POLITECNICO DI TORINO
Repository ISTITUZIONALE

CROSS-ing the Divide: A Balanced RISC-V Accelerator for Code-Based Post-Quantum Signatures

*Original*
CROSS-ing the Divide: A Balanced RISC-V Accelerator for Code-Based Post-Quantum Signatures / Piscopo, V., Dolmeta, A., Martina, M., Masera, G.. - In: IEEE ACCESS. - ISSN 2169-3536. - 14:(2026), pp. 170196-170207. [10.1109/access.2026.3749088]

*Availability:*
This version is available at: 11583/3018008 since: 2026-11-17T09:37:23Z

*Publisher:*
IEEE

*Published*
DOI:10.1109/access.2026.3749088

*Terms of use:*

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

*Publisher copyright*

(Article begins on next page)

28 November 2026

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 September 2026, accepted 28 October 2026, date of publication 2 November 2026, date of current version 6 November 2026.

*Digital Object Identifier 10.1109/ACCESS.2026.3749088*

RESEARCH ARTICLE

# CROSS-ing the Divide: A Balanced RISC-V Accelerator for Code-Based Post-Quantum Signatures

**VALERIA PISCOPO, (Member, IEEE), ALESSANDRA DOLMETA, (Member, IEEE), MAURIZIO MARTINA, (Senior Member, IEEE), AND GUIDO MASERA, (Senior Member, IEEE)**
Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Torino, Italy
Corresponding author: Valeria Piscopo (valeria.piscopo@polito.it)

**ABSTRACT** Post-Quantum Cryptography (PQC) is rapidly evolving to ensure security against quantum adversaries. While several Key Encapsulation Mechanisms and structured-lattice signatures have been standardized, a range of alternative digital signatures has remained under active evaluation for their diverse security foundations. Deploying such schemes on constrained embedded systems, however, presents a severe architectural divide: pure software execution suffers from crippling latency, while monolithic hardware accelerators consume prohibitive amounts of logic area. To bridge this gap, we focus on CROSS (Codes and Restricted Objects Signature Scheme), which was among the alternative code-based candidates evaluated for its robust zero-knowledge foundations, and present CIRCE (CROSS Integrated RISC-V Cryptographic Extension), a tightly coupled coprocessor that strikes an optimal balance between performance and resource footprint. CIRCE offloads dominant cryptographic kernels without the rigidity of fixed-function hardware. Operating a single configurable instance for both the original R-SDP and generator-based R-SDP(G) variants, the design achieves an average 3.2× speed-up across all 18 official parameter sets. Implemented on a Zynq UltraScale+ FPGA and in a 65 nm ASIC flow, CIRCE delivers substantial energy savings and latency reduction while maintaining an ultra-compact area footprint. Beyond the CROSS-specific implementation, CIRCE provides a modular kernel-oriented architecture in which the CV-X-IF integration shell and Keccak datapath are cleanly separated from scheme-specific arithmetic. This organization preserves a compact tightly coupled interface while providing a concrete architectural basis for extending the same integration strategy to other PQC accelerators.

**INDEX TERMS** RISC-V, post-quantum cryptography, CROSS, FPGA, ASIC.

## I. INTRODUCTION

The advent of quantum computing threatens the security of widely deployed public-key cryptosystems, particularly RSA and elliptic-curve-based schemes, which can be broken by Shors algorithm [1]. This risk has driven the NIST (National Institute of Standards and Technology) Post-Quantum Cryptography (PQC) standardization process [2], which has already selected ML-KEM and HQC for key encapsulation and ML-DSA, SLH-DSA, and FALCON for Digital Signatures (DS). Despite this progress, digital signatures remain a challenging problem in the post-quantum setting. To diversify the design space and reduce reliance on a single class of assumptions, NIST launched an additional call for digital signatures in 2023, the so-called *signature on-ramp*, now in its third round [3]. Among the candidates, CROSS (Codes and Restricted Objects Signature Scheme) [4] is a code-based construction derived from Zero-Knowledge (ZK) protocols and the Restricted Syndrome Decoding Problem (R-SDP), instantiated via Fiat–Shamir [5] and relying on SHAKE/*Keccak* [6] for hashing and Cryptographically Secure Pseudo-Random Number

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

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

Generator (CSPRNG). While CROSS was not selected for the subsequent stage of the NIST Additional Digital Signatures Standardization Process, it remains one of the few code-based signature schemes that successfully reached the later phases of the evaluation [3], [7]. More broadly, the relevance of code-based post-quantum cryptography is not limited to the outcome of a single candidate: European guidance and international standardization activities continue to recognize code-based constructions, based on error-correcting codes and syndrome-decoding problems, as an important family of quantum-resistant primitives [8], [9]. In the same direction, the German Federal Office for Information Security (BSI) emphasizes the need for a careful transition toward quantum-safe cryptography and supports the consideration of alternative families beyond lattice-based schemes. This reinforces the interest of studying non-lattice directions, including code-based and zero-knowledge-based constructions as candidates for future crypto-agile deployments [10]. In this context, CROSS remains a valuable non-lattice, code-based case study for assessing the implementation cost of emerging post-quantum signature schemes and for designing agile embedded architectures that can adapt as the PQC standardization landscape evolves. It offers advantages that justify its hardware acceleration: robust Zero-Knowledge foundations, highly compact public keys, and rapid signature generation compared to other non-lattice candidates. These merits make it highly competitive for secure authentication in resource-constrained embedded systems. However, its heavy reliance on finite-field arithmetic and *Keccak* permutations poses significant implementation challenges on 32-bit cores.

In this paper, we design CIRCE (CROSS Integrated RISC-V Cryptographic Extension), a tightly coupled coprocessor integrated into a RISC-V SoC through the Core-V eXtension Interface (CV-X-IF).[1] Rather than adopting a purely loosely coupled approach, which can suffer from high communication overhead, or a strictly integrated custom pipeline, CIRCE acts as a tightly coupled coprocessor that occupies a middle ground by leveraging a dedicated local memory and a specialized interface to the RISC-V core. CIRCE implements two scheme variants: the original R-SDP formulation and the generator-based R-SDP(G) [4]. Both variants share the same accelerator architecture, which integrates a dedicated *Keccak*-unit for hashing, and differ only in selected finite-field prime ($\mathbb{F}_p$) arithmetic units (hereafter referred to as *fp*-Unit) that implement the finite-field operations required by CROSS. Compared with existing CROSS accelerators, CIRCE explicitly targets the design space between the two prevailing extremes. Fully monolithic implementations achieve very low latency by offloading most of the scheme at a substantial area cost, whereas compact RISC-V instruction-set extensions minimize hardware overhead but accelerate a narrower subset of the computation.

*Contributions:* The main contributions of this work are:

1) We propose CIRCE,[2] a tightly coupled CV-X-IF coprocessor that establishes a distinct intermediate design point for CROSS between monolithic hardware accelerators and minimal RISC-V instruction-set extensions. By jointly offloading the dominant Keccak and finite-field arithmetic kernels, CIRCE provides end-to-end acceleration of KeyGen, Sign, and Verify while retaining a substantially smaller hardware footprint than fully standalone implementations and a broader acceleration scope than minimal ISE-based approaches.
2) CIRCE natively supports both R-SDP and R-SDP(G) formulations and operates across all 18 official CROSS parameter sets (*v2.2*) without hardware retuning, enabling software-controlled crypto-agility within a single lightweight instance.
3) We integrate CIRCE into a RISC-V-based SoC and evaluate it on two implementation targets: (i) **FPGA**, where we report end-to-end results validated with Known Answer Tests (KATs) [11] and compare resource footprint and runtime against state-of-the-art FPGA implementations of NIST signature on-ramp candidates; and (ii) a **65 nm ASIC flow**, where we report post-synthesis power estimates for the main accelerated kernels used in Key Generation (KeyGen), Signing (Sign), and Verification (Verify).

The architecture explicitly separates reusable integration and hashing components from CROSS-specific arithmetic. The CV-X-IF coupling, decode–execute–commit organization, and Keccak datapath therefore define a modular integration framework that can be retained when targeting additional PQC schemes, while the arithmetic datapath and instruction semantics can be adapted to the requirements of each scheme. The paper is organized as follows: section II introduces PQC acceleration and CROSS, while section III details the CIRCE architecture and section IV the integration model. section V discusses security considerations, results and comparisons are reported in section VI, and conclusions in section VII.

## II. BACKGROUND

### A. PQC ACCELERATION

As PQC transitions from algorithm selection to real-world deployment, efficient execution on embedded and IoT-class processors has become a critical challenge. Recent evaluations across heterogeneous computing platforms confirm that practical performance strongly depends on the target architecture and available resources [12]. Consequently, hardware/software co-design has emerged as a key strategy to bridge the gap between pure software execution and rigid fixed-function hardware. Existing implementations span a broad design space: monolithic accelerators minimize latency by offloading large portions of a cryptographic protocol, whereas tightly coupled Instruction Set Extensions (ISEs) reduce hardware overhead by accelerating selected

[1] https://github.com/openhwgroup/core-v-xif

[2] https://github.com/edge-group-polito/CIRCE.git

computational kernels. For lattice-based signatures, RISC-V hardware/software designs accelerate Dilithium/ML-DSA and Falcon through dedicated polynomial-arithmetic and hashing units [13], while ATHOS combines tightly and loosely coupled accelerators for CRYSTALS workloads through CV-X-IF [14]. Hash-based signatures have similarly been addressed through RISC-V-attached SPHINCS+ architectures [15], customized processors for SLH-DSA [16], and area-efficient full-hardware implementations such as SPHINCSLET [17]. Overall, these works expose a recurring trade-off among absolute performance, hardware footprint, and the amount of algorithmic functionality retained in software.

Hardware support for non-lattice and additional-signature candidates has also expanded rapidly. FPGA architectures have been proposed for CROSS [18], LESS [19], [20], SDitH [21], [22], MEDS [23], Raccoon [24], MAYO [25], and Mirath [26]. Complementary implementation-oriented studies target memory-constrained CROSS [27], [28], low-RAM MAYO [29], and evolving implementation trade-offs in VOLE-in-the-head and SDitH-v2 constructions [30], [31]. These results further highlight that implementation efficiency depends not only on the underlying cryptographic family, but also on the selected acceleration granularity, memory organization, and degree of hardware specialization.

Within RISC-V, tightly coupled cryptographic coprocessors and CV-X-IF-based extensions provide a modular approach to custom-instruction acceleration. RISQ-V introduced tightly coupled RISC-V acceleration for PQC [32], while ATHOS [14], TYRCA [33], and the Keccak coprocessor of [34] applied CV-X-IF to lattice-based, code-based, and hashing workloads, respectively. Independent work has also investigated automated generation of CV-X-IF-compatible coprocessors and compiler support [35], [36]. These approaches demonstrate the flexibility of tightly coupled RISC-V extensions across different cryptographic kernels and integration granularities.

In this design space, CIRCE targets a distinct architectural point for CROSS. The high-performance standalone accelerator of [18] minimizes latency by moving most of the scheme into hardware, whereas the tightly coupled ISE of [37] achieves an extremely small footprint by accelerating a narrower subset of the computation. CIRCE instead jointly accelerates the dominant Keccak and finite-field arithmetic kernels while retaining the remaining protocol control in software. This organization provides end-to-end acceleration of KeyGen, Sign, and Verify, supports both R-SDP and R-SDP(G) across all 18 parameter sets within the same configurable architecture, and positions CIRCE between the latency-oriented standalone and minimum-area ISE design points.

### B. CROSS

CROSS (Codes and Restricted Objects Signature Scheme) is a post-quantum digital signature scheme derived from zero-knowledge (ZK) protocols, utilizing the Fiat-Shamir transform to ensure existential unforgeability under chosen message attacks (EUF-CMA). Its security is founded on the hardness of the Restricted Syndrome Decoding Problem (R-SDP) and its generalized variant, R-SDP(G).

- **R-SDP**: Given a syndrome $s \in \mathbb{F}_p^{n-k}$ and a parity-check matrix $H \in \mathbb{F}_p^{(n-k)\times n}$, the goal is to find an error vector $e \in E^n$, where $E = \langle g \rangle \subseteq \mathbb{F}_p^\star$ is a cyclic subgroup of order $z$, such that $s = eH^\top$.
- **R-SDP(G)**: A generalization where solutions must reside in a subgroup $G \subseteq E^n$ of size $z^m$, represented compactly through a generator matrix $M \in \mathbb{F}_z^{m\times n}$.

While R-SDP uses small parameters (e.g., $p$=127, $z$=7) for highly efficient arithmetic, R-SDP(G) adopts larger primes (e.g., $p$=509, $z$=127) to compress the error vector. This structural efficiency yields highly compact public keys (tens of bytes) and signatures ranging from $\approx$ 9 kB to $\approx$ 74 kB. Following NIST requirements, it targets security categories 1, 3, and 5, with each offering three operating points: **Fast** ($f$) to favor throughput, **Balanced** ($b$) for an optimal speed-size trade-off, and **Small** ($s$) for maximal compression at the expense of latency. As documented in NIST IR 8610 [7], CROSS underwent significant security analysis during the second round of the Additional Digital Signature Standardization Process. A formal analysis of the multi-round Fiat–Shamir construction provided an updated security proof and also identified an attack that led the CROSS team to revise its parameter sets in the second-round submission. A separate issue was subsequently identified in the security analysis of the underlying CROSS-ID protocol; although this analysis did not claim a break of CROSS itself, the submission team updated its security documentation accordingly. The present work therefore targets the latest CROSS v2.2 parameter sets and implementation [38]. NIST ultimately did not advance CROSS to the third round, citing its overall performance trade-offs and the characteristics of its underlying hardness assumptions relative to alternative candidates.

*Cryptographic Operations and Bottlenecks:* CROSS operates predominantly over a finite field. Throughout this work, we use the term *ring* $\mathcal{R}$ to denote the underlying algebraic structure, specifically, the finite field $\mathbb{F}_p$ or its extensions, over which the algorithm's polynomial and matrix operations are performed. To understand the specific tasks executed by our accelerator, it is essential to trace the core cryptographic operations of CROSS:

- **KeyGen**: Focuses on generating the public syndrome $s = eH^\top$. This primarily requires expanding vectors and executing efficient vector-matrix multiplications over $\mathcal{R}$.
- **Sign**: Employs a Fiat-Shamir Zero-Knowledge proof system. It generates restricted error vectors, computes multiple commitment syndromes, and processes verifier challenges. This phase relies heavily on restricted-to-field value expansions and multiply-accumulate operations to evaluate the necessary algebraic checks.

- **Verify**: Reconstructs the commitments from the provided signature. This requires re-evaluating the commitment algebra by performing challenge-dependent syndrome updates and point-wise vector multiplications.

These distinct algorithmic steps dictate the mathematical hardware requirements of the design. The *fp*-Unit is explicitly tailored to execute these exact modular kernels (field multiply-accumulate, restricted-to-field expansions, point-wise multiplication with reduction, and syndrome updates), avoiding the latency of general-purpose software execution.

To quantify the performance impact of the recurring computational kernels, we profiled the CROSS v2.2 implementation for the balanced R-SDP-1-b and R-SDP(G)-1-b parameter sets. The software was compiled with `-O2` optimization and initially evaluated through function-level profiling with Synopsys ASIP Designer [39]. Execution cycles were grouped into three classes according to the profiled software regions: (i) hashing and pseudo-random generation, corresponding to SHAKE eXtendable-Output Functions (XOFs) based on the Keccak-$f$[1600] permutation [40]; (ii) finite-field arithmetic over the aforementioned ring $\mathcal{R}$, including multiply–accumulate operations, restricted-value expansions, and modular reductions over $\mathbb{F}_p$; and (iii) the remaining control, packing, and data-management operations.

The resulting breakdown, shown in Fig. 1, highlights hashing and randomness generation as the main contributor, accounting for up to 73.0% of the execution time, while finite-field arithmetic reaches 26.9% during signing. Together, these two classes account for the dominant computational share across **KeyGen**, **Sign**, and **Verify**.

The profiling results were independently validated on the FPGA-based X-HEEP platform using the 32-bit CV32E40PX RISC-V core, featuring a four-stage in-order pipeline. Hardware performance counters were sampled around the corresponding software regions using the same CROSS workloads. The measured distributions confirmed the ASIP Designer results, with deviations of at most 5 percentage points across the three profiling classes.

These results identify Keccak-based hashing and ring arithmetic as the main computational bottlenecks across both the common and G-variant implementations. Despite targeting different parameter sets, these operations rely on common arithmetic structures and can therefore be combined within a single configurable hardware block. This makes them well suited for acceleration through custom instructions, while preserving ISA compliance and enabling efficient hardware reuse.

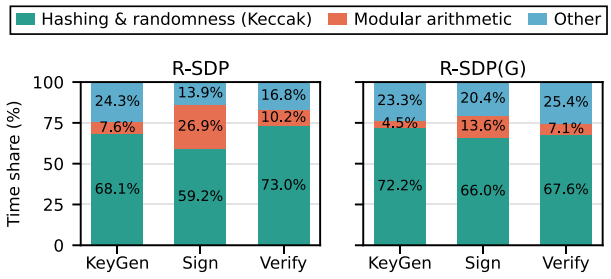



**FIGURE 1.** CROSS breakdown for R-SDP-1-b and R-SDP(G)-1-b.

## III. DESIGN

The CIRCE architecture is designed as a custom coprocessor tightly coupled to the main CPU core, providing low-latency access to specialized cryptographic arithmetic without the bus contention and driver overhead typical of memory-mapped accelerators. Rather than operating as a detached peripheral, CIRCE behaves as a native extension of the processor's execution stage: it observes the instruction stream, claims the custom opcodes that belong to it, and returns results directly to the architectural state. The internal structure of CIRCE, depicted in Figure 2, is organized into three main functional blocks that mirror the natural decode–execute–writeback progression of the host pipeline: the `decoder`, the collection of accelerators, and the `committer`. The `decoder` serves as the entry point of the coprocessor. It identifies incoming custom instructions, parses the instruction word to extract the relevant source operands, activates the corresponding internal accelerator, and manages operand forwarding, thereby acting as the internal control bridge for the rest of the design. Because the instruction-to-unit mapping is confined to a single configuration point, the supported instruction set can be extended or adjusted without disturbing the surrounding logic, which keeps the accelerator collection cleanly decoupled from the dispatch mechanism. For the core execution phase, CIRCE instantiates two primary accelerators, each targeting one of the dominant computational kernels identified in section II: the *keccak*-`Unit`, which implements the permutation used in the SHAKE XOFs for hashing and randomness generation, and the *fp*-`Unit`, responsible for the field-prime arithmetic over the ring $\mathcal{R}$. Grouping both kernels under a shared control structure allows the two most expensive workloads of **CROSS** to be served by a single coprocessor instance, maximizing hardware reuse while keeping the area footprint compact. Finally, the `committer` stage collects the computed results from the active accelerator and validates the outputs before they are processed for write-back. By gating the write-back behind this validation step, the committer ensures that results re-enter the host pipeline with the same determinism and consistency as a native arithmetic operation, preserving precise execution semantics from the perspective of the CPU. For clarity, in the notation used throughout this work: $\wedge$ denotes bitwise AND, and $\gg$ represents logical shift-right operations.

### A. KECCAK-*UNIT*

We designed a custom *keccak*-`Unit`, fully based on combinational logic, that computes a full Keccak-$f$[1600] round [40] in a single clock cycle, enabling the 24-round permutation to complete in exactly 24 cycles (`keccak-f1600` of Figure 2). Because the entire round function, comprising the $\theta, \rho, \pi, \chi$ and $\iota$ steps, is unrolled into a single

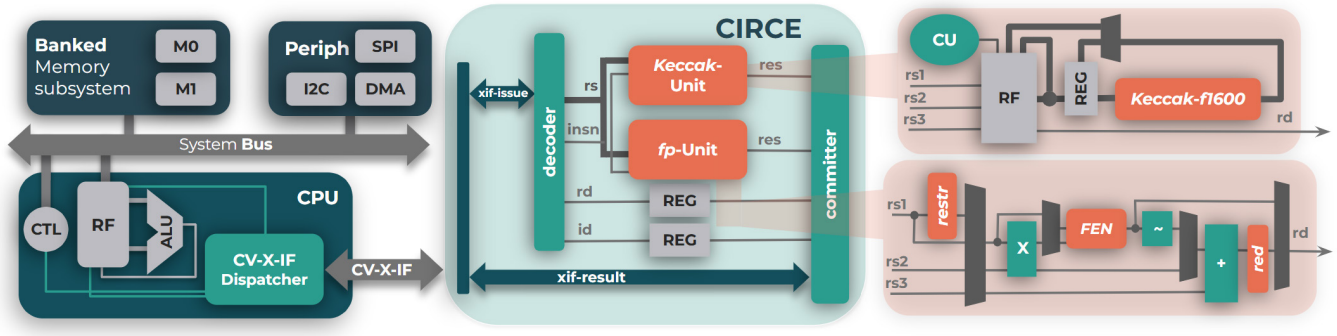



**FIGURE 2.** CIRCE architectural schematic and RISC-V pipeline integration via CV-X-IF.

combinational datapath, no intermediate state needs to be spilled to memory between rounds, which is the primary source of overhead in software and lightly-accelerated implementations. While some literature suggests loosely-coupled accelerators for large-state functions, our previous comparative research into RISC-V integration methodologies [41], [42] demonstrates that a tightly-coupled approach via CV-X-IF is superior for reducing the communication latency inherent in frequent state updates: a loosely-coupled unit would otherwise pay a recurring transfer cost on every absorb and squeeze, eroding the benefit of the accelerated permutation. To resolve the specific challenge of managing a 1600-bit state within the CPU pipeline, whose 32-bit register file cannot natively hold the full *Keccak* state, our design introduces a dual-buffer topology: a dedicated internal 1600-bit register (`reg`) handles high-speed iterative computation, keeping the working state local to the accelerator across all 24 rounds, while the Register File (`RF`) retains the architectural context and serves as the interface to the surrounding software. This separation avoids streaming the state in and out of the core on every round, confining the wide datapath to the coprocessor while preserving full ISA compliance. Data movement and RF indexing are managed internally by a compact Control Unit (`CU`) according to the issued instruction, so that the programmer interacts with the unit through a small, regular set of operations rather than manual state bookkeeping. The unit is controlled by a granular ISA extension: `init` clears the state, `load` and `store` manage data transfers between the RF and the internal register, and `kabsorb` provides a single-cycle XOR-injection for the absorb phase, folding input blocks into the state without a separate read–modify–write sequence. The permutation is then triggered by `kstart` for non-destructive processing, which preserves the source operands, or by `kperm` for in-place updates with an automated write-back to the RF. This instruction granularity lets the same hardware serve both the absorb and squeeze phases of the SHAKE XOFs used throughout **CROSS**, supporting the variable-length output demands of hashing and randomness generation without any reconfiguration of the datapath.

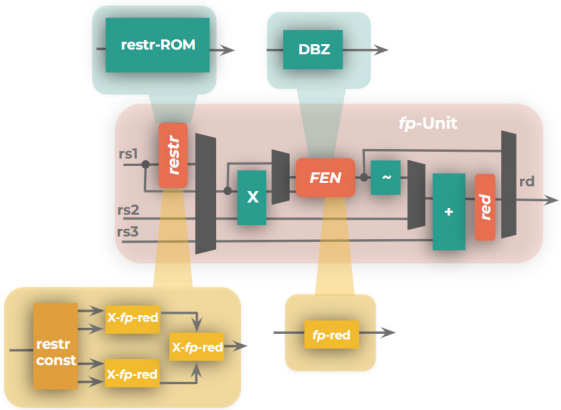



**FIGURE 3.** *fp*-`Unit` architectural schematic. The datapath highlights the `restr`, `FEN`, and `red` modules (in coral) which differ between R-SDP and R-SDP(G) variants. Supporting blocks for the R-SDP variant are shown in teal, while those specific to the R-SDP(G) variant are shown in yellow.

### B. FP-UNIT

The *fp*-`Unit` is a compact, branch-free arithmetic block designed to accelerate the finite-field operations required by the **CROSS** cryptosystem. While the high-level datapath is shared between the R-SDP and R-SDP(G) variants, the specific implementation of the restricted-domain mapping (`restr`), field element normalization (`FEN`), and reduction logic (`red`) is tailored to the underlying field representation.

The unit is driven by three source registers and a custom instruction selector. The data flow through the datapath is structured as follows:

- **Restricted-domain mapping** (`restr`): The `rs1` register can optionally be routed through this block. It expands restricted-domain elements into their corresponding valid field elements, a critical step for instructions handling encoded vectors. In the R-SDP variant, this is implemented by `restr-rom`, a 3-to-8-bit ROM lookup table that maps the 3-bit restricted index directly to its value in $\mathbb{F}_{127}$. In the R-SDP(G) variant, it is implemented by the `restr-const` module, which uses a constant-time bit-decomposition strategy:

each bit $b_k$ of the 7-bit input selects a generator power via `FP_ELEM_CMOV`$(b_k, T, F) := (mask \,\&\, T) \mid (\sim mask \,\&\, F)$, with $mask = \{16\{b_k\}\}$. The resulting four partial products are then combined through a three-level multiplication tree, each level followed by a modular reduction (`fpred_single32`).

- **Operand multiplexer**: This stage selects operands from either the raw source registers (`rs1`, `rs2`, `rs3`) or the output of the `restr` block, ensuring that the multiplier always receives data in the appropriate field representation.
- **Multiplier**: This block performs an unsigned $16 \times 16$-bit multiplication, producing a 32-bit intermediate product.
- **Field Element Normalization** (`FEN`): This block ensures that elements are maintained in canonical form. In the R-SDP variant, it is implemented by the `DBZ` (Double-Zero Normalization) module, which resolves the $127 \equiv 0$ ambiguity via a shift-and-mask reduction: $r = (x \,\&\, 0 \times 7\mathrm{F}) + (x \gg 7)$, followed by a conditional subtraction if $r \geq 127$. In the R-SDP(G) variant, canonicalization is instead handled directly by the `fp-red` block, since the Barrett-style reduction already produces a unique representative in $[0, P-1]$; the `FEN` step is therefore an identity operation in this variant.
- **Adder**: This stage accumulates the multiplier output (or normalized product) with the `rs3` register.
- **Reduction** (`red`): The final stage applies modular reduction based on the specific field modulus, mapping the adder output back into the valid range $[0, P-1]$.

## IV. INTEGRATION

CIRCE is integrated into the X-HEEP platform [43] as a co-processor attached to a RI5CY core. This integration leverages the Core-V eXtension Interface (CV-X-IF) provided by the cv32e40px implementation.[3] CV-X-IF is a standardized interface that enables instruction-set extensions to be attached to a host core in a non-invasive manner, and is supported across a growing range of OpenHW cores and platforms. By relying on this standardized bridge rather than on a custom point-to-point connection, CIRCE inherits a portable integration path that does not tie the design to a single core implementation. By utilizing the standardized CV-X-IF sub-interfaces, CIRCE can operate seamlessly as an external functional unit without requiring modifications to the base CPU RTL. The interaction with the host follows the issue–execute–commit flow exposed by the interface. Custom instructions are detected in the decode stage of the CPU pipeline and dispatched to the accelerator through the `xif-issue` package, which provides both the operand values and the full instruction word. These signals drive the CIRCE `decoder` to select the appropriate operation and route the operands to the target accelerator, so that no separate memory transaction is needed to feed the coprocessor. Once the internal computation is completed and validated by the `committer`, the 32-bit result is returned to the processor via the `xif-result` channel, where it is written back into the core register file under the host's own write-back logic. This standardized interaction provides low-latency access to custom arithmetic while preserving ISA compliance and software portability: because the extension is reached through ordinary instructions rather than driver calls or memory-mapped registers, it requires no changes to the host RTL or toolchain and remains portable across the Core-V family. This non-invasive, agility-aware coupling is also well aligned with the modular update model recommended for evolving cryptographic standards, allowing the accelerated functionality to be revised independently of the host CPU as the PQC landscape matures.

[3] https://github.com/x-heep/cv32e40px

**TABLE 1. The CIRCE instruction set. The `{,G}` suffix denotes the R-SDP and R-SDP(G) variants, which share an encoding and differ only in the selected *fp*-`Unit` datapath. All instructions use standard RISC-V R-type or R4-type encoding.**

| Instruction | Unit | Description |
|---|---|---|
| | *Keccak & State Management* | |
| `INIT` | keccak | Clears the 1600-bit internal state. |
| `LOAD` | keccak | Transfers a word from the RF into the internal register. |
| `STORE` | keccak | Reads a word from the internal register back to the RF. |
| `KABSORB` | keccak | XOR-injects input data into the state (absorb). |
| `KSTART` | keccak | Triggers Keccak-$f$[1600], preserving the source state. |
| `KPERM` | keccak | In-place permutation with automatic write-back. |
| | *Finite-Field Arithmetic over* $\mathcal{R}$ | |
| `R2V{,G}` | *fp* | Restricted-domain mapping to valid field elements. |
| `VMUL{,G}` | *fp* | Vector–matrix multiplication. |
| `RVMUL{,G}` | *fp* | Restricted vector–matrix multiplication. |
| `VPW{,G}` | *fp* | Element-wise (pointwise) vector multiplication. |
| `SYUPD{,G}` | *fp* | Syndrome update for challenge processing. |

### A. CUSTOM INSTRUCTION SET

This modular structure enables a single *fp*-`Unit` datapath to execute the core arithmetic primitives required by CROSS.

As summarized in Table 1, the unit supports five custom instructions, each provided in both R-SDP and R-SDP(G) variants that share an encoding and differ only in the selected reduction and mapping logic. The instructions are defined as follows:

- `vmul`: Vector–matrix multiplication for standard field vectors.
- `rvmul`: Restricted vector–matrix multiplication, in which the input vector is first expanded from its restricted-domain encoding.
- `vpw`: Element-wise (pointwise) multiplication of two field vectors.
- `syupd`: Syndrome update used during challenge processing.
- `r2v`: Restricted-domain mapping that expands encoded elements into valid field elements.

Together with the *Keccak* instructions listed in Table 1, these primitives cover the dominant finite-field and hashing kernels of CROSS, allowing the complete signature flow to be accelerated through a small, regular instruction set without any datapath reconfiguration between parameter sets.

### B. SCOPE AND PORTABILITY

CIRCE is not intended as a drop-in accelerator for arbitrary code-based signature schemes. Its Keccak unit, CV-X-IF integration, and decode–execute–commit organization are structurally reusable, whereas the fp-Unit and the associated custom instructions implement arithmetic and dataflows specific to CROSS. Extending the architecture to schemes such as SDitH [44] or MEDS [45] would therefore require scheme-specific arithmetic kernels and instruction semantics, followed by independent functional and performance validation. Accordingly, the generality of the proposed approach lies in the kernel-oriented co-design and integration methodology rather than in demonstrated cross-scheme hardware portability.

## V. SECURITY CONSIDERATIONS

The current CIRCE implementation targets performance, area, and energy efficiency and does not include dedicated side-channel countermeasures. Therefore, the presented results should not be interpreted as demonstrating resistance against physical side-channel attacks.

*Timing and power leakage.* At the accelerator level, the execution schedule is largely data independent: the Keccak-$f$[1600] unit always executes 24 rounds in 24 cycles, while the fp-Unit implements branch-free arithmetic and constant-time restricted-domain conversion. These properties reduce obvious timing leakage within the accelerated units, although they do not guarantee constant-time execution of the complete CROSS software stack. Power leakage remains a relevant concern, since the wide Keccak combinational datapath and the fp-Unit multiplier and reduction logic exhibit data-dependent switching activity while processing cryptographic intermediates. CIRCE should therefore presently be considered unprotected against power-analysis attacks.

*Countermeasures.* Possible hardening strategies include Boolean masking of the Keccak state, with particular protection of the nonlinear $\chi$ step, and arithmetic masking of sensitive operands in the fp-Unit multiplier and modular-reduction datapath. Complementary hiding techniques may further reduce exploitable leakage. The implementation of these countermeasures and the experimental leakage assessment are left for future work.

## VI. RESULTS

CIRCE is implemented on both FPGA and ASIC to capture complementary aspects of the design. In both cases, CIRCE is integrated into X-HEEP as a tightly coupled CV-X-IF coprocessor attached to the 32-bit CV32E40PX core, so that the same RTL architecture and software interface are evaluated across the two implementation targets.

**TABLE 2. Hierarchical hardware cost breakdown.**

| Block | ASIC [$\mu$m$^2$] | | FPGA [LUT/FF/DSP] | |
|---|---|---|---|---|
| | R-SDP | R-SDP(G) | R-SDP | R-SDP(G) |
| `top` | **76,876** | **92,403** | **9,459/3,376/1** | **9,733/3,375/38** |
| `decoder` | ≈ 1,166 | | ≈ 3509/112/0 | |
| `circe` | **75,600** | **91,129** | **5914/3254/1** | **6214/3254/38** |
| `RF` | 25,908 | | 832/1600/0 | |
| `keccak` | ≈ 44,051 | | 5077/1653/0 | |
| `fp-Unit` | **2,752** | **18,276** | **5/0/1** | **305/0/38** |
| `committer` | ≈19.4 | | 5/0/0 | |

For FPGA validation, it targets a Zynq UltraScale+ ZCU104 (XCZU7EV-2FFVC1156) and is synthesized using Vivado 2022.2 at 200 MHz. FPGA resource utilization is reported in terms of LUTs, FFs, BRAMs, and DSPs, both for the complete design and for the main CIRCE submodules. To provide FPGA-independent insight into the cost, we also implement CIRCE in an ASIC flow. The ASIC architecture is synthesized using Synopsys Design Compiler 2020.09 in a 65 nm CMOS technology at a target clock frequency of 160 MHz (with 5% clock uncertainty). Optimization is enabled via `compile_ultra`. It should be noted that these metrics reflect post-synthesis results before the Place and Route (P&R) phase.

### A. AREA RESULTS

Table 2 reports the hardware cost of CIRCE in terms of ASIC area and FPGA resources for both variants. The common sub-modules exhibit nearly identical area results across both designs; consequently, their costs are reported as approximate values in Table 2 to account for minor synthesis and optimization noise.

The hardware overhead of R-SDP(G) is primarily driven by the `fp-Unit`. Its ASIC area increases nearly sevenfold, from 2,752 to 18,276 $\mu$m$^2$, while other modules remain stable. On FPGA, this trend is mirrored by a significant leap in arithmetic density, with DSPs increasing to support the additional G-module datapaths.

### B. PERFORMANCE RESULTS

Table 3 reports the performance results in kilo-clock cycles (**kCC**) for the three core operations KeyGen, Sign, and Verify, across different parameter sets and security levels. All benchmarks were compiled with GCC `-O2`. The speed-up factors are obtained as the ratio between the cycles required by the software baseline and the optimized design. The average speed-ups for KeyGen, Sign, and Verify are, respectively, 3.6×, 2.9×, and 3.0×, demonstrating the effectiveness of the proposed solution.

### C. POWER MEASUREMENTS

Table 4 reports clock-cycle (C.C.) counts by profiling specifically triggered operations, while post-synthesis power is extracted from ASIC gate-level simulations using switching activity on the synthesized netlist. Power is estimated with Synopsys PrimePower 2024.09, and energy is computed

**TABLE 3.** Performance results executing KeyGen, Sign, and Verify. [CC] are Clock Cycles, [×] represent speed-up factor. The table reports original execution cycles, optimized cycles, and the total cycles saved.

| CROSS | KeyGen | | | | Sign | | | | Verify | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Base [CC] | Opt. [CC] | Saved [CC] | [×] | Base [CC] | Opt. [CC] | Saved [CC] | [×] | Base [CC] | Opt. [CC] | Saved [CC] | [×] |
| 1-f | 793,105 | 246,081 | *547,024* | **3.2** | 36,383,211 | 13,968,432 | *22,414,779* | **2.6** | 20,817,645 | 7,429,112 | *13,388,533* | **2.8** |
| 1-b | 793,088 | 241,503 | *551,585* | **3.3** | 70,436,891 | 24,727,334 | *45,709,557* | **2.9** | 37,672,119 | 9,929,085 | *27,743,034* | **3.8** |
| 1-s | 793,210 | 246,902 | *546,308* | **3.2** | 144,084,551 | 48,336,220 | *95,748,331* | **3.0** | 74,511,304 | 17,955,871 | *56,555,433* | **4.2** |
| 3-f | 1,874,312 | 529,198 | *1,345,114* | **3.5** | 97,388,401 | 39,894,256 | *57,494,145* | **2.4** | 53,941,772 | 20,842,615 | *33,099,157* | **2.6** |
| 3-b | 1,872,055 | 529,441 | *1,342,614* | **3.5** | 224,340,113 | 63,460,982 | *160,879,131* | **3.5** | 82,224,209 | 23,160,334 | *59,063,875* | **3.6** |
| 3-s | 1,868,610 | 528,705 | *1,339,905* | **3.5** | 254,018,881 | 95,194,512 | *158,824,369* | **2.7** | 121,015,102 | 32,218,811 | *88,796,291* | **3.8** |
| 5-f | 3,307,245 | 939,114 | *2,368,131* | **3.5** | 188,949,663 | 85,897,290 | *103,052,373* | **2.2** | 105,508,415 | 43,986,752 | *61,521,663* | **2.4** |
| 5-b | 3,307,811 | 942,306 | *2,365,505* | **3.5** | 259,453,019 | 132,729,144 | *126,723,875* | **2.0** | 145,419,531 | 43,526,802 | *101,892,729* | **3.3** |
| 5-s | 3,322,404 | 929,517 | *2,392,887* | **3.6** | 429,923,712 | 198,129,335 | *231,794,377* | **1.9** | 217,799,188 | 58,024,456 | *159,774,732* | **3.8** |
| (G)1-f | 408,111 | 106,892 | *301,219* | **3.9** | 23,522,504 | 7,681,213 | *15,841,291* | **3.1** | 14,393,667 | 5,209,105 | *9,184,562* | **2.8** |
| (G)1-b | 404,732 | 107,314 | *297,418* | **3.8** | 51,440,815 | 14,421,556 | *37,019,259* | **3.6** | 31,002,129 | 9,359,602 | *21,642,527* | **3.3** |
| (G)1-s | 407,221 | 106,099 | *301,122* | **3.8** | 103,324,403 | 28,470,118 | *74,854,285* | **3.6** | 62,107,774 | 18,298,331 | *43,809,443* | **3.4** |
| (G)3-f | 840,519 | 217,322 | *623,197* | **3.9** | 55,658,210 | 20,043,804 | *35,614,406* | **2.8** | 35,467,991 | 13,965,412 | *21,502,579* | **2.5** |
| (G)3-b | 840,108 | 212,544 | *627,564* | **4.0** | 76,056,337 | 25,154,619 | *50,901,718* | **3.0** | 47,571,440 | 16,874,225 | *30,697,215* | **2.8** |
| (G)3-s | 842,671 | 227,105 | *615,566* | **3.7** | 148,726,902 | 47,116,488 | *101,610,414* | **3.2** | 91,040,116 | 31,437,803 | *59,602,313* | **2.9** |
| (G)5-f | 1,403,812 | 388,441 | *1,015,371* | **3.6** | 103,506,554 | 41,099,230 | *62,407,324* | **2.5** | 66,306,199 | 29,803,771 | *36,502,428* | **2.2** |
| (G)5-b | 1,403,227 | 386,915 | *1,016,312* | **3.6** | 138,558,304 | 49,827,882 | *88,730,422* | **2.8** | 82,422,661 | 35,064,118 | *47,358,543* | **2.4** |
| (G)5-s | 1,403,509 | 384,204 | *1,019,305* | **3.7** | 369,574,811 | 90,049,633 | *279,525,178* | **4.1** | 146,412,995 | 60,597,210 | *85,815,785* | **2.4** |

as $\mathbf{E} = \mathbf{P_{tot}} \times \mathbf{C.C.} \times (1/\text{freq})$ with freq $= 160\,\text{MHz}$. We performed fine-grained power measurements to compare the baseline software execution of **CROSS** (white rows of Table 4) against the CIRCE-enhanced version (light-blue rows in Table 4). Rather than reporting only aggregate power for **KeyGen**, **Sign**, and **Verify**, which would mix cryptographic kernels with auxiliary steps (e.g., packing/unpacking and transcript handling), we isolate a small set of self-contained blocks with clear cryptographic meaning that dominate runtime and exercise the accelerated datapath. For **KeyGen**, we measure `synd`, i.e., the public syndrome computation, including normalization and packing. For **Sign**, we report `cm-cmp` (per-round commitment computation and digest finalization), `sgn-chal1` (first Fiat–Shamir challenge derivation, hashing-dominated), and `rsp-gen` (response generation and normalization). For **Verify**, we measure `r-closed`, which reconstructs and normalizes verifier-side values for unopened rounds, and `r-open`, which recomputes commitment-side values for opened rounds and verifies them via syndrome processing and digest recomputation. The results in Table 4 reflect the different proportions of accelerated finite-field kernels and *Keccak* operations across the measured blocks. The `synd` step is purely finite-field arithmetic, and therefore its energy reduction is driven exclusively by the shorter execution time of the $\mathbb{F}_p$ datapath. The `cm-cmp` block, instead, combines arithmetic-intensive vector/matrix operations and syndrome processing with the computation of the commitment digests. As a consequence, it benefits from both the accelerated $\mathbb{F}_p$ kernels and *Keccak* offloading, and the two contributions compound into a larger overall energy reduction than in arithmetic-only blocks. This separation is even clearer in `sgn-chal1`, which is essentially hashing-only: the speedup and the associated energy saving are dominated by the *Keccak* accelerator, yielding a marked reduction despite the absence of $\mathbb{F}_p$ computation. Conversely, `rsp-gen` is dominated by finite-field arithmetic and therefore follows the same trend as `synd`.

**TABLE 4.** Instruction-level characterization for R-SDP and R-SDP(G) architectures. Columns report cycles (C.C.), total power ($P_{tot}$) [µW], energy (E) [mJ], energy difference ($\delta E$), and energy saving.

| Test | Function | C.C. | | $P_{tot}$ | | E | | $\delta E$ | | Energy Saving | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | R-SDP | R-SDP(G) | R-SDP | R-SDP(G) | R-SDP | R-SDP(G) | R-SDP | R-SDP(G) | R-SDP | R-SDP(G) |
| **KeyGen** | *synd* | 73,276 | 22,275 | 217 | 244 | 99.4 | 34.0 | 34 | 10.0 | ×1.53 | ×1.42 |
| | | 41,209 | 8,323 | 253 | 460 | 65.2 | 23.9 | | | | |
| **Sign** | *cm-cmp* | 186,700 | 123,241 | 217 | 245 | 253 | 189 | 126 | 107 | ×1.99 | ×2.32 |
| | | 79,084 | 42,953 | 257 | 303 | 127 | 81.3 | | | | |
| | *sgn-chal1* | 51,726 | 52,150 | 217 | 245 | 70.2 | 79.9 | 57.4 | 64.8 | ×5.50 | ×5.31 |
| | | 8,147 | 8,188 | 251 | 244 | 12.8 | 15.0 | | | | |
| | *rsp-gen* | 4,909 | 7,712 | 216 | 244 | 6.63 | 11.8 | 2.49 | 9.98 | ×1.60 | ×6.59 |
| | | 2,676 | 673 | 247 | 424 | 4.13 | 1.78 | | | | |
| **Verify** | *r-closed* | 88,617 | 78,176 | 217 | 245 | 120 | 120 | 88 | 74.7 | ×3.76 | ×2.66 |
| | | 21,047 | 27,021 | 243 | 266 | 32 | 44.5 | | | | |
| | *r-open* | 109,194 | 68,213 | 217 | 245 | 148 | 104 | 53.9 | 58.4 | ×1.57 | ×2.28 |
| | | 62,023 | 29,887 | 243 | 244 | 94.2 | 45.6 | | | | |

On the verifier side, round processing splits according to the values of the second challenge. The closed-round branch (`r-closed`) mainly performs a hash and a short reconstruction/normalization of the response vector; the opened-round branch (`r-open`) includes additional unpacking and a longer reconstruction chain (pointwise products, matrix–vector multiplication, syndrome update/normalization) followed by hashing. Consequently, `r-open` remains the most expensive path in absolute terms, while `r-closed` shows a higher relative energy saving because a larger fraction of its workload falls in the parts most effectively accelerated.

### D. STATE-OF-THE-ART (SOTA) COMPARISON

Table 5 compares CIRCE with representative FPGA implementations of post-quantum signature schemes spanning different algorithmic families and architectural approaches. ASIC results are excluded to retain a more homogeneous implementation baseline. Since the considered works target different FPGA families, operating frequencies, acceleration scopes, and memory organizations, the comparison is intended to characterize the design space rather than provide a technology-normalized absolute ranking. Runtime is recomputed from the reported cycle counts and operating frequencies whenever both quantities are available.

To provide a common area indicator across Xilinx FPGA families, resources are expressed through an equivalent-slice metric. For Xilinx UltraScale devices, featuring 8 LUTs and 16 FFs per slice, we use $\text{eSlices} = \max\left\{\frac{\text{LUT}}{8}, \frac{\text{FF}}{16}\right\}$, with BRAM excluded for the UltraScale designs considered here, since CIRCE and [37] do not use BRAMs. For Xilinx 7-Series devices, featuring 4 LUTs and 8 FFs per slice, and converting each BRAM36 into 128 equivalent slices, we use $\text{eSlices} = \max\left\{\frac{\text{LUT}}{4} + 128 \cdot \text{BRAM}, \frac{\text{FF}}{8}\right\}$ following [47]. KeSlice denotes $10^3$ equivalent slices. Runtime is reported in milliseconds as kCC$/f$, where kCC denotes thousands of clock cycles and $f$ is the operating frequency in MHz.

Because an UltraScale slice contains twice as many LUTs as a 7-Series slice, the two eSlice figures are not strictly interchangeable. For reference, applying the 7-Series formula to CIRCE yields 2.36 and 2.43 KeSlices for R-SDP-1-b and R-SDP(G)-1-b, respectively; the cross-family area ratios discussed below should therefore be read as indicative.

In addition to absolute latency and area, Table 5 reports area-normalized throughput, $\eta_A = 1000/(t_{\text{ms}} A_{\text{KeSlice}})$, in op/s/KeSlice for **KeyGen**, **Sign**, and **Verify**. This metric captures the amount of end-to-end performance delivered per normalized FPGA area, but must be interpreted together with absolute latency and acceleration scope. In particular, designs that offload almost the complete protocol can achieve high $\eta_A$ through very low latency, whereas extremely small ISEs can obtain favorable normalized values from a very small area denominator despite retaining substantial execution in software.

Energy-per-operation is not included in Table 5 because the available works do not share a consistent power-measurement boundary. In particular, the CIRCE results of Table 4 are obtained from 65-nm post-synthesis ASIC simulations of selected computational kernels, whereas Table 5 reports end-to-end FPGA implementations, several of which do not provide comparable power data. Combining these measurements would therefore produce a potentially misleading energy-efficiency ranking.

The most direct comparison is provided by the two existing **CROSS** implementations [18], [37], which represent opposite architectural extremes. The standalone accelerator of [18] prioritizes minimum latency and requires 13.6 and 11.5 KeSlices for R-SDP-1-b and R-SDP(G)-1-b, respectively. CIRCE reduces these footprints to only 1.18 and 1.22 KeSlices, corresponding to approximately 11.5× and 9.4× lower normalized

**TABLE 5. Comparison with state-of-the-art FPGA implementations. Runtime is computed as kCC/$f$ and expressed in ms, where $f$ is the operating frequency in MHz. KeSlice refers to Xilinx 7-Series, while † denotes the UltraScale series. The final columns report area-normalized throughput $\eta_A$ in op/s/KeSlice for KeyGen, Sign, and verify.**

| Ref. | Design | Resources | | Freq. | KeyGen | | Sign | | Verify | | Throughput/Area | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | LUT/FF/BRAM/DSP | KeSlice | [MHz] | [kCC] | [ms] | [kCC] | [ms] | [kCC] | [ms] | KeyGen | Sign | Verify |
| [18] | **CROSS** R-SDP-1-b | 25393/9076/57.0/0 | 13.6 | 108 | 4.212 | 0.039 | 116.64 | 1.08 | 85.86 | 0.80 | 1885.37 | 68.08 | 91.91 |
| | **CROSS** R-SDP(G)-1-b | 27235/10348/37.0/27 | 11.5 | 119 | 11.071 | 0.093 | 90.916 | 0.76 | 62.713 | 0.53 | 935.02 | 114.42 | 164.07 |
| [37] | **CROSS** R-SDP-1-b | 370/12/0/0 | 0.046† | 90 | – | – | 37,504 | 416.71 | 19,554 | 217.27 | – | 52.17 | 100.06 |
| | **CROSS** R-SDP(G)-1-b | | | | – | – | 25,646 | 284.96 | 15,229 | 169.21 | – | 76.29 | 128.47 |
| [19] | LESS-L1-b | 54800/39900/59.5/0 | 21.3 | 200 | 29.06 | 0.15 | 77.54 | 0.39 | 174.5 | 0.87 | 312.99 | 120.38 | 53.96 |
| [20] | LESS-L1-b | 2892/971/8/0 | 1.747 | 78.6 | – | – | 3,578,328 | 45,525 | 3,150,238 | 40,079 | – | 0.0126 | 0.0143 |
| [21] | SDitH-L1-GF256 | 16592/8778/164.5/0 | 25.2 | 164 | 55 | 0.34 | 6,730 | 41.04 | 8,688 | 52.98 | 116.71 | 0.97 | 0.75 |
| | SDitH-L1-GF251 | 17423/16336/164.5/196 | 25.4 | 164 | 57 | 0.35 | 21,450 | 130.79 | 23,403 | 142.70 | 112.49 | 0.30 | 0.28 |
| [23] | MEDS-9923-bp | 24308/20154/112/66 | 20.4 | 116 | – | – | 7,562 | 65.19 | 7,485 | 64.53 | – | 0.75 | 0.76 |
| | MEDS-13220-b | 21680/19491/70/66 | 14.4 | 130 | – | – | 1,334 | 10.26 | 1,252 | 9.63 | – | 6.77 | 7.21 |
| [46] | ML-DSA-44 | 13975/6845/35/4 | 7.974 | 163 | 63.2 | 0.388 | 113.9 | 0.699 | 67.9 | 0.417 | 323.45 | 179.47 | 301.06 |
| [17] | SLH-DSA-SHAKE-128f | 10197/7357/8/0 | 3.573 | 150 | – | – | 2,895.583 | 19.30 | 286.064 | 1.91 | – | 14.50 | 146.75 |
| [25] | MAYO1 | 91266/42113/45/2 | 28.577 | 100 | 12.182 | 0.122 | 49.926 | 0.499 | 12.722 | 0.127 | 287.26 | 70.09 | 275.07 |
| [24] | Raccoon-128 | 13957/11284/30.5/4 | 7.4 | 222 | 104.642 | 0.47 | 278.898 | 1.26 | 101.642 | 0.46 | 287.52 | 107.25 | 293.77 |
| CIRCE | **CROSS** R-SDP-1-b | 9,459/3,376/0/1 | 1.18† | 200 | 242 | 1.21 | 24,727 | 123.64 | 9,929 | 49.65 | 700.38 | 6.85 | 17.07 |
| | **CROSS** R-SDP(G)-1-b | 9,733/3,375/0/38 | 1.22† | | 107 | 0.54 | 14,422 | 72.11 | 9,360 | 46.80 | 1517.91 | 11.37 | 17.51 |

area (5.8× and 4.7× when both designs are expressed with the 7-Series formula). At the opposite extreme, the ISE of [37] requires only 0.046 KeSlice but accelerates a narrower portion of the scheme and does not support **KeyGen**. For R-SDP-1-b, CIRCE completes **Sign** and **Verify** in 24.7 M and 9.9 M cycles, compared with 37.5 M and 19.6 M cycles for [37], corresponding to 1.5× and 2.0× fewer cycles. For R-SDP(G)-1-b, CIRCE requires 14.4 M and 9.4 M cycles against 25.6 M and 15.2 M, i.e., 1.8× and 1.6× fewer cycles. Including the higher operating frequency of CIRCE, the corresponding runtime reductions range from 3.4× to 4.4×. Thus, the additional area invested by CIRCE translates into a substantially broader end-to-end acceleration scope.

Across the wider set of signature accelerators, the principal strength of CIRCE is its compact absolute footprint. Apart from the minimal **CROSS** ISE of [37], CIRCE is among the most compact architectures in Table 5: its footprint is comparable to that of the area-centric LESS implementation [20], which however requires tens of seconds per signature, and substantially below the ML-DSA, SLH-DSA, MAYO, Raccoon, SDitH, MEDS, and high-performance LESS accelerators. This compactness is achieved while retaining complete **KeyGen**, **Sign**, and **Verify** acceleration and supporting both R-SDP and R-SDP(G) within the same architecture. For the balanced R-SDP-1-b parameter set, CIRCE completes **KeyGen**, **Sign**, and **Verify** in 1.21 ms, 123.64 ms, and 49.65 ms, respectively, while R-SDP(G)-1-b requires 0.54 ms, 72.11 ms, and 46.80 ms.

The normalized results further clarify this positioning. CIRCE achieves particularly strong **KeyGen** efficiency, reaching 700.38 and 1517.91 op/s/KeSlice for R-SDP-1-b and R-SDP(G)-1-b, respectively. Its **Sign** and **Verify** throughput-per-area values are lower than those of latency-oriented monolithic designs, but remain above several compact or software-dominated alternatives, including the area-centric LESS implementation, SDitH, and most MEDS configurations. Conversely, the ultra-compact **CROSS** ISE of [37] obtains high normalized **Sign** and **Verify** values because of its exceptionally small area, despite absolute runtimes of hundreds of milliseconds and the absence of **KeyGen** acceleration.

Regarding the individual accelerated kernels, the *Keccak* unit processes a complete block in 388 cycles, placing it between software-centric implementations such as [48], requiring 22,355 cycles, and more specialized hardware extensions [32], [34], [49], reporting 308–1,632 cycles. The *fp*-`Unit` is instead specific to the finite-field operations of **CROSS** and cannot be directly isolated against existing implementations exposing different acceleration boundaries.

Overall, CIRCE occupies an intermediate architectural region between the latency-optimized standalone **CROSS** accelerator and the minimum-area ISE. Rather than maximizing either absolute throughput or minimum hardware cost in isolation, it dedicates a limited amount of logic to the two dominant *Keccak* and finite-field bottlenecks, providing complete **KeyGen**, **Sign**, and **Verify** acceleration while retaining software programmability and crypto-agility across all 18 **CROSS** parameter sets and both R-SDP and R-SDP(G). The resulting combination of a 1.18–1.22 KeSlice footprint, end-to-end acceleration, and broad parameter-set support is the distinguishing design point targeted by CIRCE.

## VII. CONCLUSION

This work presented CIRCE, a tightly coupled RISC-V coprocessor that accelerates **CROSS** through a compact set of custom instructions. By targeting the dominant *Keccak*-based hashing and finite-field arithmetic kernels, CIRCE occupies an intermediate design point between fully monolithic accelerators and minimal instruction-set extensions, preserving RISC-V programmability while improving end-to-end execution time. The architecture supports both R-SDP and R-SDP(G) variants across all 18 official **CROSS** parameter sets without hardware retuning, enabling software-controlled crypto-agility within a single lightweight instance.

The FPGA and ASIC results show that this balanced approach provides meaningful acceleration with limited area overhead, while the instruction-level power analysis confirms consistent energy savings on the accelerated kernels. More broadly, these results indicate that accelerating only the dominant kernels through a tightly coupled interface recovers most of the benefit of dedicated hardware for **CROSS**, while leaving protocol control and parameter-set selection in software. Its modular organization separates the CV-X-IF integration shell and Keccak datapath from the scheme-specific finite-field arithmetic, providing a clear extension path in which common integration and hashing infrastructure can be retained while arithmetic kernels and instruction semantics are adapted to other PQC workloads. Investigating such extensions and quantifying their hardware and performance trade-offs represents a natural direction for future work.

## REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, 1997.

[2] D. Moody, R. Perlner, D. Smith-Tone, A. Robinson, R. Peralta, and L. Chen, *Post-Quantum Cryptography: NIST's Plan for the Future*, NIST Standard IR 8105, 2016.

[3] National Institute of Standards and Technology (NIST). (May 2026). *Post-Quantum Cryptography: Additional Digital Signature Schemes - Round 3 Additional Signatures*. Accessed: June 2, 2026. [Online]. Available: https://csrc.nist.gov/projects/pqc-dig-sig/round-3-additional-signatures

[4] M. Baldi, S. Bitzer, A. Pavoni, P. Santini, A. Wachter-Zeh, and V. Weger, "Zero knowledge protocols and signatures from the restricted syndrome decoding problem," in *Public-Key Cryptography—PKC 2024* (Lecture Notes in Computer Science), Q. Tang and V. Teague, Eds., Cham, Switzerland: Springer, 2024, pp. 243–274.

[5] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology—CRYPTO*(Lecture Notes in Computer Science), A. M. Odlyzko, Ed., Heidelberg, Germany: Springer, 1987, pp. 186–194.

[6] National Institute of Standards and Technology (U.S.), *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, NIST Standard FIPS 202, 2015.

[7] G. Alagic, M. Bros, P. Ciadoux, Q. Dang, T. H. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, H. Silberg, D. Smith-Tone, and N. Waller, *Status Report on the Second Round of the Additional Digital Signature Schemes for the NIST Post-Quantum Cryptography Standardization Process*, NIST Standard IR 8610, Gaithersburg, MD, USA, 2026.

[8] European Union Agency for Cybersecurity (ENISA). (May 2021). *Post-Quantum Cryptography: Current State and Quantum Mitigation*. [Online]. Available: https://www.enisa.europa.eu/publications/post-quantum-cryptography-current-state-and-quantum-mitigation

[9] European Cybersecurity Certification Group. (Apr. 2025). *Agreed Cryptographic Mechanisms*. [Online]. Available: https://certification.enisa.europa.eu/document/download/a845662b-aee0-484e-9191-890c4cfa7aaa_en

[10] Bundesamt für Sicherheit in der Informationstechnik (BSI). (May 2022). *Quantum-safe Cryptography: Fundamentals, Current Developments and Recommendations*. Federal Office for Information Security (BSI), Bonn, Germany. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.html

[11] *Post-Quantum Cryptography Standardization: Evaluation Process*, NIST Computer Security Resource Center, 2017. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-process

[12] M. Abbasi, F. Cardoso, P. Váz, J. Silva, and P. Martins, "A practical performance benchmark of post-quantum cryptography across heterogeneous computing environments," *Cryptography*, vol. 9, no. 2, p. 32, May 2025.

[13] P. Karl, J. Schupp, T. Fritzmann, and G. Sigl, "Post-quantum signatures on RISC-V with hardware acceleration," *ACM Trans. Embedded Comput. Syst.*, vol. 23, no. 2, pp. 1–23, Mar. 2024.

[14] A. Dolmeta, M. Martina, and G. Masera, "ATHOS: A hybrid accelerator for PQC CRYSTALS-algorithms exploiting new CV-X-IF interface," *IEEE Access*, vol. 12, pp. 182340–182352, 2024.

[15] J. Lopez-Valdivieso and R. Cumplido, "Design and implementation of hardware–software architecture based on hashes for SPHINCS+," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 17, no. 4, pp. 1–22, Dec. 2024.

[16] Z. Ye, X. Li, C. Wang, R. C. C. Cheung, and K. Huang, "RVSLH: Acceleration of postquantum standard SLH-DSA with customized RISC-V processor," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 33, no. 7, pp. 1999–2003, Jul. 2025.

[17] S. Deshpande, Y. Lee, C. Karakuzu, J. Szefer, and Y. Paek, "SPHINCSLET: An area-efficient accelerator for the full SPHINCS+ digital signature algorithm," *ACM Trans. Embedded Comput. Syst.*, vol. 24, no. 5, pp. 1–19, Sep. 2025.

[18] P. Karl, F. Antognazza, A. Barenghi, G. Pelosi, and G. Sigl, "High-performance FPGA accelerator for the post-quantum signature scheme CROSS," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2026, no. 3, pp. 98–121, Jul. 2026.

[19] L. Beckwith, R. Wallace, K. Mohajerani, and K. Gaj, "A high-performance hardware implementation of the LESS digital signature scheme," in *Post-Quantum Cryptography—PQCrypto 2023* (Lecture Notes in Computer Science), T. Johansson and D. Smith-Tone, Eds., Cham, Switzerland: Springer, 2023, pp. 57–90.

[20] G. Cutrera, A. Dolmeta, V. Piscopo, M. Martina, and G. Masera, "A lightweight accelerator for the LESS digital signature scheme," *Cryptography*, vol. 10, no. 4, p. 45, Jul. 2026.

[21] S. Deshpande, J. Howe, J. Szefer, and D. Yue, "SDitH in hardware," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2024, no. 2, pp. 215–251, Mar. 2024.

[22] S. Halmans, N. Höher, D. Hesse, S. Deshpande, J. Szefer, and T. Güneysu, "Storing less in-the-head: An area-efficient hardware architecture for SDitH-v2," *Cryptology ePrint Archive*, 2026.

[23] S. Deshpande, Y. Lee, M. Nawan, K. Nawaz, R. Niederhagen, Y. Paek, and J. Szefer, "Unified MEDS accelerator," in *Selected Areas in Cryptography—SAC 2025* (Lecture Notes in Computer Science), C. Boura, A. Mashatan, and A. Miri, Eds., Cham, Switzerland: Springer, 2026, pp. 391–421.

[24] Z. Ni, A. Khalid, Z. Zhang, Y. Cui, W. Liu, and M. O'Neill, "HRaccoon: A high-performance configurable SCA resilient raccoon hardware accelerator," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2025, no. 3, pp. 413–436, Jun. 2025.

[25] F. Hirner, M. Streibl, F. Krieger, A. C. Mert, and S. S. Roy, "Whipping the multivariate-based MAYO signature scheme using hardware platforms," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Dec. 2024, pp. 3421–3435.

[26] S. Manasidis, Q. Norga, S. Kundu, and I. Verbauwhede, "MPSpeed: Implementing and optimizing MPC-in-the-head digital signatures in hardware: Accelerating mirath on FPGA," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2026, no. 3, pp. 982–1006, Jul. 2026.

[27] H. Hart, P. Mondal, S. Kundu, S. Adhikary, A. Karmakar, and C. Li, "Lightening CROSS: Memory optimized implementations of CROSS," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2026, no. 2, pp. 630–658, Apr. 2026.

[28] J. Schupp, M. Gianvecchio, A. Barenghi, P. Karl, G. Pelosi, and G. Sigl, "Optimizing the post quantum signature scheme CROSS for resource constrained devices," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2026, no. 3, pp. 224–246, Jul. 2026.

[29] S. Bauer, F. De Santis, and F. Wilde, "MAYO Lite: A Low-RAM implementation of the MAYO signature scheme," *Cryptol. ePrint Archive*, 2026.

[30] T. Chou, "Representations of elementary vectors in VOLE-in-the-head-based schemes," *Cryptol. ePrint Archive*, 2025.

[31] J. L. Delgado, "Cross-Signature Signing-Key Recovery and Domain-Separation Repair for SDitH v2," *Cryptol. ePrint Archive*, 2026.

[32] T. Fritzmann, G. Sigl, and J. Sepúlveda, "RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 239–280, Aug. 2020.

[33] A. Dolmeta, S. Di Matteo, E. Valea, M. Carmona, A. Loiseau, M. Martina, and G. Masera, "TYRCA: A RISC-V tightly-coupled accelerator for code-based cryptography," in *Proc. Design, Autom. Test Eur. Conf. (DATE)*, Mar. 2025, pp. 1–7.

[34] A. Dolmeta, V. Piscopo, M. Mirigaldi, M. Martina, and G. Masera, "RISC-V based Keccak Co-processor for NIST post-quantum cryptography standards," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2025, pp. 1–5.

[35] K. Hepola, T. R. Arachchige, J. Multanen, and P. Jääskeläinen, "Fully automatic compiler retargeting and CV-X-IF hardware interface generation for RISC-V custom instructions," in *Proc. IEEE Nordic Circuits Syst. Conf. (NorCAS)*, Oct. 2024, pp. 1–7.

[36] K. Hepola, T. Ranasinghe Arachchige, J. Multanen, and P. Jääskeläinen, "Automatically retargeting hardware and code generation for RISC-V custom instructions," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 33, no. 10, pp. 2852–2861, Oct. 2025.

[37] L. Ricci, M. Donati, L. Fanucci, and P. Nannipieri, "A tightly-coupled RISC-V ISA extension to speed-up a code-based post quantum digital signature scheme," *IEEE Access*, vol. 14, pp. 60784–60791, 2026.

[38] M. Baldi, A. Barenghi, S. Bitzer, P. Karl, F. Manganiello, A. Pavoni, G. Pelosi, F. Pintore, and P. Santini, *CROSS: Codes and Restricted Objects Signature Scheme—Algorithm Specifications and Supporting Documentation*, 2025.

[39] Synopsys. *ASIP Designer*. Accessed: Sep. 12, 2025. [Online]. Available: https://www.synopsys.com/dw/ipdir.php?ds=asip-designer.

[40] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. (Jan. 2011). *The Keccak Reference, Version 3.0*. [Online]. Available: https://keccak.team/files/Keccak-reference-3.0.pdf

[41] A. Dolmeta, M. Martina, and G. Masera, "Exploring the new CV-X-IF interface to customize RISC-V instruction sets: A case of study in cryptography," in *Applications in Electronics Pervading Industry, Environment and Society (APPLEPIES 2024)* (Lecture Notes in Electrical Engineering), M. Ruo Roch, F. Bellotti, R. Berta, M. Martina, and P. Motto Ros, Eds., Cham, Switzerland: Springer, 2025, pp. 39–47.

[42] V. Piscopo, A. Dolmeta, M. Mirigaldi, M. Martina, and G. Masera, "A deep dive into integration methodologies in RISC-V," in *Proc. 22nd ACM Int. Conf. Comput. Frontiers: Workshops Special Sessions*, NY, NY, USA, May 2025, pp. 30–33.

[43] S. Machetti, P. D. Schiavone, T. Christoph Müller, M. Peón-Quirós, and D. Atienza, "X-HEEP: An open-source, configurable and extendible RISC-V microcontroller for the exploration of ultra-low-power edge accelerators," 2024, *arXiv:2401.05548*.

[44] C. A. Melchor, S. Bettaieb, L. Bidoux, T. Feneuil, P. Gaborit, N. Gama, S. Gueron, J. Howe, A. Hülsing, D. Joseph, A. Joux, M. Kulkarni, E. Persichetti, T. H. Randrianarisoa, M. Rivain, and D. Yue, *The Syndrome Decoding in the Head (SD-in-the-Head) Signature Scheme: Algorithm Specifications and Supporting Documentation*, Feb. 2025.

[45] T. Chou, R. Niederhagen, E. Persichetti, L. Ran, T. H. Randrianarisoa, K. Reijnders, S. Samardjiska, and M. Trimoska, *MEDS: Matrix Equivalence Digital Signature*, Jul. 2023.

[46] N. Gupta, A. Jati, A. Chattopadhyay, and G. Jha, "Lightweight hardware accelerator for post-quantum digital signature CRYSTALS-dilithium," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 8, pp. 3234–3243, Aug. 2023.

[47] AMD Xilinx. (Dec. 2022). *UltraScale Archit. Configurable Log. Block*. [Online]. Available: https://docs.xilinx.com/r/en-U.S./ug574-ultrascale-clb
[48] C. Gewehr, L. Luza, and F. G. Moraes, "Hardware acceleration of crystals-kyber in low-complexity embedded systems with RISC-V instruction set extensions," *IEEE Access*, vol. 12, pp. 94477–94495, 2024.
[49] J. Lee, W. Kim, and J.-H. Kim, "A programmable crypto-processor for National Institute of Standards and Technology post-quantum cryptography standardization based on the RISC-V architecture," *Sensors*, vol. 23, no. 23, p. 9408, Nov. 2023.



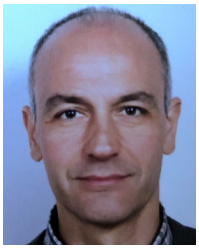
**VALERIA PISCOPO** (Member, IEEE) received the M.Sc. degree in electronic engineering from Politecnico di Torino, Italy, in 2024, where she is currently pursuing the Ph.D. degree in electronics and communications engineering. Her research focuses on the design and optimization of specialized cryptographic hardware accelerators for secure and energy-efficient embedded systems, with particular emphasis on the RISC-V ecosystem.



**ALESSANDRA DOLMETA** (Member, IEEE) received the M.Sc. degree in electronic engineering and the Ph.D. degree in electronics and communications engineering from Politecnico di Torino, Italy, in 2022 and 2025, respectively. Since 2026, she has been a Postdoctoral Researcher with the Department of Electronics and Telecommunications, Politecnico di Torino. Her research interests include computer architecture, hardware security, post-quantum cryptography, and the design of specialized hardware accelerators for secure and low-power embedded systems. Her current research focuses on hardware-software co-design for the RISC-V ecosystem, custom instruction set extensions, and quantum-resistant cryptographic accelerators.



**MAURIZIO MARTINA** (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in electronic engineering and electronic and communications engineering from Politecnico di Torino, Italy, in 2000 and 2004, respectively. He is currently a Professor with the Department of Electronics and Telecommunications, Politecnico di Torino. He has authored or co-authored more than 200 scientific publications and holds two patents. His research interests include computer architecture and VLSI design of digital integrated circuits for image and video coding, forward error correction, cryptography, and artificial intelligence. He served as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, from 2018 to 2022, and a Guest Editor for several special issues, including BioCAS 2017 Special Issue in IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS and ISCAS 2023 special issue in IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS. He has been a member of the organizing and technical committees of several IEEE conferences, including BioCAS 2017, ICECS 2019, AICAS 2020, and PRIME, since 2020.



**GUIDO MASERA** (Senior Member, IEEE) received the Dr.-Ing. (summa cum laude) and Ph.D. degrees in electronic engineering from Politecnico di Torino, Italy. He has been a Professor with the Electronics and Telecommunications Department, Politecnico di Torino, since 1992. He has more than 200 publications, two patents, and has been a designer of several ASIC components. His research interests include several aspects of the design of digital integrated circuits and systems, with a special emphasis on high-performance architectures for communications, forward error correction, image and video coding, cryptography, and hardware accelerators for machine learning. He is an Associate Editor of *Electronics* (MDPI) and a former Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, and *IET Circuits, Devices & Systems*.