Exploring Accelerator Integration with Core-V eXtention InterFace (CV-X-IF) for Kyber

Alessandra Dolmeta, Stefano Di Matteo, Emanuele Valea

Exploring Accelerator Integration with Core-V eXtention InterFace (CV-X-IF) for Kyber

Alessandra Dolmeta[1,2]*, Stefano Di Matteo[2,3] and Emanuele Valea[3]

[1]DET, Politecnico di Torino, Torino, Italy, [2]Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France, [3]Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France

Abstract

This paper presents the integration and optimization of an acceleration module for Number Theoretic Transform (NTT) and Inverse NTT (INTT) algorithms within the CRYSTALS-Kyber cryptographic framework. The accelerator is seamlessly integrated using the CV-X-IF interface, allowing for tight coupling without necessitating modifications to the core. Various optimization strategies are explored, focusing on memory access patterns, compilation flags, and code optimization techniques. The accelerator is rigorously tested to ensure compatibility and performance gains across different optimization configurations. Results demonstrate that the CV-X-IF interface facilitates the integration of cryptographic accelerators with a core, resulting in notable performance enhancements. Moreover, this approach can be applied to any accelerator designed for generic tasks.

Introduction

In this era of advancing technology, 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. On the other hand, tightly coupled accelerators and co-processors reside within the CPU's microarchitecture, necessitating core and toolchain modifications. The emerging *eXtension Interface* (CV-X-IF) [1] allows a seamless support of co-processors. In fact, the CV-X-IF can enhance the CPU with custom or standardized instructions for co-processor support, still without altering the CPU's decode unit. It ensures low-latency and tightly-integrated read and write access to the CPU register file from the co-processor. Unused opcodes within the CPU can be utilized for extensions, although it is advisable not to employ opcodes reserved or used by RISC-V International. Recently, the CV-X-IF has been successfully utilized and integrated into the CV32E40P core, resulting in the labeled variant known as CV32E40PX [2]. In this study, we leverage the CV-X-IF to show the integration of a co-processor specialized in the acceleration of a cryptographic application. Specifically, we target the *Number Theoretic Transform* (NTT) computing kernel, that is present in numerous lattice-based *Post-Quantum Cryptography* (PQC) algorithms, such as CRYSTALS-Kyber. In fact, implementing PQC algorithms on common microcontrollers is a challenge, since many of these algorithms require extensive computational resources, which may exceed the capabilities of embedded platforms [1].

The present work shows that the CV-X-IF is a promising paradigm, that can potentially pave the way to large-scale development of PQC-oriented RISC-V ISA extensions supported by customized co-processors.

Proposed CV-X-IF Integration

Number Theoretic Transfor (NTT) is a mathematical operation that allows to efficiently compute polynomial multiplications and convolutions; it is commonly used in lattice-based cryptographic algorithms such as CRYSTALS-Kyber.

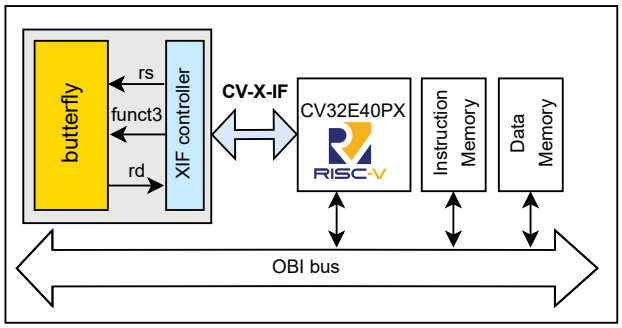

Figure 1: *Architecture of the proposed RISC-V SoC.*

NTT is tipycally executed through butterfly computations which operates on pairs of input elements

*Corresponding author: alessandra.dolmeta@polito.it
[1] https://github.com/openhwgroup/core-v-xif/tree/main
[2] https://github.com/esl-epfl/cv32e40px

plus a fixed coefficient (typically a primitive root of unity, named twiddle factor), and produces pairs of output elements. A butterfly unit for Kyber typically involves modular multiplications and modular additions/subtractions arranged in a specific pattern. In this work, we propose the butterfly unit reported in this work [2], and we connected it to the RISC-V core through the CV-X-IF interface. Figure 1 shows the architecture of the proposed System-on-Chip (SoC). Two kinds of instructions were implemented to integrate the butterfly unit into the NTT computation. The RISC-V instruction formats that are used with the "`.insn`" pseudo directive are: `R`-type (.insn r opcode6, func3, func7,rd, rs1, rs2) and `R4`-type (.insn r4 opcode6, func3, func2, rd, rs1, rs2, rs3). The additional instructions are inserted using the "asm" keyword, which provides precise low-level hardware control, aiding performance optimization and interfacing with hardware. The method initially tests various optimization flags for pure software implementation (`-Os`, `-O2`). Next, both `R`-type and `R4`-type instructions were implemented and evaluated. Since each butterfly operation requires three 16-bit input data (two input coefficients, and the index of the twiddle factor), a concatenation of the two input operands is necessary when using the `R`-type instruction. Conversely, in the `R4`-type instructions with three source registers, this concatenation operation can be circumvented. The method also involves the exploitation of software optimization techniques like loop unrolling.

Results

Looking at the results in Figure 2, it can be observed a progressive improvement in both clock cycles and instructions moving from lower optimization levels to higher ones, and then to the addition of specific instructions and loop unrolling. In the first two cases, the accelerator is not used yet. Moving from `Os` to `O2`, there is a significant improvement in both clock cycles and instructions, since this optimization level focuses more on improving performance while balancing code size compared to `-Os`. The next case introduces `R`-type instructions, resulting in further reductions in both clock cycles (13992) and instructions (12068). This suggests that incorporating specific instructions optimized for the target architecture enhances performance by executing operations more efficiently. When using `R4`-type instructions, there are further reductions in clock cycles (12200) and instructions (10276), because the concatenation and packing of the two inputs operand in one input registers is no longer required. Finally, with the fifth case, where loop unrolling is added, we achieve the lowest number of clock cycles (10255) and instructions (9345). Loop unrolling helps to further optimize the code by reducing loop overhead and enhancing instruction-level parallelism, resulting in the most efficient execution. With fewer loop control instructions, instruction-level parallelism is enhanced, optimizing code execution efficiently.

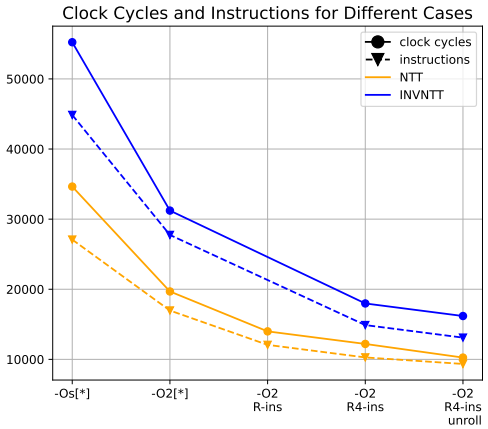

Figure 2: *Integration Results: [*] without accelerator*

It's noteworthy to mention that while the CV-X-IF version utilized in this work predates the latest release as of February 2024, there's a clear intention to adapt and enhance it to align with the latest standards. This commitment ensures that the interface remains up-to-date and coherent with evolving technological advancements, further enhancing its utility and applicability in future cryptographic implementations.

Conclusion

Overall, these results demonstrate how different optimization techniques, including specific instructions and loop unrolling, contribute to the progressive improvement of performance in terms of clock cycles and instructions. CV-X-IF interface proves to be exceptionally valuable and immensely helpful in accelerating cryptographic operations, as demonstrated in this study. The seamless integration of custom instructions, along with the flexibility to optimize code at various levels, showcases the versatility and efficiency of this interface.

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] Duc-Thuan Dam et al. "A Survey of Post-Quantum Cryptography: Start of a New Race". In: *Cryptography* 7.3 (2023).

[2] Pietro Nannipieri et al. "A RISC-V Post Quantum Cryptography Instruction Set Extension for Number Theoretic Transform to Speed-Up CRYSTALS Algorithms". In: *IEEE Access* 9 (2021), pp. 150798–150808.