.1109/WCAE53984.2021.9707149.

[62] A. Castellanos. (2019). *RISC-V Assembler and Runtime Simulator*. [Online]. Available: https://github.com/andrescv/Jupiter

[63] V. M. de Morais Costa. (2022). *RISC-V Instruction Set Simulator (Built for Education)*. [Online]. Available: https://github.com/vmmc2/Vulcan

[64] G. Savaton. (2022). *A Visual Simulator for Teaching Computer Architecture Using the RISC-V Instruction Set*. [Online]. Available: https://github.com/Guillaume-Savaton-ESEO/emulsiV

[65] W. Stallings, *Operating Systems: Internals and Design Principles*, 9th ed. Indianapolis, IN, USA: Pearson, 2018. [Online]. Available: https://www.pearson.com/us/higher-education/program/Stallings-Operating-Systems-Internals-and-Design-Principles-9th-Edition/PGM1262980.html

[66] B. James, H. Quinn, M. Wirthlin, and J. Goeders, "Applying compiler-automated software fault tolerance to multiple processor platforms," *IEEE Trans. Nucl. Sci.*, vol. 67, no. 1, pp. 321–327, Jan. 2020, doi: 10.1109/TNS.2019.2959975.

[67] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed. Reading, MA, USA: Addison-Wesley, 2011. [Online]. Available: https://dl.acm.org/doi/10.5555/2029110

[68] M. Bohman, B. James, M. J. Wirthlin, H. Quinn, and J. Goeders, "Microcontroller compiler-assisted software fault tolerance," *IEEE Trans. Nucl. Sci.*, vol. 66, no. 1, pp. 223–232, Jan. 2019, doi: 10.1109/TNS.2018.2886094.

[69] R. Stahl, D. Mueller-Gritschneder, and U. Schlichtmann, "Driver generation for IoT nodes with optimization of the hardware/software interface," *IEEE Embedded Syst. Lett.*, vol. 12, no. 2, pp. 66–69, Jun. 2020, doi: 10.1109/LES.2019.2948264.

[70] Q. Zhang, J. Qiao, Q. Meng, and Y. Chen, "Automatic kernel code synthesis and verification," *Comput. Secur.*, vol. 91, Apr. 2020, Art. no. 101733, doi: 10.1016/j.cose.2020.101733.

[71] M. N. Ince, J. Ledet, and M. Gunay, "Building an open source Linux computing system on RISC-V," in *Proc. 1st Int. Informat. Softw. Eng. Conf. (UBMYK)*, Ankara, Turkey, Nov. 2019, pp. 1–4, doi: 10.1109/UBMYK48245.2019.8965559.

[72] C. Dietrich and D. Lohmann, "OSEK-V: Application-specific RTOS instantiation in hardware," *SIGPLAN Notices*, vol. 52, no. 5, pp. 111–120, Jun. 2017, doi: 10.1145/3140582.3081030.

[73] M. Malenko and M. Baunach, "Device driver and system call isolation in embedded devices," in *Proc. 22nd Euromicro Conf. Digit. Syst. Design (DSD)*, Kallithea, Greece, Aug. 2019, pp. 283–290, doi: 10.1109/DSD.2019.00049.

[74] I. Wali, A. Sanchez-Macian, A. Ramos, and J. A. Maestro, "Analyzing the impact of the operating system on the reliability of a RISC-V FPGA implementation," in *Proc. 27th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Glasgow, U.K., Nov. 2020, pp. 1–4, doi: 10.1109/ICECS49266.2020.9294858.

[75] B. Mezger, F. Bortoluzzi, C. A. Zeferino, P. R. O. Valim, and D. R. Melo, "A basic microkernel for the RISC-V instruction set architecture," *Anais Comput. Beach*, vol. 12, pp. 057–063, Apr. 2021, doi: 10.14210/cotb.v12.p057-063.

[76] S. Savas, Z. Ul-Abdin, and T. Nordström, "A framework to generate domain-specific manycore architectures from dataflow programs," *Microprocessors Microsyst.*, vol. 72, Feb. 2020, Art. no. 102908, doi: 10.1016/j.micpro.2019.102908.

[77] X. Wang, M. Magno, L. Cavigelli, and L. Benini, "FANN-on-MCU: An open-source toolkit for energy-efficient neural network inference at the edge of the Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4403–4417, May 2020, doi: 10.1109/JIOT.2020.2976702.

[78] C. Trippel, Y. A. Manerkar, D. Lustig, M. Pellauer, and M. Martonosi, "TriCheck: Memory model verification at the trisection of software, hardware, and ISA," in *Proc. 22nd Int. Conf. Architectural Support Program. Lang. Oper. Syst.*, New York, NY, USA, 2017, pp. 119–133, doi: 10.1145/3037697.3037719.

[79] A. S. Tanenbaum and H. Bos, *Modern Operating Systems*, 4th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2014. [Online]. Available: https://dl.acm.org/doi/10.5555/2655363

[80] D. Du, Z. Hua, Y. Xia, B. Zang, and H. Chen, "XPC: Architectural support for secure and efficient cross process call," in *Proc. ACM/IEEE 46th Int. Symp. Comput. Architecture*, Phoenix, AZ, USA, Jun. 2019, pp. 671–684, doi: 10.1145/3307650.3322218.

[81] L. Kohútka and V. Stopjaková, "Novel efficient on-chip task scheduler for multi-core hard real-time systems," *Microprocessors Microsyst.*, vol. 76, Jul. 2020, Art. no. 103083, doi: 10.1016/j.micpro.2020.103083.

[82] M. Naylor, S. W. Moore, and D. Thomas, "Tinsel: A manythread overlay for FPGA clusters," in *Proc. 29th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2019, pp. 375–383, doi: 10.1109/FPL.2019.00066.

[83] D. Schrammel, S. Weiser, S. Steinegger, M. Schwarzl, M. Schwarz, S. Mangard, and D. Gruss, "Efficient in-process isolation for RISC-V and ×86," in *Proc. USENIX Security*. Boston, MA, USA: USENIX Association, Aug. 2020, pp. 1677–1694. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/schrammel

[84] J. Liu, Y. Qin, and D. Feng, "SeRoT: A secure runtime system on trusted execution environments," in *Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Guangzhou, China, Dec. 2020, pp. 30–37, doi: 10.1109/TrustCom50675.2020.00018.

[85] A. Bolat, L. Cassano, P. Reviriego, O. Ergin, and M. Ottavi, "A microprocessor protection architecture against hardware trojans in memories," in *Proc. 15th Design Technol. Integr. Syst. Nanosc. Era (DTIS)*, Marrakesh, Morocco, Apr. 2020, pp. 1–6, doi: 10.1109/DTIS48698.2020.9080961.

[86] T. Hunt, Z. Jia, V. Miller, C. J. Rossbach, and E. Witchel, "Isolation and beyond: Challenges for system security," in *Proc. Workshop Hot Topics Oper. Syst.*, New York, NY, USA, May 2019, pp. 96–104, doi: 10.1145/3317550.3321427.

[87] D. Hwang, M. Yang, S. Jeon, Y. Lee, D. Kwon, and Y. Paek, "RiskiM: Toward complete kernel protection with hardware support," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Florence, Italy, Mar. 2019, pp. 740–745, doi: 10.23919/DATE.2019.8715277.

[88] D. Šišejković, F. Merchant, L. M. Reimann, R. Leupers, M. Giacometti, and S. Kegreiß, "A secure hardware-software solution based on RISC-V, logic locking and microkernel," in *Proc. 23th Int. Workshop Softw. Compil. Embedded Syst.*, New York, NY, USA, May 2020, pp. 62–65, doi: 10.1145/3378678.3391886.

[89] M. Fischer, F. Langer, J. Mono, C. Nasenberg, and N. Albartus, "Hardware penetration testing knocks your SoCs off," *IEEE Design Test*, vol. 38, no. 1, pp. 14–21, Feb. 2021, doi: 10.1109/MDAT.2020.3013730.

[90] T.-T. Hoang, C. Duran, D.-T. Nguyen-Hoang, D.-H. Le, A. Tsukamoto, K. Suzaki, and C.-K. Pham, "Quick boot of trusted execution environment with hardware accelerators," *IEEE Access*, vol. 8, pp. 74015–74023, 2020, doi: 10.1109/ACCESS.2020.2987617.

[91] A. S. Siddiqui, G. Shirley, S. Bendre, G. Bhagwat, J. Plusquellic, and F. Saqib, "Secure design flow of FPGA based RISC-V implementation," in *Proc. IEEE 4th Int. Verification Secur. Workshop (IVSW)*, Rhodes, Greece, Jul. 2019, pp. 37–42, doi: 10.1109/IVSW.2019.8854418.

[92] G. Dessouky, D. Gens, P. Haney, G. Persyn, A. Kanuparthi, H. Khattri, J. M. Fung, A.-R. Sadeghi, and J. Rajendran, "HardFails: Insights into software-exploitable hardware bugs," in *Proc. USENIX Security*. Santa Clara, CA, USA: USENIX Association, Aug. 2019, pp. 213–230. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/dessouky

[93] R. Dofferhoff, M. Goebel, K. Rietveld, and E. van der Kouwe, "IScanU: A portable scanner for undocumented instructions on RISC processors," in *Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Valencia, Spain, Jun. 2020, pp. 306–317, doi: 10.1109/DSN48063.2020.00047.

[94] A. Fell, H. T. Pham, and S.-K. Lam, "TAD: Time side-channel attack defense of obfuscated source code," in *Proc. 24th Asia South Pacific Design Autom. Conf.*, New York, NY, USA, Jan. 2019, pp. 58–63, doi: 10.1145/3287624.3287694.

[95] T. Nyman, G. Dessouky, S. Zeitouni, A. Lehikoinen, A. Paverd, N. Asokan, and A.-R. Sadeghi, "HardScope: Hardening embedded systems against data-oriented attacks," in *Proc. 56th ACM/IEEE Annu. Design Autom. Conf.*, Las Vegas, NV, USA, Jun. 2019, pp. 1–6, doi: 10.1145/3316781.3317836.

[96] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann, 2021, doi: 10.1016/B978-0-12-818105-8.00011-5.

[97] J. Laurent, V. Beroulle, C. Deleuze, F. Pebay-Peyroula, and A. Papadimitriou, "Cross-layer analysis of software fault models and countermeasures against hardware fault attacks in a RISC-V processor," *Microprocessors Microsyst.*, vol. 71, Nov. 2019, Art. no. 102862, doi: 10.1016/j.micpro.2019.102862.

[98] J. Laurent, V. Beroulle, C. Deleuze, and F. Pebay-Peyroula, "Fault injection on hidden registers in a RISC-V rocket processor and software countermeasures," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Florence, Italy, Mar. 2019, pp. 252–255, doi: 10.23919/DATE.2019.8715158.

[99] P. Adelt, B. Koppelmann, W. Mueller, and C. Scheytt, "Register and instruction coverage analysis for different RISC-V ISA modules," in *Proc. 24th GMM/ITG/GI Workshop Methods Description Lang. Modeling Verification Circuits Syst. (MBMV)*. Frankfurt, Germany: VDE, 2021, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9399723

[100] P. Adelt, B. Koppelmann, W. Mueller, and C. Scheytt, "A scalable platform for QEMU based fault effect analysis for RISC-V hardware architectures," in *Proc. 23rd GMM/ITG/GI Workshop Methods Description Lang. Modeling Verification Circuits Syst. (MBMV)*, Stuttgart, Germany, 2020, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9094540

[101] C. De Sio, S. Azimi, A. Portaluri, and L. Sterpone, "SEU evaluation of hardened-by-replication software in RISC-V soft processor," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Athens, Greece, Oct. 2021, pp. 1–6, doi: 10.1109/DFT52944.2021.9568342.

[102] C. Imianosky, P. R. O. Valim, C. A. Zeferino, and F. Viel, "Evaluating the CCSDS 123 compressor running on RISC-V and ARM architectures," in *Proc. X Brazilian Symp. Comput. Syst. Eng. (SBESC)*, Florianopólis, Brazil, Nov. 2020, pp. 1–7, doi: 10.1109/SBESC51047.2020.9277854.

[103] I. Karageorgos, K. Sriram, J. Vesely, M. Wu, M. Powell, D. Borton, R. Manohar, and A. Bhattacharjee, "Hardware-software co-design for brain-computer interfaces," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit. (ISCA)*, Valencia, Spain, May 2020, pp. 391–404, doi: 10.1109/ISCA45697.2020.00041.

BENJAMIN W. MEZGER (Student Member, IEEE) received the B.S. degree in computer science from the University of Vale do Itajaí, in 2019, where he is currently the M.S. degree in applied computer science. His research interests include RISC-V, operating systems, and fault tolerance.

DOUGLAS A. SANTOS (Student Member, IEEE) received the B.E. degree in computer engineering and the M.S. degree in applied computer science from the University of Vale do Itajaí, in 2019 and 2021, respectively. He is currently pursuing the Ph.D. degree with the Laboratoire d'Informatique, de Robotique et de Microelectronique de Montpellier (LIRMM). His research interests include RISC-V, embedded systems, and fault tolerance.

LUIGI DILILLO (Member, IEEE) received the Diploma degree in electronic engineering from the Politecnico di Torino, Italy, in 2001, and the Ph.D. degree in microelectronics from the University of Montpellier. He is currently a CNRS Researcher with the LIRMM Laboratory. His research interests include memory test and reliability, power-aware test, radiation impact on electronics, radiation monitoring, and space and radiation-hardened system design.

CESAR A. ZEFERINO (Member, IEEE) received the Ph.D. degree in computer science from the Federal University of Rio Grande do Sul, Brazil, in 2003. He is currently a Full Professor and the Director of the School of Sea, Science and Technology, University of Vale do Itajaí—Univali, Brazil, where he also heads the Laboratory of Embedded and Distributed Systems. His research interests include digital systems design, embedded systems, networks-on-chip, and hardware acceleration.

DOUGLAS R. MELO (Member, IEEE) received the B.E. degree in computer engineering and the M.S. degree in applied computer science from the University of Vale do Itajaí, in 2008 and 2012, respectively, and the Ph.D. degree in electrical engineering from the Federal University of Santa Catarina, in 2020. He is currently an Adjunct Professor with the University of Vale do Itajaí and a Researcher with the Laboratory of Embedded and Distributed Systems. His research interests include systems-on-chip, networks-on-chip, and fault tolerance.