A Deep Dive into Integration Methodologies in RISC-V

Valeria Piscopo
Politecnico di Torino
Turin, Italy
valeria.piscopo@polito.it

Alessandra Dolmeta
Politecnico di Torino
Turin, Italy
alessandra.dolmeta@polito.it

Mattia Mirigaldi
Politecnico di Torino
Turin, Italy
mattia.mirigaldi@polito.it

Maurizio Martina
Politecnico di Torino
Turin, Italy
maurizio.martina@polito.it

Guido Masera
Politecnico di Torino
Turin, Italy
guido.masera@polito.it

Abstract

The integration methodology can significantly affect the performance of dedicated accelerators. This work explores this aspect, considering Keccak, a pivotal hashing standard in Post-Quantum Cryptography (PQC), as a case of study. The paper presents three versions of KRONOS (Keccak RISC-V Optimized eNgine fOr haShing): a loosely coupled memory-mapped accelerator, a tightly coupled approach, and an Instruction Set Extension (ISE). The latter two versions leverage the Core-V eXtension InterFace (CV-X-IF) interface, with and without, respectively, an additional register file to store the Keccak state. Experimental results demonstrate that the tightly coupled integration achieves the highest throughput-to-area ratio (1.070 Mb/(s·LUTs)), outperforming both the loosely coupled (0.937 Mb/(s·LUTs)) and coprocessor-based (0.386 Mb/(s·LUTs)) implementations. This confirms that a tightly integrated accelerator balances resource consumption and performance most effectively.

CCS Concepts

• **Hardware**; • **Security and privacy**;

Keywords

Hardware Accelerators, RISC-V, Keccak

ACM Reference Format:
Valeria Piscopo, Alessandra Dolmeta, Mattia Mirigaldi, Maurizio Martina, and Guido Masera. 2025. A Deep Dive into Integration Methodologies in RISC-V. In *22nd ACM International Conference on Computing Frontiers (CF Companion '25), May 28–30, 2025, Cagliari, Italy*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3706594.3726969

1 Introduction

In the domain of embedded systems design, the increasing need to address computational challenges has led to the integration of tailor-made accelerators. Regardless of the specific application, the integration methodology significantly impacts the overall system's performance and efficiency. Three primary integration methodologies exist: **loosely coupled**, **tightly coupled**, and **coprocessors**.

Loosely coupled accelerators, typically implemented as memory-mapped units, communicate with the CPU via system buses (e.g., AXI, AHB). They operate independently, making them highly versatile and reusable across different architectures. However, due to their detached nature, data transfer overhead and latency from memory accesses can limit performance gains (i.e., [3], [2]).
Tightly coupled accelerators, on the other hand, are integrated directly into the CPU microarchitecture and interact with internal registers. This approach minimizes data movement latency and improves computational efficiency but requires modifications to the processor core and its toolchain. While offering low-latency execution, tightly coupled accelerators are less flexible due to their dependence on the specific CPU design (i.e., [4], [5]).
Coprocessors offer an intermediate solution by remaining externally interfaced with the CPU while optionally incorporating dedicated register files when needed. Unlike tightly coupled accelerators, they provide greater architectural flexibility and scalability for multi-core systems, making them suitable for computationally intensive tasks (i.e., [6]).
With the rise of open-source architectures, **RISC-V** has played a key role in promoting research and development of custom accelerators. The introduction of the Core-V eXtension InterFace (CV-X-IF) [1] further simplifies the integration of tightly coupled accelerators by providing standardized signals, thus reducing the complexity of modifying CPU toolchains and pipelines.

This study investigates different integration strategies for KRONOS, a Keccak RISC-V Optimized eNgine for haShing. Keccak is the core hash function in many post-quantum cryptography (PQC) schemes, where efficient hardware acceleration is crucial for performance optimization. Our analysis compares the trade-offs among the three integration methodologies, demonstrating that while the tightly coupled approach achieves the lowest speed-up, it provides the best *Throughput/Area* efficiency with minimal resource overhead.
The implementation of this work is publicly available on GitHub. [2].

2 PRELIMINARIES

Keccak [1] is the winner of the SHA-3 Cryptographic Hash Algorithm Competition, as it provides superior robustness and security when compared to other hash standards. Its sponge structure is based on the Keccak-f permutation, whose 1600-bit state, defined

[1]https://github.com/openhwgroup/core-v-xif/tree/main
[2]https://github.com/vlsi-lab/KRONOS.git

as the sum of the rate r and capacity c, is composed of a 5×5 matrix of 64-bit words. The permutation consists of 24 rounds of 5 steps each: namely, θ, ρ, π, χ, and ι (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 $0 \leq i < 24$, $0 \leq j < 5$, $0 \leq k < 2$.
(1) for $round = 0$ to $n_r - 1$ do
    (a) $\theta(A)$              // Non-linear mixing operation on the state
    (b) $\rho(A)$                // Diffusion by rotating bits in each lane
    (c) $\pi(A)$                 // Diffusion by rearranging lanes
    (d) $\chi(A)$                // Non-linear mixing operation within lane
    (e) $\iota(A, round)$        // Injects a round constant into the state
```

SHA-3 [8] includes multiple output-length variants: SHA3-224, SHA3-256, SHA3-384, and SHA3-512, which differ in their output sizes and internal rate/capacity configurations. Additionally, the SHA-3 standard defines SHAKE128 and SHAKE256, extendable-output functions (XOFs) that allow variable-length output generation, making them suitable for applications requiring customizable digest sizes. In this work, we specifically evaluate the performance of our accelerator using SHA3-384, which produces a 384-bit hash output. However, since SHA-3 is directly based on Keccak and uses the same permutation function, our accelerator can be used for any SHA-3 variant.

3 DESIGN

In the following, the three versions of KRONOS are presented: loosely coupled, tightly coupled, and coprocessor. All three are integrated into **X-HEEP** (eXtendable Heterogeneous Energy-Efficient Platform)[7], a RISC-V-based microcontroller, albeit in different ways. X-HEEP provides a simple customized MCU (Micro Controller Unit) that can be easily extended with any accelerator by simply instantiating it in the design. This is made possible by employing two different interfaces: the Extendible Accelerator InterFace (XAIF), and the CV-X-IF interface. Therefore, no modifications are required to the system per see. The XAIF interface provides enhanced connectivity to external accelerators and includes slave/master ports, connected to the internal bus using the OBI bus protocol, and interrupt ports, to inform the host CPU at the end of accelerator operations. The CV-X-IF has been successfully utilized and integrated into the CV32E40P core, resulting in the labelled variant known as CV32E40PX [3].

Loosely coupled. The first approach consists of a loosely coupled, memory-mapped component, accelerating the complete Keccak-f permutation (Fig. 1).

Fifty additional 32-bit registers (*KECCAK REG* in Fig. 1) are used to store the state, reducing the load/store operations to and from the memory. The XAIF enables communication between the primary RISC-V CPU core and accelerators. This interface utilizes a

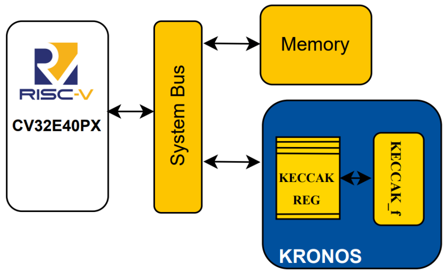

Figure 1: Loosely coupled approach.

dedicated register file for processor control and supports multiple protocols, including APB, AXI-Lite, OBI, and AXI, ensuring broad compatibility across different microcontroller architectures. The necessary registers and their related parameters are defined in *hjson* format, which allows for the automatic generation of SystemVerilog structures used in the communication protocol. System management is handled through dedicated software drivers that coordinate multiple accelerators, overseeing their configuration, execution, and synchronization. During initialization, essential components such as interrupt controllers and DMA are set up to enable smooth interaction between the CPU and the accelerator module and then addressed during the execution of the application. A dedicated driver is used to call KRONOS when needed, which dynamically configures specific bits within the control register file. This structured methodology enables software drivers to efficiently manage communication and coordination between the processor and accelerators, enhancing overall system performance and optimizing resource utilization.

Tigthly coupled. The second approach consists of a tightly coupled accelerator, which leverages the CV-X-IF interface (Fig. 2).

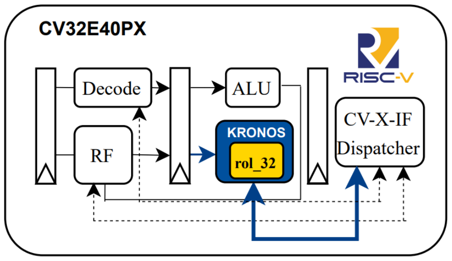

Figure 2: Tigthly coupled approach.

The CV-X-IF controller operates by monitoring the instruction opcode from the core's issue request. When it detects a custom instruction, it triggers a response signal, activating the corresponding accelerator. Once validated, the accelerator samples input data directly from the CPU's register file, ensuring a synchronized data transfer. It then executes the operation and, when the result is ready and the core is prepared to receive it, the output is written back to the register file through the result interface. This streamlined process ensures efficient execution while maintaining tight integration with the CPU. This implementation is fully compliant with

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

RISC-V instruction set extensions (ISE). As a matter of fact, the new instructions introduced adhere to the standard two-source, one-destination register format, ensuring compatibility with the scalar register file. Since Keccak was originally designed for 64-bit architectures, we initially implemented a software-based rol32 function that reconstructs 64-bit rotations using two 32-bit words. This approach, while functional, introduces significant overhead due to multiple shift and XOR operations. To overcome this performance bottleneck, we designed a dedicated RISC-V `rol_32` instruction using inline assembly. This new instruction efficiently executes the 64-bit rotation by operating directly on two 32-bit registers, significantly reducing instruction count and execution latency.

Coprocessor. The third version is a tightly coupled coprocessor (Fig. 3). As in the first case, the complete Keccak-f permutation is performed, but the CV-X-IF facilitates communication with the core.

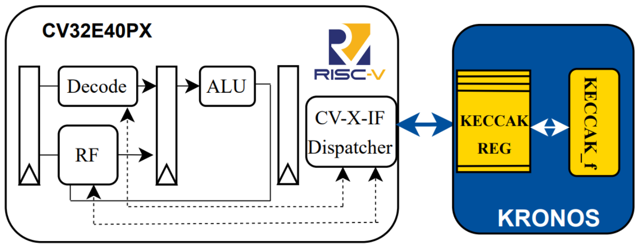

Figure 3: Coprocessor approach.

This 1600-bit register (*KECCAK REG* in Fig. 3) efficiently stores the entire permutation state, enabling streamlined processing and retrieval by the core. To support this functionality, three R-type custom instructions have been added to store/load the state into *KECCAK REG* and to start the 24-round permutation. `load` operation that transfers the state matrix into the Keccak register in 64-bit chunks; the `start` command that triggers the 24-round computation; and a specialized `store` instruction that writes the final output back to the core's register file in 32-bit segments.

4 RESULTS

In this section, we report our implementation results. Tab. 1 illustrates the cycle counts for executing the same function on X-HEEP platform using the three different methods, and their relative speed-ups. The `-O2` flag is used for reference and accelerated simulations. The SHA3-384 function's reference implementation requires 56,529 clock cycles. When accelerated using the loosely coupled method, the cycle count drops to 4,169, achieving a 13.56× speed-up. The tightly coupled method reduces the cycle count to 31,527, resulting in a 1.79× speed-up. Finally, the coprocessor-based approach achieves the lowest cycle count of 7,553, corresponding to a 7.48× speed-up compared to the reference implementation. This analysis highlights the efficiency gains of different acceleration methods, with the loosely coupled approach providing the highest speed-up, while the coprocessor implementation also achieves significant performance improvements with a different trade-off.

Tab. 2 presents a comparison of the FPGA area results for the three versions, implemented on the Zynq UltraScale+ ZCU104 board

Table 1: Cycle counts (SHA3-384).

Method	Reference	Accelerated	Speed-up factor
Loosely	56,529	4,169	13.56×
Tightly		31,527	1.79×
Coproc		7,553	7.48×

(xczu7ev-ffvc1156-2-e). The synthesis and implementation are performed using Xilinx Vivado, with a global clock of 50 MHz. The tightly approach provides the most compact area, as this implementation just accelerates a part of the whole Keccak function and does not comprehend a dedicated register file.

Table 2: Resource utilizations (ZCU104, 50 MHz).

Method	LUT	Registers
Loosely		
⋄ KRONOS	4,915	3,252
∘ Controller	7	35
∘ Keccak	4,908	1,617
∘ Keccak Reg	0	1,600
Tightly		
⋄ KRONOS	569	139
∘ Controller	569	102
∘ rol_32	0	32
Coprocessor		
⋄ KRONOS	6,591	3,372
∘ Controller	1,181	125
∘ Keccak	5,409	1,615
∘ Keccak Reg	0	1,600

A final comparison is summarised in Tab. 3. The last column of Tab. 3 presents the *Throughput/Area* results, allowing for a final comparison among the three approaches. The optimal trade-off between area and throughput is provided by the tightly coupled version. Although it offers the lowest throughput and speed-up among the three cases, it provides a lower number of LUTs with respect to the loosely and coprocessor versions of almost 9 and 11 times, respectively. Additionally, the absence of a dedicated Keccak register file results in a substantially more compact solution when compared to the other integration methodologies.

Table 3: Design comparison (ZCU104 - 50 MHz).

Method	Thr. [Mb/s]	Thr./Area [Mb/(s·LUTs)]
Loosely	4.61	0.937
Tightly	0.61	**1.070**
Coproc	2.54	0.386

5 CONCLUSIONS

This work explores the impact of integration methodologies on accelerator performance, using Keccak as a case study in post-quantum cryptography. By evaluating three versions of KRONOS, we demonstrate the trade-offs between flexibility, latency, and architectural complexity. The CV-X-IF interface proves instrumental in enabling efficient accelerator integration while maintaining RISC-V compliance. Overall, the results underscore the importance of selecting the right integration strategy based on application needs, ensuring optimized performance for cryptographic workloads in post-quantum secure systems.

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] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, and R. Van Keer. 2015. FIPS PUB 202 - SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. (2015). https://keccak.team/hardware.html

[2] Alessandra Dolmeta, Mattia Mirigaldi, Maurizio Martina, and Guido Masera. 2023. Implementation and integration of Keccak accelerator on RISC-V for CRYSTALS-Kyber. In *Proceedings of the 20th ACM International Conference on Computing Frontiers* (Bologna, Italy) *(CF '23)*. Association for Computing Machinery, New York, NY, USA, 381–382. https://doi.org/10.1145/3587135.3591432

[3] Tim Fritzmann, Uzair Sharif, Daniel Müller-Gritschneder, Cezar Reinbrecht, Ulf Schlichtmann, and Johanna Sepulveda. 2019. Towards Reliable and Secure Post-Quantum Co-Processors based on RISC-V. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1148–1153. https://doi.org/10.23919/DATE.2019.8715173

[4] Tim Fritzmann, Georg Sigl, and Johanna Sepúlveda. 2020. RISQ-V: Tightly Coupled RISC-V Accelerators for Post-Quantum Cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020, 4 (Aug. 2020), 239–280. https://doi.org/10.13154/tches.v2020.i4.239-280

[5] Junhao Huang, Alexandre Adomnicăi, Jipeng Zhang, Wangchen Dai, Yao Liu, Ray C. C. Cheung, Çetin Kaya Koç, and Donglong Chen. 2024. Revisiting Keccak and Dilithium Implementations on ARMv7-M. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2024, 2 (Mar. 2024), 1–24. https://doi.org/10.46586/tches.v2024.i2.1-24

[6] Jihye Lee, Whijin Kim, and Ji-Hoon Kim. 2023. A Programmable Crypto-Processor for National Institute of Standards and Technology Post-Quantum Cryptography Standardization Based on the RISC-V Architecture. *Sensors* 23, 23 (2023). https://doi.org/10.3390/s23239408

[7] Simone Machetti, Pasquale Davide Schiavone, Thomas Christoph Müller, Miguel Peón-Quirós, and David Atienza. 2024. X-HEEP: An Open-Source, Configurable and Extendible RISC-V Microcontroller for the Exploration of Ultra-Low-Power Edge Accelerators. arXiv:2401.05548 [cs.AR]

[8] National Institute of Standards and Technology. 2024. SHA-3 Project. https://csrc.nist.gov/projects/hash-functions/sha-3-project Accessed: February 21, 2025.