RISC-V-based Acceleration Strategies for Post-Quantum Cryptography

Ivan Sarno[1]*, Stefano Di Matteo[1,2], Emanuele Valea[1] and Cyrille Chavet[3]

[1]Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France, [2]Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France, [3]Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA, 38000 Grenoble, France

Abstract

In this paper, we explore the main existing approaches to accelerate Post-Quantum Cryptography (PQC) algorithms taking advantages of the flexibility offered by the RISC-V platforms. Tightly-coupled accelerators reside inside the CPU pipeline enabling fast execution of custom instructions. Coprocessors are connected directly to the CPU to execute more complex tasks. Loosely-coupled accelerators are accessed through the bus, offering flexibility and high performance. We show a synthetic comparison among these acceleration strategies for Number Theoretic Transform (NTT) and SHA-3, evaluating performance and area overhead on FPGA.

Introduction

The possibility of the realization in the near future of large scale quantum computers represents a threat to digital communication. This massively employs public key cryptography, which can be easily broken using quantum algorithms. In response, NIST launched a competition to develop cryptographic schemes resistant to quantum attacks, namely *Post-Quantum Cryptography* (PQC). ML-KEM and ML-DSA, both from the family of lattice-based cryptography, have been standardized in 2024. These schemes require substantial computational resources, particularly for the Number Theoretic Transform (NTT), used for polynomial multiplication, and the SHA-3 family of hash algorithms, used for pseudo-randomness generation. The rise of RISC-V as an open and extensible Instruction Set Architecture (ISA) has pushed the growth of an ecosystem where companies and research institutions can freely extend open-source cores and the ISA itself to meet their needs. This flexibility enables new acceleration approaches beyond traditional memory-mapped accelerators, leveraging custom instructions. In this paper, we describe the main existing acceleration approaches for PQC primitives that leverage the flexibility of RISC-V architectures. In addition, we provide a comparison of these approaches relying on performance and FPGA area figures.

Acceleration strategies for PQC

Figure 1 summarizes the main acceleration strategies that can be applied in RISC-V System-on-Chips.

Tightly-coupled accelerators are integrated directly into the RISC-V processor's pipeline as custom functional units. The close integration minimizes communication overhead, allows seamless resource sharing, and simplifies the software-hardware interface. However, they introduce challenges like increased design complexity and scalability issues. This approach is ideal for accelerating memory-intensive tasks while minimizing hardware resource costs, making it well-suited for IoT devices and embedded systems. An example of tightly-coupled acceleration for NTT is reported in [1], where the authors accelerate at coefficient level the NTT operation directly inside the pipeline of the processor. For SHA-3, authors of [2] execute a complete round of Keccak into the pipeline of the core by using floating-point registers to store the state of Keccak.

Loosely-coupled accelerators operate independently of the main processor, connecting via a bus and a memory-mapped interface. They can execute tasks in parallel with the CPU, enabling concurrency between cryptographic and general-purpose operations. This design provides scalability, high throughput, and energy efficiency. However, it also introduces challenges such as communication overhead, resource replication, and memory and synchronization management. Despite these challenges, this approach is well-suited for systems that require both flexibility and high performance. The work in [3] shows an example of loosely-coupled accelerator for NTT. For the SHA-3 algorithm, the work in [4] performs all SHA3/SHAKE set of functions autonomously, also handling padding operations.
Coprocessor-based accelerators are a compromise between loosely and tightly approaches. The coprocessor is directly connected to the core, reducing communication latency and enabling the implementation of custom instructions. On the other side, it has its own pipeline and resources, in this way it can be designed independently of the core, simplifying the integration, and is able to perform complex tasks, like supporting vector instructions. An example is the work in [5], which implements a coprocessor that executes custom instructions for SHA3. An example of a

*Corresponding author: `ivan.sarno@cea.fr`

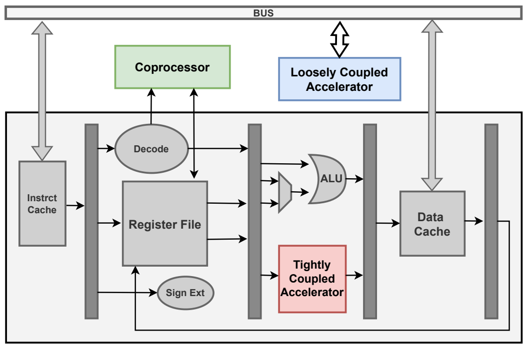

Figure 1: *Acceleration strategies in RISC-V SoCs.*

high-performance coprocessor for NTT is reported in [6], which executes parallel and fast NTT.

Comparison

In Table 1, we compare the implementation results of the identified acceleration strategies for NTT and SHA-3 algorithms, in the framework of PQC. The resource consumption is evaluated on the Artix-7 FPGA. For the software implementations, we considered the PQClean[1] library executed on a 32-bit SoC based on the CV32E40P processor with an OBI interconnect and 256 KB of memory (data + instruction). For the NTT, the tightly-coupled accelerator of [1] provides a gain of around 5x, with a limited resource consumption of only combinational blocks. The loosely-coupled accelerator of [3] provides a huge speed-up (x20); to ensure these performances, the accelerator requires its own memory and four butterfly units, plus a high-speed slave interface to minimize the time spent in I/O. The resource consumption is around 3500 LUTs plus DSPs and BRAMs, which is strongly bigger than the tightly-coupled solution. The work in [6] proposes a high-parallel (32 butterfly units) and high-performance coprocessor for NTT, at a huge cost in terms of resources. This example shows the high level of flexibility of the coprocessor approach. These results suggest that tightly-coupled accelerators for NTT are well suited when small acceleration and small area usage are required. On the contrary, the other approaches allow reaching very high-performance. For what concerns SHA-3 accelerators, the tightly-coupled implementation of SHA-3 of [2] shows an innovative integration approach: it executes a complete round of Keccack inside the pipeline of the core by exploiting floating-point registers to save the state of Keccak. The performance gain is impressive with a limited resource consumption (only combinational logic). Clearly, if the core does not support floating-point, the state of Keccak must be stored in additional logic. Loosely-coupled [4] and coprocessor [5] approaches provide similar results for SHA-3 algorithms: significant clock-cycles reduction (from x30 to x60) and comparable resources usage.

[1] https://github.com/PQClean/PQClean

Table 1: *Performance Comparison of different acceleration strategies on RISC-V SoC for NTT and SHA-3*

Works	Algorithm	Clock cycles	Area
SW only			
PQ-Clean	NTT	20,000	–
PQ-Clean	SHA-3	18,000	–
Tightly-Coupled			
[1]	NTT	4,753	459 LUTs + 2 DSPs
[2]	SHA-3	308	3,847 LUTs
Loosely-Coupled			
[3]	NTT	1022	3651 LUTs + 1430 FFs 6 BRAMs + 4 DSPs
[4]	SHA-3	623	6900 LUTs + 4700 FFs
Co-Processor			
[6]	NTT	32	25674 LUTs + 3137 FFs 6 BRAMs + 64 DSPs
[5]	SHA-3	320	3583 LUTs + 2698 FFs

Conclusions

This abstract presented three different acceleration approaches for PQC on RISC-V platforms. The tightly-coupled approach shows promising results for NTT, offering low area usage and decent acceleration, while also yielding good performance for SHA-3 when the core is significantly modified. Coprocessor and loosely-coupled approaches provide a high level of flexibility. The latter can be more efficient when the processor executes tasks in parallel, with cryptographic operations offloaded to the accelerator.

Acknowledgement

This work received funding from the France 2030 program, managed by the French National Research Agency under grant agreement No. ANR-22-PETQ-0008 PQ-TLS.

References

[1] Miteloudi K. et al. “PQ. V. ALU. E: Post-quantum RISC-V Custom ALU Extensions on Dilithium and Kyber”. In: *International Conference on Smart Card Research and Advanced Applications*. 2023.

[2] Fritzmann T. et al. “RISQ-V: Tightly Coupled RISC-V Accelerators for Post-Quantum Cryptography”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020).

[3] Rafael Carrera Rodriguez et al. *Hardware Implementation and Security Analysis of Local-Masked NTT for CRYSTALS-Kyber*. Cryptology ePrint Archive, Paper 2024/1194. 2024.

[4] Diamante Simone Crescenzo et al. “Hardware Accelerator for FIPS 202 Hash Functions in Post-Quantum Ready SoCs”. In: *2024 IEEE 30th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. 2024.

[5] Zhenjiang Wang et al. “An Instruction Extension Based SHA-3 Algorithm Co-Processor Design Scheme”. In: *2023 3rd International Symposium on Computer Technology and Information Science (ISCTIS)*. 2023.

[6] Yifan Zhao et al. “A high-performance domain-specific processor with matrix extension of RISC-V for module-LWE applications”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 69.7 (2022), pp. 2871–2884.