Improving Post-Quantum Crytography coupling Near-Memory Computing and RISC-V Cores

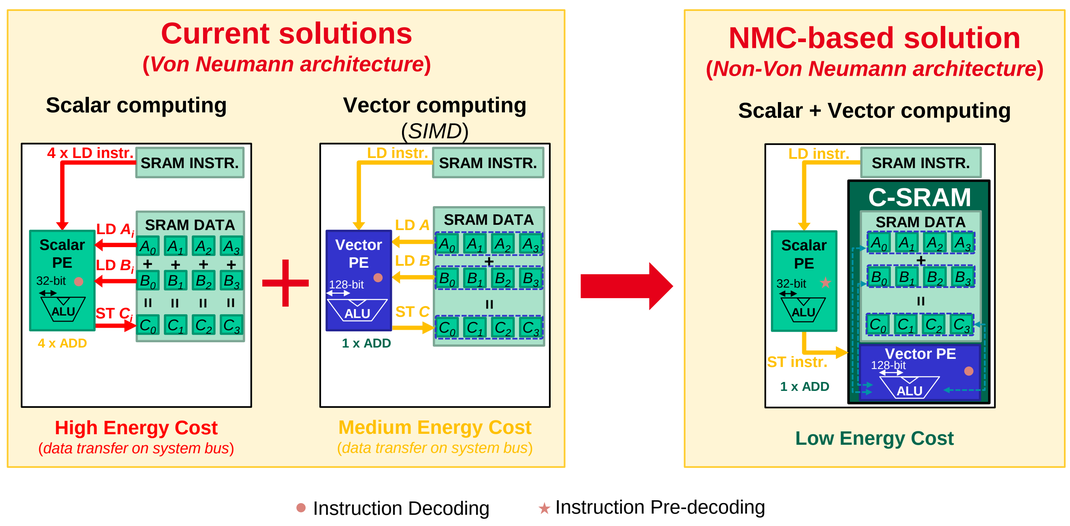
María Ramírez Corrales, Emanuele Valea and Jean-Philippe Noel

Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France
maria.ramirez-corrales@cea.fr

Abstract

Near-Memory Computing (NMC) is a promising architectural approach to accelerate and improve the efficiency of matrix products, since it drastically reduces the transfer of data between the CPU and the data memory. In this poster, we propose to **couple a NMC co-processor with a RISC-V based CPU** to accelerate the matrix product in Post-Quantum Cryptography (PQC) algorithms. Experimental results on the matrix product of FrodoKEM PQC algorithm show a **4x improvement** in performance with respect to the same implementation without NMC approach.

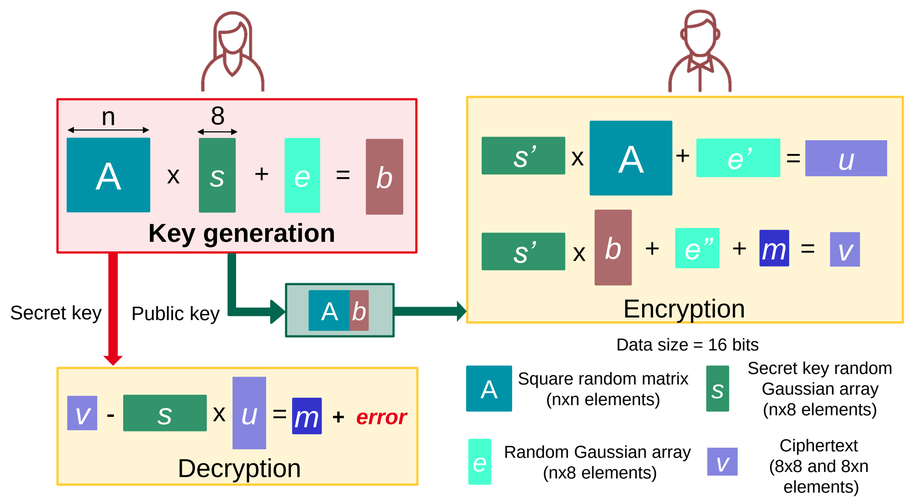
Near-Memory Computing



Thanks to our C-SRAM, we **can compute vectorial instructions near the SRAM array**, using it as a **vectorial co-processor**. This reduces the data transfers between the RISC-V and the memory, what leads to an acceleration on execution time and energy consumption reduction, what makes it perfect for secured embedded applications [1].

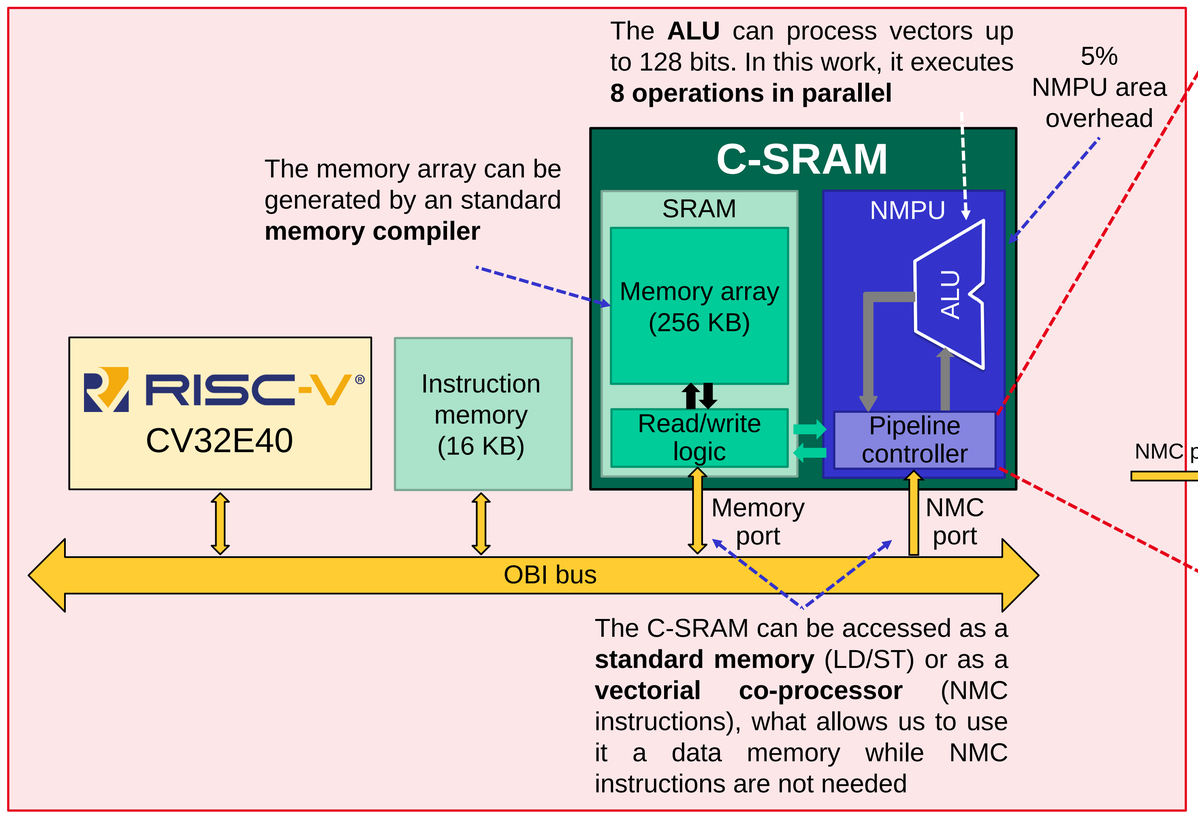
Post-Quantum Cryptography

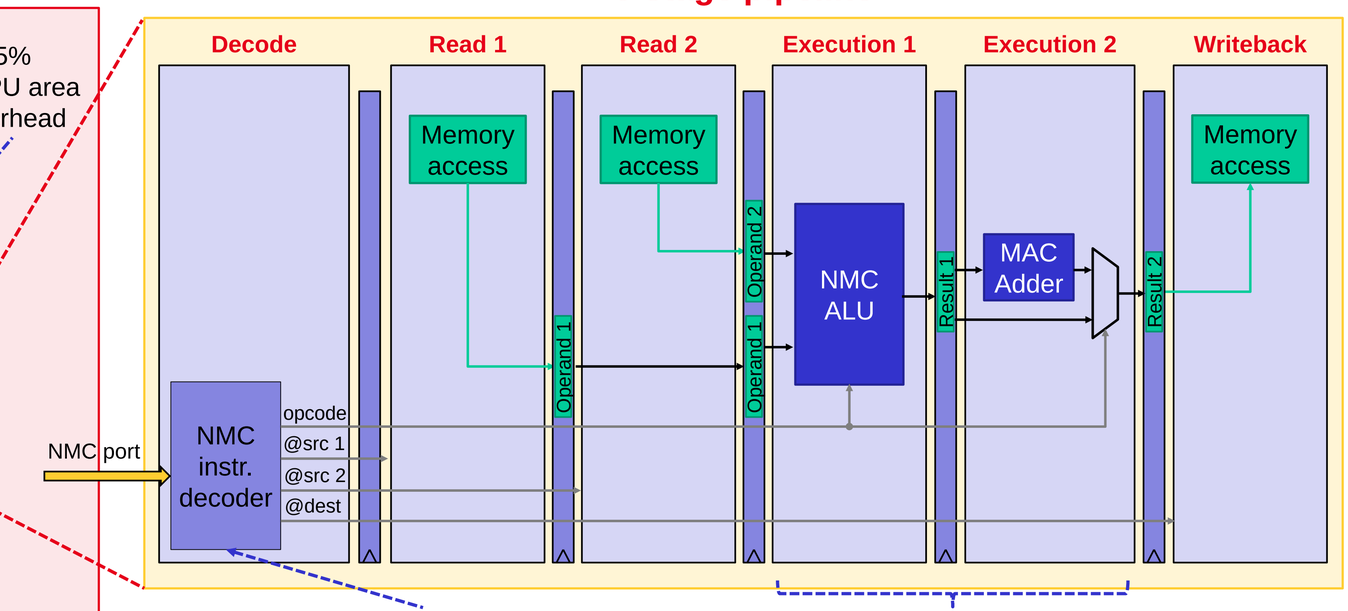
FrodoKEM example



FrodoKEM has three main functions, but in this work we are focused on the **key generation** one as a prove of concept. The matrix product takes half of the execution time and it can be easily parallelized in our C-SRAM.

Our solution



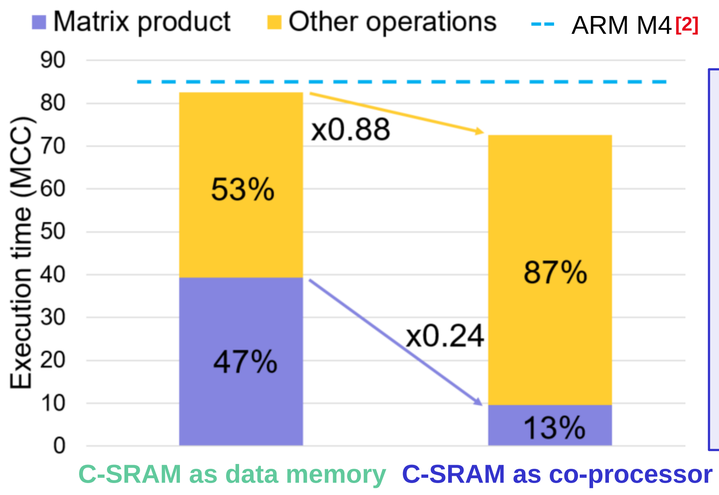


NMC instruction content sent by NMC port and decoded by the NMC decoder in the pipeline controller:

- Opcode
- Source addresses
- Destination address

The only operation executed in two clock cycles is Multiply-Accumulate (MAC)

Results



3% and 15% time reduction compared to the state of the art

12% time reduction on key generation function

4x speed up on matrix product execution

Even if the expected speedup is 8x (8 MACs are executed in parallel), some overhead to compute the 8 partial results in the RISC-V core is added, what leads to just a 4x speed up in matrix product and a total of 12% reduction on total execution time.

We made **two software implementations** of FrodoKEM-640 key generation function:

1. Using C-SRAM as data memory, so the RISC-V core run the whole function
2. Using C-SRAM as a vectorial co-processor for matrix product operation

They were compiled with `riscv32-unknown-elf-gcc` compiler, with `-O1` optimization level

1 MATRIX MULTIPLICATION

The **4x speed up on matrix multiplication** proved the great interest of linking NMC approach and RISC-V processors to accelerate matrix multiplication, not only for PQC applications (IA, image processing, pre-quantum cryptography…).

2 PQC APPLICATIONS

As PQC functions spend half of the execution time on matrix product, with this NMC approach and a RISC-V processor we can accelerate at least half of the function, leading to a **12% reduction on total execution time**.

3 OTHER OPERATIONS

By accelerating the matrix product on the implemented function, the other operations run by RISC-V CPU get more importance (87% of the execution time). We have **to explore** which of them can be accelerated by the NMC co-processor.

References

[1] M. Kooli et al. “Towards a Truly Integrated Vector Processing Unit for Memory-Bound Applications Based on a Cost-Competitive Computational SRAM Design Solution”. In: 18.2 (Apr. 2022).

[2] J. Howe et al. “Standard Lattice-Based Key Encapsulation on Embedded Devices”. In: IACR Transactions on Cryptographic Hardware and Embedded Systems 2018.3 (Aug. 2018), pp. 372–393.