Exploring the New CV-X-IF Interface to Customize RISC-V Instruction Sets: A Case of Study in Cryptography

Exploring the new CV-X-IF Interface to Customize RISC-V Instruction Sets: a Case of Study in Cryptography

Alessandra Dolmeta[1], Maurizio Martina[2], and Guido Masera[2]

Politecnico di Torino, Italy
name.surname@polito.it

Abstract. The development of embedded systems has notably evolved, emphasizing the inclusion of specialized accelerators within microcontrollers to enhance performance and efficiency. A key factor in this integration is the level of connectivity between the central processing unit (CPU) and these accelerators, as well as the specific type of data being handled. This article presents a method to tailor a RISC-V instruction set to specific requirements, enhancing performance while preserving flexibility; this method is explored on a relevant case of study, the Keccak cryptographic hash function. By leveraging the CV32E40PX core and the Core-V eXtension Interface (CV-X-IF), this approach introduces innovative opportunities in accelerator integration. The CV-X-IF simplifies the addition of new instructions to the Instruction Set Architecture (ISA) and streamlines the integration of tightly coupled accelerators. Consequently, these accelerators become more adaptable and can be employed with any RISC-V core, broadening their applicability and potential impact. Experimental results are analyzed and compared with respect to the same accelerator, integrated as loosely-coupled accelerator using the Extendible Accelerator Interface (XAIF). Moreover, the designed RISC-V units are likened to other implementations available in the literature to highlight the potential of the adopted method.

Keywords: Hardware design, Accelerator, RISC-V, CV-X-IF, XAIF Post-Quantum Cryptography, SHA-3, Keccak

1 Introduction

Advances in quantum computing significantly threaten the security of conventional public-key cryptosystems. In response, Post-Quantum Cryptography (PQC) has been developed to safeguard data confidentiality in communication channels. The National Institute of Standards and Technology (NIST) has taken the lead in standardizing PQC, announcing selected algorithms for standardization and round 4 candidates in 2022 [1]. Currently, CRYSTALS-Kyber is one of the most promising schemes and it can be implemented on classical computers and integrated into contemporary communication infrastructures. Given the memory-intensive nature and repetitive operations of PQC, extensive efforts have been

made to expedite its processes through hardware and software enhancements. In particular, the progression of embedded systems has experienced a significant shift towards incorporating specialized accelerators into microcontrollers, intending to boost performance and efficiency. RISC-V, an open instruction set architecture (ISA), fosters processor innovation and enables open-source hardware development with the integration of domain-specific ISA extensions, accelerators, and co-processors. A crucial aspect of this integration revolves around the degree of connectivity between the central processing unit (CPU) and these accelerators, as well as the nature of the data being processed. Loosely-coupled accelerators, typically memory-mapped and connected via system buses, handle compute-heavy tasks with large data sets. In contrast, tightly coupled accelerators and co-processors reside within the CPU's microarchitecture, necessitating core and toolchain modifications. The emerging **eXtension Interface** (CV-X-IF) allows seamless support of co-processors, enhancing the CPU with custom or standardized instructions without altering the CPU's decode unit. This interface ensures low-latency and tightly-integrated read and write access to the CPU register file from the co-processor, utilizing unused opcodes within the CPU while avoiding those reserved or used by RISC-V International.

Organization. Section 2 analyzes hardware-software acceleration methods for PQC algorithms and Keccak algorithm. Section 3 covers implementation and integration methods, focusing on the CV-X-IF integration method. Section 4 discusses results and compares them with other approaches. Finally, Section 5 outlines future work and summarizes findings.

2 Background

Software-only implementations of PQC present notable inefficiencies due to the intricate schemes, frequent memory accesses, and iterative operations inherent to these algorithms. Consequently, researchers have increasingly focused on hardware-based solutions to accelerate PQC implementations, especially specific primitives and hashing functions, such as the ones included in Keccak [12]. These accelerators can be broadly categorized into three distinct methods: i) **loosely coupled accelerators**, ii) **tightly coupled accelerators**, and iii) **coprocessors**. Loosely coupled accelerators, integrated onto system buses like OBI, AXI, and AHB, handle compute-heavy tasks efficiently. For instance, [2] uses HW/SW co-design techniques to accelerate Number Theoretic Transform (NTT) and hash generation. The same approach has been used for signature algorithms in [3]. [4] presents a memory-mapped Keccak accelerator for Kyber. Tightly coupled accelerators, embedded within the CPU's microarchitecture, speed up PQC tasks through custom instruction sets and optimized functional units. [5] integrates tightly coupled accelerators to speed up lattice-based PQC, with an optimized and masked version proposed later in [6]. [7] provides an Instruction Set Extension (ISE) for finite field operations with result reduction. Meanwhile, [9] exploits a PQC arithmetic unit, featuring accelerations for *fqmul* and *reduce*, along with butterfly operation logic for the CRYSTALS algorithm. Coprocessors, connected

to the CPU core via dedicated interfaces, offer another layer of acceleration by handling specific computational tasks. The coprocessor introduced in [11] adopts a domain-specific approach, utilizing a matrix extension of the RISC-V tailored for Module-LWE. Meanwhile, [10] suggests a RISC-V Instruction Set Extension (ISE) optimized for NIST PQC standard algorithms and round 4 candidates.

2.1 Keccak

Keccak f permutation, characterized by its parameters bit-rate (r) and capacity (c), is a fundamental component of the Keccak cryptographic hash function. A schematic of the algorithm is reported in Alg. 1.

Algorithm 1 Keccak Permutation

Input: State array A of size 200 bytes, number of rounds n_r
Output: Permuted state array A

Initialize round constants
$RC[i, j, k]$ for $0 \leq i < 24$, $0 \leq j < 5$, $0 \leq k < 2$
for $round = 0$ **to** $n_r - 1$ **do**
  θ(A) ▷ Non-linear mixing operation on the state
  ρ(A) ▷ Diffusion by rotating bits in each lane
  π(A) ▷ Diffusion by rearraging lanes
  χ(A) ▷ Non-linear mixing operation within lane
  ι(A, round) ▷ Injects a round constant into the state
end

The width of the state in the Keccak f permutation is determined by the sum of r and c, resulting in a 1600-bit state comprised of a 5x5 matrix of 64-bit words. With a width of 1600 bits, the permutation operates through 24 rounds (n_r). Each of these rounds involves five distinct steps: θ, ρ, π, χ, and ι [12]. These steps collectively transform the permutation state, contributing to its cryptographic strength and ensuring robustness against attacks. Through its carefully designed structure and intricate round operations, Keccak permutation is a cornerstone in modern cryptographic protocols, offering resilience and security.

2.2 Hardware Implementations

In this manuscript, we introduce two different versions of the same Keccak accelerator (for further details, refer to [4]). The first version (hereinafter referred to as Keccak XAIF) communicates with the primary RISC-V CPU core via the novel Extendible Accelerator Interface (XAIF), while the second version (Keccak CV-X-IF) uses the CV-X-IF extension interface, facilitating smooth integration and offering flexibility, low implementation cost, and low latency. The XAIF interface employs a register file for processor control and is compatible with various protocols like APB, AXI-Lite, OBI, and AXI, enhancing its adaptability across microcontrollers. Descriptions of required registers in *hjson* format enable the generation of System-Verilog structures, such as `i_data_regfile` and

`i_ctrl_regfile`, facilitating seamless integration and communication. System management involves distinct software drivers overseeing multiple accelerators, and managing their setup, operation, and synchronization. Initialization configures components like interrupt controllers and DMA, ensuring seamless interaction between the CPU and accelerator module. The second version, detailed in Section 3, will be compared comprehensively with the first in Section 4.

3 CV-X-IF

The proposed hardware architecture for the coprocessor module is depicted on the right side of Figure 1. The CV-X-IF has been successfully utilized and integrated into the CV32E40P core, resulting in the labeled variant known as CV32E40PX[1].

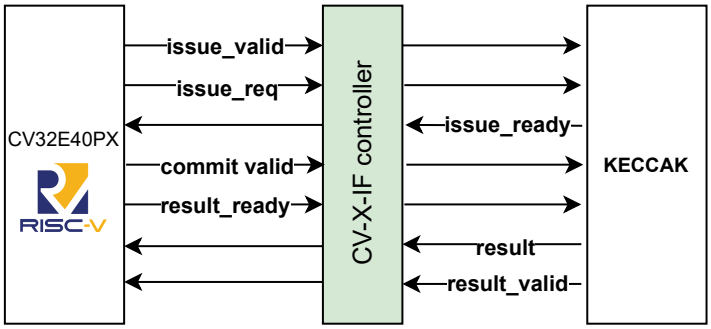

Fig. 1. CV/X.IF Interface.

In this study, we leverage the CV-X-IF to show the integration of a coprocessor specialized in the acceleration of a cryptographic application. Specifically, we target the *Keccak* computing kernel, that is present in numerous PQC algorithms, such as CRYSTALS-Kyber. This module's interface is meticulously crafted to ensure seamless communication between the core and the accelerator unit, employing structured packaged signals (*italicized*). Initially, the wrapper scrutinizes the instruction opcode sourced from the instruction field of the issue request package (*issue_req* in Figure 1). Upon identifying a match with one of the custom instructions' opcode, it activates the issue valid' signal, signaling the accelerator's readiness to receive data via *issue_resp*'. The validity of input source registers is also determined by the issue request package (rs_valid signals from *issue req*' package). Upon validation, the accelerator samples its inputs synchronously in the same clock cycle, ensuring efficient data transfer. Once the instruction is executed, if the core is prepared to receive the result and the result is deemed valid, it is written back into the register file via the result interface, as

[1] `https://github.com/esl-epfl/cv32e40px`

illustrated on the left side of Figure 1. It's important to note that instructions offloaded by the RISC-V core during its ID stage are speculative and require thorough verification before the accelerator can write back results through commit and result interfaces. For more detailed information about the protocol of this interface, refer to the original repository [2].

4 Result and Comparison

In this section, we report our implementation results and their comparison with other prior work in the state-of-art. Table 1 illustrates the cycle counts for executing the same function on X-HEEP platform using the two different methods, XAIF and CV-X-IF, and provides a comparison of their performance in terms of reference and accelerated cycle counts, along with the corresponding speed-up factors. X-HEEP (eXtendable Heterogeneous Energy-Efficient Platform) is an open-source RISC-V microcontroller described in SystemVerilog [3].

Table 1. Cycle counts for RISC-V platforms

	Method	**Reference**	**Accelerated**	**Speed-up factor**
Keccak	XAIF	56,556	1,065	55×
	CV-X-IF		553	102×

The `-O2` flag is used both for pure software and accelerated simulations of Table 1. The Keccak function's reference implementation requires 56,556 clock cycles. Accelerated using the XAIF method, the cycle count drops to 1,065, achieving a 55x speed-up. The CV-X-IF method further reduces the cycle count to 553, resulting in a 102x speed-up. These results highlight the effectiveness of both acceleration methods, with CV-X-IF demonstrating higher efficiency than XAIF, significantly improving performance on RISC-V platforms.

Table 2. Area Results for Keccak CV-X-IF and Keccak XAIF

Parameter	**Keccak CV-X-IF** [μm^2]	**Keccak XAIF** [μm^2]
Combinational area	181,875	180,011
Buf/Inv area	7,005	6,753
Noncombinational area	239,527	238,590
Total cell area	**421,403**	**418,602**

Table 2 presents a comparison of the ASIC area results for two accelerators, implemented on 65 µm technology. Design Vision by Synopsys is employed for comprehensive analysis, elaboration, and compilation of the design. The frequency at which the two system have been synthetized is the same, 285 MHz. The analysis shows that Keccak XAIF has a higher complexity in interconnect and interface, with more ports (184,291 vs. 166,298) and nets (236,430 vs. 226,114) than Keccak CV-X-IF. Despite this, Keccak XAIF has a slightly lower total

[2] `https://github.com/openhwgroup/core-v-xif/tree/main`
[3] `https://github.com/esl-epfl/x-heep`

cell count (115,903 vs. 116,435), with fewer combinational (76,281 vs. 76,874) and sequential cells (32,670 vs. 32,809). Keccak CV-X-IF has larger area metrics, as shown in Table 2. The total cell area for Keccak CV-X-IF is 421,403 μm^2, slightly higher than Keccak XAIF's 418,602 μm^2. These findings indicate that although both accelerators are efficiently designed, Keccak XAIF achieves a marginally more compact circuit, suggesting a slight advantage in area efficiency while maintaining a comparable overall cell structure. This difference is likely attributable to the additional control unit required by the CV-X-IF interface, known as the CV-X-IF controller. In the case of Keccak XAIF, the hardware IP is managed in software through dedicated hardware support, eliminating the need for a separate control unit. Conversely, the CV-X-IF approach does not require a software driver, instead relying on the hardware controller to manage the accelerator directly. While this adds to the area overhead for each accelerator, the impact can be mitigated when multiple accelerators are integrated, as the CV-X-IF controller can be shared among them. Therefore, the apparent area inefficiency of Keccak CV-X-IF could be offset in systems with multiple accelerators, where the shared control unit leads to a more favorable area utilization.

Table 3. Efficiency comparison between Keccak CV-X-IF and Keccak XAIF

	Method	**Freq.** [MHz]	**Total Cell Area** [μm^2]	**Latency** [µs]	**Efficiency** [$\mu m^2 \cdot$ ms]
Keccak	XAIF	666	418,593	1,598	668,912
	CV-X-IF	555	423,841	0.995	421,722

The Table 3 shows the overall performance of the two approaches, considering their maximum frequency. As seen, the XAIF Keccak achieves a higher synthesis frequency compared to CV-X-IF. This was expected since the CV-X-IF version, being tightly coupled with the core, slightly increases its internal critical path. Despite the different synthesis frequencies, the area results confirm the previous analysis from Table 2. Latency was evaluated as the product of each approach's clock period and the number of clock cycles reported in Table 1. Despite its lower frequency, CV-X-IF outperforms XAIF, taking less than 1 µs to execute Keccak compared to XAIF's 1,598 µs. Overall, neither the lower frequency nor the higher area makes CV-X-IF unfavorable. In fact, when evaluating efficiency as the product of latency and total cell area, the CV-X-IF version outperforms XAIF by 36.95%.

The Table 4 shows significant performance variations across different methods and devices. The CVA6 (CV-X-IF) with a coprocessor method achieves the lowest Keccak operation count at 1,632 clock cycles [10]. The PULPissimo with a memory-mapped method performs the transformation with 1,348 clock cycles [4]. In comparison, the tightly coupled methods on a general RISC-V device and PULPino yield lower clock cycles, with 404 [8] and 308 [5] clock cycles, respectively. Our implementations on the RISC-V (XHEEP) platform show that the XAIF method achieves 1,065 Keccak clock cycles, demonstrating strong performance, and a lower amount of clock cycles with respect to its loosely-coupled counterparts. The CV-X-IF method achieves 553 clock cycles. This is 36% higher

Table 4. Clock cycle comparison of methods on different RISC-V devices for Keccak operations

Reference	Device	Method	Keccak
[10]	CVA6 (CV-X-IF)	Coprocessor	1,632
[8]	RISC-V	Tightly coupled	404
[5]	PULPino	Tightly coupled	308
[4]	RISC-V (PULPissimo)	Memory-mapped	1,348
OURS	RISC-V (XHEEP)	XAIF CV-X-IF	**1065** **553**

than [8] and 79% higher than [5], but still competitive with the state of art. Despite the higher number of clock cycles for CV-X-IF compared to other tightly coupled accelerators, its integration method is simpler, does not require toolchain modifications, and can use loosely coupled accelerators easily adapted to connect directly to the core pipeline.

5 Conclusion

The CV-X-IF interface marks a notable advancement in RISC-V acceleration, and our work has demonstrated its potential by integrating it with the CV32E40PX core. Interest in this interface is on the rise, particularly with the release of a recent candidate version. Our future efforts will focus on further exploring its capabilities, optimizing performance, and implementing security measures to prevent potential attacks. While the XAIF approach also has its advantages, the decision to choose one over the other depends on factors such as the specific applications and design requirements. Nonetheless, the CV-X-IF interface shows great potential and promises significant advancements in RISC-V acceleration.

Acknowledgments. This work was funded by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. This work received funding from the Key Digital Technologies Joint Undertaking (KDT-JU) under grant agreement No 101095947

References

1. National Institute of Standards and Technology: PQC Candidates to be Standardized and Round 4. `https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4`, (2022).
2. Fritzmann, Tim, Sharif, Uzair, Müller-Gritschneder, Daniel, Reinbrecht, Cezar, Schlichtmann, Ulf, Sepulveda, Johanna: Towards Reliable and Secure Post-Quantum Co-Processors based on RISC-V. 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1148-1153 (2019). doi: 10.23919/DATE.2019.8715173
3. Karl, Patrick, Schupp, Jonas, Fritzmann, Tim, Sigl, Georg: Post-Quantum Signatures on RISC-V with Hardware Acceleration. ACM Trans. Embed. Comput. Syst. (2023). doi: 10.1145/3579092

4. Dolmeta, Alessandra, Mirigaldi, Mattia, Martina, Maurizio, Masera, Guido: Implementation and integration of Keccak accelerator on RISC-V for CRYSTALS-Kyber. Proceedings of the 20th ACM International Conference on Computing Frontiers, pp. 381–382 (2023). doi: 10.1145/3587135.3591432
5. Fritzmann, Tim, Sigl, Georg, Sepúlveda, Johanna: RISQ-V: Tightly Coupled RISC-V Accelerators for Post-Quantum Cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems, vol. 2020, no. 4, pp. 239–280 (Aug. 2020). doi: 10.13154/tches.v2020.i4.239-280
6. Fritzmann, Tim, Van Beirendonck, Michiel, Basu Roy, Debapriya, Karl, Patrick, Schamberger, Thomas, Verbauwhede, Ingrid, Sigl, Georg: Masked Accelerators and Instruction Set Extensions for Post-Quantum Cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems, vol. 2022, no. 1, pp. 414–460 (Nov. 2021). doi: 10.46586/tches.v2022.i1.414-460
7. Alkim, Erdem, Evkan, Hülya, Lahr, Norman, Niederhagen, Ruben, Petri, Richard: ISA Extensions for Finite Field Arithmetic - Accelerating Kyber and NewHope on RISC-V. Cryptology ePrint Archive, Paper 2020/049 (2020). url: https://eprint.iacr.org/2020/049
8. Huang, Junhao, Adomnicăi, Alexandre, Zhang, Jipeng, Dai, Wangchen, et Al.: Revisiting Keccak and Dilithium Implementations on ARMv7-M. IACR Transactions on Cryptographic Hardware and Embedded Systems, vol. 2024, no. 2, pp. 1–24 (Mar. 2024). doi: 10.46586/tches.v2024.i2.1-24
9. Nannipieri, Pietro, Di Matteo, Stefano, Zulberti, Luca, Albicocchi, Francesco, Saponara, Sergio, Fanucci, Luca: A RISC-V Post Quantum Cryptography Instruction Set Extension for Number Theoretic Transform to Speed-Up CRYSTALS Algorithms. IEEE Access, vol. 9, pp. 150798-150808 (2021). doi: 10.1109/ACCESS.2021.3126208
10. Lee, Jihye, Kim, Whijin, Kim, Ji-Hoon: 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, article no. 9408 (2023). doi: 10.3390/s23239408
11. Zhao, Yifan, Xie, Ruiqi, Xin, Guozhu, Han, Jun: A high-performance domain-specific processor with matrix extension of RISC-V for module-LWE applications. IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 69, no. 7, pp. 2871–2884 (2022). doi: 10.1109/TCSI.2022.3162679
12. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: FIPS PUB 202 - SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (2015). Available online: `https://keccak.team/hardware.html`
13. May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)
14. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
15. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)
16. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
17. National Center for Biotechnology Information, `http://www.ncbi.nlm.nih.gov`