A Hardware Accelerator for Secure Communications through Post Quantum Cryptography

Ambily Suresh[1], Andrew Wilson[1], Diego Gigena-Ivanovich[2], Manuel Freiberger[1], and Willibald Krenn[1]

[1]Silicon Austria Labs GmbH, Sandgasse 34, A-8010, Graz, Austria
[2]Silicon Austria Labs GmbH, Science Park 4, Altenberger Straße 66c, A-4040, Linz, Austria

Abstract

We present our efforts to develop open source hardware (HW) accelerators in the RISC-V platform, particularly for cryptography and machine learning (ML) applications. With the advancements in quantum computers, we find that the development of post quantum cryptography (PQC) algorithms and their HW implementation is a crucial research topic. This presentation describes the architecture, performance estimates, and demonstration plans for our accelerator for the Classic McEliece cryptosystem. We also discuss our efforts to develop a quantum-safe System-on-Chip (SoC) capable of distributed learning by combining the PQC accelerator with an ML accelerator with potential space applications.

Introduction

The past few years have seen great interest and advancements in the development of open source hardware (HW) designs and tools, particularly in Europe after the European Chips act came into effect. Among its other goals, the act seeks to improve Europe's capacity to design, manufacture, and package advanced chips. Open source processors such as RISC-V cores are critical to achieving this goal, as they can empower the whole community from academic researchers to industrial partners to work towards developing high performance and/or low-power HW solutions. As part of various Chips Joint Undertaking (Chips - JU) - funded projects such as TRISTAN[1] and ISOLDE[2], there is an enhanced effort to develop and verify multiple RISC-V based cores and supporting peripherals, both for low power applications and high performance application class processors. These projects also aim to develop various HW accelerators that can be integrated to the RISC-V ecosystem, targeted for applications such as cryptography, neuromorphic computing, and safety & security. These projects also support the development of the open-source software, firmware, and EDA tools for compilation, validation, and implementation of these designs on FPGA or ASIC platforms.

As part of the ISOLDE project, we have been working on the development of a number of open source HW accelerators – with a special focus on cryptography and neural networks and with the final goal of developing a safe and secure, federated learning system. This is done in parallel with the efforts on the development and implementation for neural networks (NN) capable of processing Earth Observation data from a CubeSat platform [1]. Adapting these artificial intelligence (AI) techniques to onboard computers require great efforts to optimise the HW due to the power and area limitations of any satellite bus. The increased interest in CubeSat and small satellite missions, both for as technology demonstrations and for scientific observation purposes, provides a low cost platform to test these HW and algorithms, albeit with even more stringent constraints on power, area, and development time.

Our current efforts are concentrated on implementing quantum safe, cryptographic algorithms on FPGA and implementing and testing NNs along with a RISC-V core on an ASIC platform [2]. These modules are expected to lead to a System-on-Chip (SoC) implementing acceleration of distributed-learning tasks via Quantum-Safe Cryptography (QSC) as conceptualised in Fig. 1

Post Quantum Cryptography

Post Quantum Cryptography (PQC) involves the development of security algorithms and cryptosystems that are secure against the attacks which exploit the advantages provided to processing performed by quantum computers [3]. Technology advancements in PQC algorithms, both in their HW and SW implementation, are essential to ensuring the security of future satellites

[1] https://tristan-project.eu/
[2] https://isolde-project.eu/

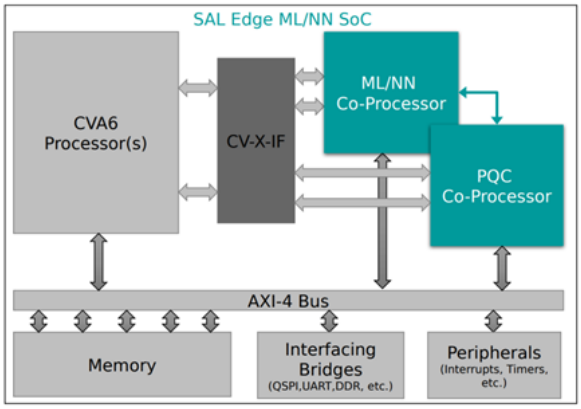

Figure 1: *System architecture of the proposed quantum safe SoC capable of machine learning*

and to ensure long-term data security with rapid advancements in quantum computers. With the 10 – 20 year life time of a large mission, replacing the onboard computers due to possible attacks would lead to astronomical costs. Even smaller satellites demand advance Technology Readiness Levels (TRL) for these PQC implementations to be used onboard, which requires that we invest in ground based demonstrations and testing to enhance the TRL for the coming decades.

PQC algorithms includes the main cryptographic functions such as Public Key Encryption (PKE), Key Exchange, and Digital Signatures. PQC for PKE, which enable secure communications without prior agreement on a secret key, are typically categorised based on their underlying problems: the "Learning with Errors" problem for Lattice-based algorithms and the Syndrome Decoding problem for the Code-based algorithms. Existing non-quantum algorithms rely on mathematical problems which are "assumed hard to solve", such as prime factorisation, discrete logarithms, and elliptic curve solving. Following the development of Shor's and Grover's algorithms, along with the developments in quantum computing over the years, the search computation for these classic PKE problems would receive a significant speed-up. This would make the central mathematical problems more easily solvable (less time-complex), making the existing algorithms no longer secure. Additionally, communication protocols which make use of symmetric key based cryptography, such as Advanced Encryption Standard (AES) and the Secure Hash Algorithm (SHA-2, SHA-3) require significantly larger key sizes in order to maintain security against a quantum-based attack.

We have picked the Classic McEliece (CM) Key Encapsulation Method (KEM) [4], which is a code-based system and a finalist in the National Institute of Standards and Technology (NIST)'s efforts to select and standardise PQC algorithms[3]. We base our overall HW design on previous FPGA based implementations of the CM cryptosystem [5] and primitives, aiming to accelerate the key functions involved at multiple points in the execution of CM. This gives more flexibility in which algorithms and security levels can be implemented, and easier modification of the overall functionality of the system.

Our implementation has selected the Number Theoretic Transform (NTT) as the key primitive to accelerate. This is a PQC operation typically implemented using Fast Fourier Transforms (FFT), which also exploits the convolution theorem and provides a "linearithmic" ($O(n \log n)$ from polynomial $O(n^2)$) time-complexity speed-up to commonly executed polynomial multiplications. One main advantage of accelerating the NTT operation is that it is useful in other non-PQC computations such as error-correction and Homomorphic Encryption (HE), while also providing better performance for the symmetric-key portion of lattice-based communications.

Hardware Architecture

The accelerator is designed in a loosely coupled, memory mapped configuration. We assessed various open-source RISC-V based platforms considering their performance and resource requirements to incorporate into the design. The system uses the Open-HW Group's[4] CVA6 RISC-V core. The overall architecture is based on existing open-source non-PQC-based accelerators and co-processors, such as the PULP platform's vector co-processor, Ara [6]. The core also gives us the option to use the the Core-V-eXtensible-InterFace (CV-X-IF), to transfer any special instructions to the co-processor in the future. It provides native support for the implementation of the AXI-4 bus[5] and has plenty of flexibility in configuration parameters, as a result of which it has received a significant amount of attention from the community.

Fig. 2 gives an outline of our PQC implementation as a memory mapped accelerator via the AXI bus, while optimising the SW to come up with specific instructions for a co-processor implementation via the CV-X-IF interface. One final module is a memory management unit for the larger-keys to boost the efficiency and security in key storage and retrieval. Handling and storage of large-memory keys involved in PQC, is a critical step for providing good Side-Channel Attack (SCA) safety

[3] https://csrc.nist.gov/projects/post-quantum-cryptography/
[4] https://www.openhwgroup.org/
[5] https://github.com/pulp-platform/axi

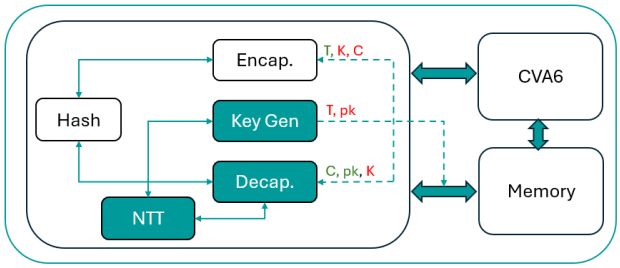

Figure 2: *Architecture of our RISC-V based Classic McEliece PQC system. The modules developed as part of this work are highlighted in dark/turquoise with other open source modules in white. The inputs and outputs from the memory to the PQC modules are marked in green and red respectively. C is the cipher texts, K is the session key, T is the public key, pk is the private key which consists of (δ, c, g, α, s) [4]*

The CM cryptosystem defines three main mathematical functions [4]:

1. Key generation (KEYGEN) – generates the public and private key pairs from random bits
2. Encapsulation (ENCAP) – generates a cipher text and session key from a public key and random bits
3. Decapsulation (DECAP) – outputs a session key when given a cipher text and a private key.

We reuse the basic frame works for the three main modules from the open source implementation in [5], and benchmarked the performance. While the original implementation is parameterised to generate multiple KEMs, we currently target only the mceliece348864 parameter set [4], and are evaluating their performance after incorporating NTT operations.

Firmware and Software

Our SW and firmware design includes a modular compiler that not only optimises the SW to specific architectures but also ensures compatibility across heterogeneous HW. We use the RISC-V tool-chains from the Open-HW group along with the CVA6 core to test the RISC-V core and peripherals, with on-going efforts to introduce and optimise PQC specific instructions in the Instruction Set Architecture (ISA). Open-source projects offer a robust suite of peripherals and interface bridges designed to facilitate the development, simplifying the design and verification process. We plan to exploit the Multi-Level Intermediate Representation (MLIR) framework[7], as it provides a flexible approach to compilation with a gradual lowering of code abstraction to object code, while enabling optimisations at each stage, making it well suited for the heterogeneous landscape of RISC-V HW.

Summary and Future Work

This presentation gives an overview of our Classic McEliece implementation as a HW accelerator based on RISC-V. We are performing functional simulations and optimisations on the overall HW design to determine the range of parameters and coefficients to be accelerated by the NTT module. The complete system including the CVA6 processor will be implemented and tested on an AMD Virtex™ UltraScale™ VCU128 Evaluation Kit. The next milestone would be to optimise the design for low power applications to explore its applications as a quantum-safe SoC for on-board AI.

Acknowledgements

This work has received funding from the ISOLDE project, No. 101112274, supported by the Chips Joint Undertaking of the European Union's Horizon Europe's research and innovation program and its members Austria, Czechia, France, Germany, Italy, Romania, Spain, Sweden, Switzerland.

References

[1] Nicolás Rodríguez, Lothar Ratschbacher, Chunlei Xu, and Pedro Julián. Exploration of deep neural networks with symmetric simplicial layers for on-satellite earth observation processing. In *2022 Argentine Conference on Electronics (CAE)*, pages 31–36, 2022.

[2] Nicolás Rodríguez, Diego Gigena Ivanovich, Martín Villemur, and Pedro Julián. Risc-v based soc platform for neural network acceleration. In *2024 Argentine Conference on Electronics (CAE)*, pages 142–147, 2024.

[3] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors. *Introduction to post-quantum cryptography*, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[4] Martin R. Albrecht et al. *The Classic McEliece Public Key Cryptosystem; Cryptosystem specification, Design rationale, and NIST Round 4 Submission Overview*, Oct. 2022.

[5] Po-Jen Chen et al. Complete and improved fpga implementation of classic mceliece. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, page 71–113, Jun. 2022.

[6] Matteo Perotti, Matheus Cavalcante, Nils Wistoff, Renzo Andri, Lukas Cavigelli, and Luca Benini. A "new ara" for vector computing: An open source

highly efficient risc-v v 1.0 vector processor design. In *2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 43–51, 2022.

[7] Chris Lattner et al. Mlir: Scaling compiler infrastructure for domain specific computation. In *IEEE/ACM International Symposium on Code Generation and Optimization*, pages 2–14, 2021.