2025 38th SBC/SBMicro/IEEE Symposium on Integrated Circuits and Systems Design (SBCCI) | 979-8-3315-5555-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/SBCCI66862.2025.11218649

RS5-SoC: A Flexible Open-Source RISC-V Platform for Embedded Systems

Rafael Faccenda, Willian Nunes, Angelo Dal Zotto, Carlos Gewehr, Lucas Luza,
Lucas Damo, Vitor Zanini, Eduardo Bernardon, Vinícius Mibielli, Nathan Cidal, Fernando Gehm Moraes
School of Technology, Pontifical Catholic University of Rio Grande do Sul – PUCRS – Porto Alegre, Brazil
rafael.faccenda@edu.pucrs.br, fernando.moraes@pucrs.br

***Abstract*—Open-source System-on-Chip (SoC) platforms with flexible, software-supported RISC-V processors are required for embedded system development. This paper introduces RS5-SoC, the digital part of an inter-institutional research-driven SoC named WiMED SoC. Contributions of this platform include its flexible peripheral integration capabilities, support for the open-source multitasking Zephyr OS, and processor configurability through customizable instruction set extensions (ISE), which incorporate native support for encryption and atomic instructions. The paper details the hardware and software components of the SoC digital components and discusses their implementation in FPGA and ASIC environments.**

***Index Terms*—RISCV-V, SoC, Embedded Systems, RTOS.**

I. Introduction and Related Work

Open-source System-on-Chip (SoC) platforms with flexible, software-supported RISC-V processors are required for embedded system development in FPGA and ASIC environments. Open-source RISC-V microcontrollers (MCUs) are increasingly used in applications like IoT devices, real-time control systems, and *wearables* which can measure and monitor vital signs. These typically feature 32-bit RISC-V cores, on-chip SRAM, and standard peripherals, emphasizing low power consumption and ease of development while supporting bare-metal or real-time operating systems.

For instance, Chiu et al. [1] focus on balancing low-latency and low-power requirements in real-time edge computing by employing a RISC-V CVA6 processor integrated with custom hardware accelerators on the ESP SoC platform [2]. Using the Eigen C++ library (linear-algebra workloads) and specialized accelerator tiles for general matrix multiplication, their platform demonstrated performance and energy efficiency improvements, achieving up to 57x performance gain and 125x better energy efficiency than software-only solutions.

Similarly, Azad et al. [3] propose a RISC-V-based SoC designed to optimize encryption and decryption processes for homomorphic encryption in edge computing contexts. This platform integrated a unified encryption/decryption datapath and memory optimization techniques, resulting in energy efficiency improvements ranging from 48x to nearly 39,729x compared to traditional processor-only systems.

Addressing virtualized environments, Sá et al. [4] introduce enhancements to the RISC-V CVA6 core, integrating RISC-V Hypervisor extensions. Their contributions included architectural modifications such as additional execution modes (HS-mode and VS-mode) and a Memory Management Unit for handling two-stage address translations, validated through FPGA prototyping and post-layout simulations in 22nm technology.

Expanding on configurability and peripheral integration, Machetti et al. [5] develop X-HEEP, a configurable, low-power platform based on CORE-V processors such as CV32E20 and CV32E40X/P. This platform introduced the eXtendible Accelerator Interface (XAIF), which integrates specialized accelerators. Their experiments with healthcare-oriented accelerators demonstrated energy efficiency improvements of up to approximately 4.9x over CPU-only implementations.

Wang et al. [6] present an open-source framework optimized for energy-efficient neural network inference on IoT microcontrollers. Their toolkit supports ARM Cortex-M and RISC-V PULP processors, integrating standard MCU peripherals and hardware accelerators tailored for neural network execution.

In the industry and academia, platforms like SiFive FE310 [7], CORE-V MCU from OpenHW [8], Shakti RIMO [9], and OpenTitan [10] have emerged, providing reference designs. SiFive FE310 is a low-cost MCU reference design due to its open-source RTL, simplicity, and compatibility with RTOS software. CORE-V MCU highlighted peripheral extensibility through embedded FPGAs, targeting IoT and edge machine learning applications. Meanwhile, Shakti RIMO offered an open-source, 64-bit processor for embedded industrial and automotive applications. OpenTitan, emphasizing security, provided a root-of-trust MCU that integrates cryptographic accelerators and secure firmware execution capabilities for embedded security applications.

This paper introduces RS5-SoC, the digital block of the *WiMED SoC* (Wireless SoC for Medical Monitoring of Vital Signs, focusing on Security and Low Power Consumption). The *WiMED SoC* is an inter-institutional research-driven platform designed for biomedical data acquisition, using an RS5 [11] RISC-V core, and secure data transmission via Bluetooth Low Energy (BLE). The contributions of RS5-SoC include its flexible peripheral integration, support for the open-source multitasking Zephyr OS, and processor flexibility through customizable instruction set extensions (ISE).

II. RS5-SoC Hardware

Figure 1 presents the architecture of the RS5-SoC. It is important to highlight that all modules were developed in-house, without third-party components. There are five groups of modules:

1) **Computing subsystem**: includes the RS5 processor [11], a dual-port 64 KB RAM, a real-time clock (RTC), and a Platform-Level Interrupt Controller (PLIC);
2) **MCU peripherals**: GPIO, QSPI, SPI, and UART modules;

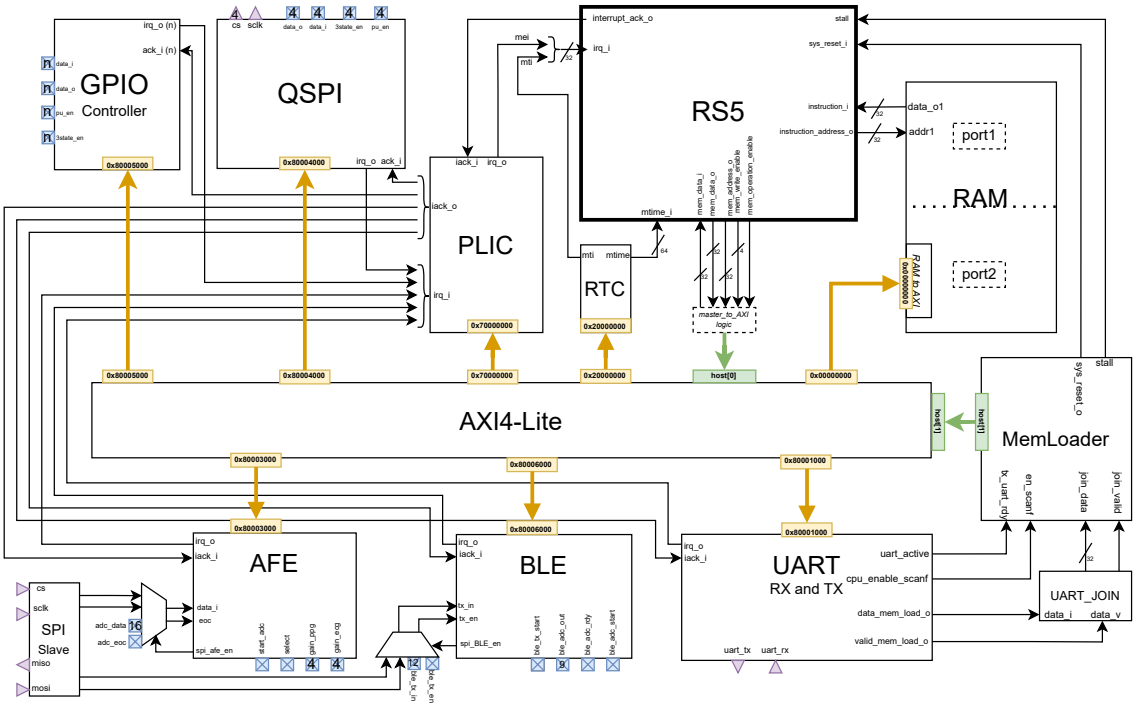

Fig. 1. RS5-SoC block diagram. In the AXI4-Lite bus, green boxes correspond to master interfaces, and yellow boxes correspond to slave interfaces.

3) **WiMED SoC-related peripherals**: analog front-end (AFE), which receives data from ADCs, and the BLE module, which is the interface with the Bluetooth block;
4) **Multi-master AXI4-Lite bus** [12]: the processor and the MemLoader can act as bus masters, with the MemLoader assigned a higher priority;
5) **MemLoader**: receives commands from an external host processor via the UART module.

RS5 [11] is a RISC-V processor supporting Machine and User privilege levels. Table I shows the RS5 top-level parameters. The RS5-SoC adopts the RV32IMAC ZIHPM ZKNE ISA, a 32-bit **I**nteger ISA extended with: (i) **M**: multiplication and division operations; (ii) **A**: atomic operations, which enable the implementation of software-level concurrency through read–modify–write instructions; (iii) **C**: compressed instructions, which reduce the memory footprint; (iv) ZIHPM: standard extension for hardware performance counters; (v) ZKNE: supports the execution of the $aes32esi$ and $aes32esmi$ instructions [13], [14]. These extensions enable the processor to run multitasking operating systems and provide support for hardware-accelerated cryptographic operations.

RS5 adopts an interrupt treatment approach based on a proposal by SiFive [15]: **PLIC**. PLIC manages global or external interrupts generated by peripheral devices. Within PLIC, each interrupt can be assigned a configurable priority level ranging from 1 to 7 (highest priority), and 0 disables the interrupt. The PLIC is configurable and provides an interface to Memory-Mapped Registers (MMR).

RS5 supports timer interrupts through a **RTC** and external interrupts managed by PLIC. The RTC is a 64-bit cycle counter, accessible through MMR and also routed to the **time** counter in Zicntr (ZIHPM register). The OS uses the RTC to generate timer interrupts to control the scheduling time slice.

TABLE I
RS5 DESIGN PARAMETERS.

Parameter	Description	Options
Environment	Environment type	ASIC, FPGA
MULEXT	Include Hardware Multiplication/Division extension	MUL_OFF, MUL_ZMMUL, MUL_M
AMOEXT	Include Atomic operation extension	AMO_OFF, AMO_ZALRSC, AMO_ZAAMO, AMO_A
COMPRESSED	Include Compressed extension	TRUE, FALSE
XOSVMEnable	Include XOSVM extension (MMU)	TRUE, FALSE
ZIHPMEnable	Include ZIHPM extension (Performance Monitors)	TRUE, FALSE
ZKNEEnable	Include ZKNE extension (AES Hardware acceleration)	TRUE, FALSE
BRANCHPRED	Include Branch prediction	TRUE, FALSE

The **peripherals** (both MCU and WiMED-related) are designed as AXI4-Lite slaves, controlled through MMRs. These peripherals use interruptions via PLIC to send data to the processor. The OS drivers manage the MMRs and handle the corresponding interruptions.

The **MemLoader** module acts as a "command interpreter" for instructions received via UART (the SPI interface is planned to be a backup communication channel). Its primary function is to receive a binary code from a host computer and write it to memory. Once loading is complete, the MemLoader activates the processor to execute the loaded programs. In future versions of the SoC, a bootloader will be included to load data from a flash memory connected through the QSPI peripheral. As a master of the AXI4 bus, the MemLoader also operates as a "universal debugger", since it can read from and write to both memory and peripherals. This enables memory dumps and data to be written across any address range.

III. RS5-SoC Software

The software layer implements drivers, linker scripts, a hardware abstraction layer (HAL), and libraries to use previously described hardware resources. This platform supports two software options: bare-metal or Zephyr RTOS.

A. Bare-metal

Bare-metal programming includes only the essential features required to use the RS5-SoC's core resources and to facilitate testing and debugging. A custom library, called *libsoc*, provides the support features for bare-metal programming. *Libsoc* includes drivers for each peripheral, whether an MCU peripheral or a WiMED peripheral, as well as other functionalities such as interrupt handling and AES and SHA algorithms adapted from TinyCrypt [16].

The bare-metal support uses newlib-nano as its standard C library (*libc*), designed for embedded SoCs, balancing performance and a memory footprint. Newlib-nano provides essential *libc* features such as input/output operations, memory management, and string handling, while *libsoc* implements the *libc* interface with RS5-SoC.

However, a multitasking OS and a more robust software layer are required to deploy the RS5-SoC in complex scenarios and support sophisticated applications.

B. Zephyr RTOS

Recent works [17] employ FreeRTOS, which, despite its widespread adoption, faces scalability, modularity, and security limitations. Alternatively, Zephyr [18] offers a flexible, scalable architecture with active community support, targeting diverse hardware platforms across IoT, automotive, and industrial domains.

Zephyr is an open-source, portable, scalable, and power-efficient real-time operating system [18]. It provides a comprehensive development environment, including toolchain and HAL management, debugging, and testing capabilities. Zephyr's architecture separates application logic from hardware dependencies, treating them as distinct entities: applications and boards. It supports over 750 pre-configured boards, ensuring compatibility with a wide array of devices, sensors, and hardware architectures (ARM, Intel, RISC-V, MIPS).

This extensive hardware support includes drivers for RS5-SoC modules, such as PLIC and GPIO, which are compliant with SiFive implementations and available within the Zephyr ecosystem. Additionally, drivers for RTC and BLE modules are provided. Zephyr's toolchain also supports various RISC-V extensions used by the RS5-SoC, except for the Zkne extension.

C. Porting Zephyr to RS5-SoC

Porting Zephyr to a custom SoC requires creating a dedicated board directory with specific configurations, including device trees, pin mappings, and peripheral drivers. This structured approach ensures hardware abstraction while maintaining flexibility, allowing developers to adapt Zephyr to a custom SoC while leveraging its real-time capabilities.

Figure 2 illustrates the required directory structure for building Zephyr with the RS5-SoC as the target hardware. To customize the SoC, three groups of files are required:

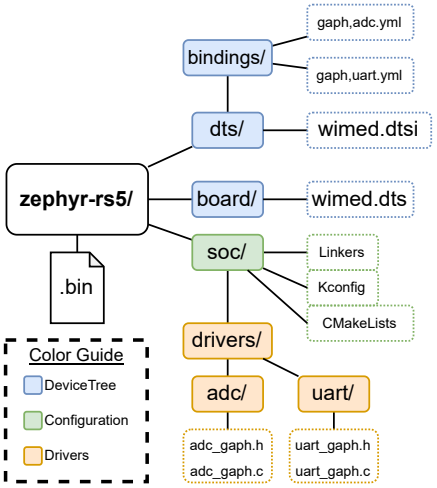

Fig. 2. Custom Zephyr project directory, with the necessary additions.

Device Tree configurations (blue) are specified through two primary input file types: Device Tree Source (DTS) files and Device Tree binding files. The DTS files encode the hierarchical structure and parameters of the hardware configuration, while the binding files formally define the data types and constraints of the referenced modules. The build system synthesizes these inputs to generate a corresponding C header file.

SoC configuration files (green) define general SoC parameters. *Kconfig* specifies design-time resource settings, such as clock frequencies and flags for specific modules; *CMakeLists* defines the libraries to be compiled with the application and more system settings; finally, the *Linker* script determines how the object code is placed into the appropriate memory sections.

Drivers (orange) are the software components that enable the correct utilization of system resources. In this case, Zephyr already provides drivers for the GPIO, PLIC, RTC, and other modules of the RS5-SoC. However, for the UART and AFE modules, it was necessary to develop custom drivers. The following subsection provides further details on integrating custom modules into Zephyr.

The outcome of the Zephyr build process is a binary file, which is loaded onto the platform either through the *MemLoader* using the testbench in simulation scenarios or via a specific command-line interface called *SoC-CLI*, developed by the group to enable communication over the USB connection of the Nexys A7 (FPGA) board.

D. Example Case: Analog Front End (AFE)

The Analog Front End (AFE) peripheral is a custom module designed to interface the digital block of the SoC-WiMED with the analog block responsible for acquiring health-monitoring data. As illustrated in Figure 1, the AFE block is connected to the RS5 processor via the AXI bus and an interrupt line through the PLIC. Within the AFE block, a buffer stores data from sensors monitoring heart activity (ECG) or blood flow characteristics (PPG), and a FSM implements a protocol that triggers an interrupt to the RS5 whenever the configured number of samples has been stored in the buffer.

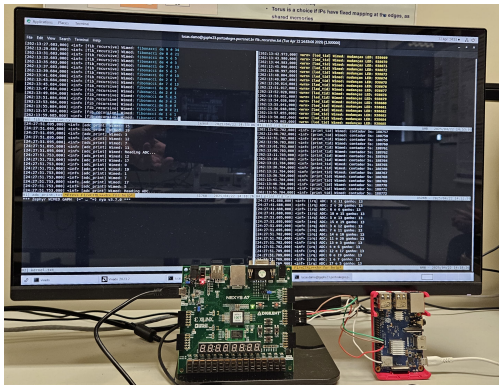

Fig. 3. Prototyping environment to validate the RS5-SoC.

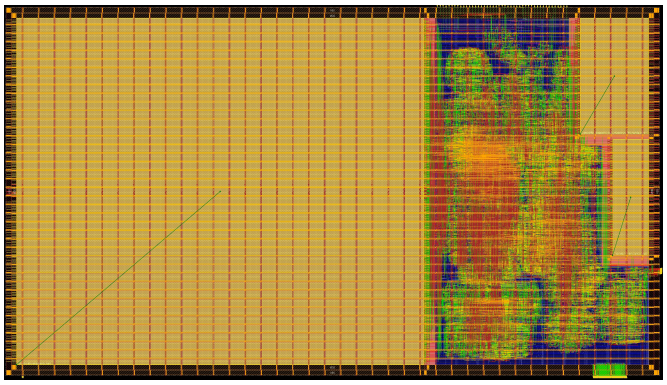

Fig. 4. Layout of the RS5-SoC (600 μm x 350 μm) – TSMC 28nm technology.

To make Zephyr recognize and correctly include the AFE module, the porting directory (Figure 2) now contains: (*i*) *gaph,adc.yml*: the DeviceTree binding file for the AFE; (*ii*) *wimed.dtsi*: the SoC DTS file with the AFE reference and AXI address; (*iii*) *drivers/adc* directory: files with algorithms and necessary code to use the AFE.

When building Zephyr, the DeviceTree header file will include the *gaph-afe* driver to the OS, and this resource will be available to the applications.

IV. Results

A. Memory Footprint

To validate the RS5-SoC platform with Zephyr OS, we developed an application with four threads: one prints encrypted text via UART, another calibrates the ADC and reads values from the Raspberry Pi board, a third runs recursive functions, and the last flashes an LED through the GPIO interface.

Table II shows the application's memory footprint, from the baseline ISA (RV32IM ZIHPM) to RV32IMAC_Zkne. RS5 ISA extensions, particularly the Compressed one, reduced the memory footprint by 22% compared to the baseline ISA. This confirms Zephyr's suitability for embedded systems with limited resources (64 KB for RS5-SoC).

TABLE II
Memory Footprint for each Zephyr Configuration.

Software Parameter	Code Size (B)
Baseline (**RV32IM** + ZIHPM)	49,344
+ ZKNE (RV32IM_ZKne)	48,096
+ Atomic (RV32IM**A**_ZKne)	48,016
+ Compressed (RV32IMA**C**_ZKne)	38,160

B. FPGA Design

The RS5-SoC is described in SystemVerilog, with synthesis pragmas for ASIC or FPGA implementations (*Environment* column in Table I). The primary difference between these descriptions lies in implementing the register bank and memory blocks. We use BRAMs for FPGA memory blocks, whereas memory generators are employed for ASIC designs.

Figure 3 illustrates the debug environment of the SoC prototyped on an Artix FPGA. The environment comprises the Nexys A7 board connected to a host computer and a Raspberry PI board transmitting data to the FPGA board, emulating the AFE module behavior (transmission of ADC samples). The RS5-SoC sends/receives messages to/from the host computer, using the *SoC-CLI*. The debug screen displays the execution of five threads, with Zephyr messages shown in the lower-left corner. This environment enables the validation of the hardware and software of the proposed SoC.

C. ASIC Design

The synthesis flow used Cadence and Siemens tools, following the macro-steps: (1) generation of the memory blocks; (2) floorplanning (Innovus); (3) floorplan-guided logic synthesis (Genus); (4) Logic Equivalence Check (LEC) between the initial RTL and the generated netlist (Conformal); (5) placement and routing (Innovus); (6) LVS and DRC (Calibre). Upon completion of these steps, the layout is exported to GDSII for the final phase of the ASIC design, in which the RS5-SoC interconnects with the pad ring. Table III presents the key parameters obtained after synthesis. Figure 4 illustrates the layout, with 64 KB of dual-port memory on the left, the digital modules in the center, and the FIFOs for the QSPI interface on the top right.

TABLE III
Cadence Innovus synthesis results for TSMC 28nm.

Frequency	Slack	Area (core)	Cell Count	Power
200 MHz	0.132 ns	0.178 mm^2	20,663	9.38 mW

The layout has undergone LVS, DRC and timing verification (*signoff*). SDF files were generated for the slow, typical, and fast corners. Using multi-corner simulations of the generated netlist, the regression tests detected no errors, implying that the RS5-SoC is ready for fabrication.

V. Conclusions and Future Work

This paper presented the RS5-SoC, which targets embedded systems with RISC-V processors. The RS5-SoC contains a set of MCU peripherals, an AXI4-Lite bus, and a specialized MemLoader module for software deployment and debugging. The successful porting of Zephyr RTOS to the RS5-SoC highlights the platform's capability to support multitasking and complex software stacks.

Future work will pursue RISC-V certification for the RS5 processor to ensure compliance with industry standards. Further developments include integrating a bootloader for QSPI flash support, expanding Zephyr's driver ecosystem, and validating the ASIC implementation through tape-out and post-silicon testing.

ACKNOWLEDGMENTS

This work was financed in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), 305621/2024-6; Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS), 23/2551-0002200-1.

REFERENCES

[1] K. Chiu, G. Eichler, B. B. Seyoum, and L. P. Carloni, "EigenEdge: Real-Time Software Execution at the Edge with RISC-V and Hardware Accelerators," in *Cyber-Physical Systems and Internet of Things Week*, 2023, pp. 209–214, https://doi.org/10.1145/3576914.3587510.

[2] ESP, "ESP – the open-source SoC platform," 2025, https://www.esp.cs.columbia.edu/.

[3] Z. Azad, G. Yang, R. Agrawal, D. Petrisko, M. B. Taylor, and A. Joshi, "RACE: RISC-V SoC for En/decryption Acceleration on the Edge for Homomorphic Computation," in *ISLPED*, 2022, pp. 1–6, https://doi.org/10.1145/3531437.3539725.

[4] B. Sá, F. Marques, M. Rodriguez, J. Martins, and S. Pinto, "Holistic RISC-V Virtualization: CVA6-based SoC," in *ACM International Conference on Computing Frontiers*, 2023, pp. 389–390, https://doi.org/10.1145/3587135.3591436.

[5] S. Machetti, P. D. Schiavone, T. C. Müller, M. P. Quirós, and D. Atienza, "X-HEEP: An Open-Source, Configurable and Extendible RISC-V Microcontroller for the Exploration of Ultra-Low-Power Edge Accelerators," *CoRR*, vol. abs/2401.05548, pp. 1–21, 2024, https://doi.org/10.48550/arXiv.2401.05548.

[6] 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 of Things Journal*, vol. 7, no. 5, pp. 4403–4417, 2020, https://doi.org/10.1109/JIOT.2020.2976702.

[7] SiFive, "SiFive Launches Industry's First Open-Source RISC-V SoC," 2016, https://www.sifive.com/press/sifive-launches-industrys-first-open-source-risc-v-soc.

[8] OpenHW, "The CORE-V Family of Open-Source RISC-V Cores," 2024, https://github.com/openhwgroup/core-v-cores.

[9] A. Jadhav, "An Overview of SHAKTI Processor Program," 2020, https://abopen.com/news/an-overview-of-shakti-processor-program.

[10] OpenTitan, "OpenTitan Earl Grey Chip Datasheet," 2024, https://opentitan.org/book/hw/top_earlgrey/doc/specification.html.

[11] W. Nunes, A. Dal Zotto, C. Borges, and F. G. Moraes, "RS5: An Integrated Hardware and Software Ecosystem for RISC-V Embedded Systems," in *IEEE Latin American Symposium on Circuits and Systems (LASCAS)*, 2024, pp. 1–5, https://doi.org/10.1109/LASCAS60203.2024.10506171.

[12] ARM, "AXI4 and AXI4-Lite interfaces," 2025, https://developer.arm.com/documentation/dui0534/b/Parameter-Descriptions/Interface/AXI4-and-AXI4-Lite-interfaces/.

[13] C. Gewehr, N. Moura, L. Luza, E. Bernardon, N. Calazans, R. Garibotti, and F. G. Moraes, "Hardware Acceleration of Authenticated Encryption with Associated Data via RISC-V Instruction Set Extensions in Low Power Embedded Systems," in *LASCAS*, 2024, pp. 1–5, https://ieeexplore.ieee.org/document/10506132.

[14] C. Gewehr and F. G. Moraes, "Improving the Efficiency of Cryptography Algorithms on Resource-Constrained Embedded Systems via RISC-V Instruction Set Extensions," in *SBCCI*, 2023, pp. 1–6, https://doi.org/10.1109/SBCCI60457.2023.10261964.

[15] SiFive, Inc, *SiFive Interrupt Cookbook, Version 1.2*, 2020, https://www.starfivetech.com/uploads/sifive-interrupt-cookbook-v1p2.pdf.

[16] Intel, "TinyCrypt Cryptographic Library," 2017, https://github.com/intel/tinycrypt.

[17] A. I. Silva, A. Susin, F. L. Kastensmidt, A. C. S. Beck, and J. R. Azambuja, "NoX: a Compact Open-Source RISC-V Processor for Multi-Processor Systems-on-Chip," in *Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2024, pp. 1–5, https://doi.org/10.1109/SBCCI62366.2024.10703976.

[18] Zephyr, "Zephyr Project RTOS," 2025, https://www.zephyrproject.org/.