Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
SERICS SECURITY AND RIGHTS IN THE CYBERSPACE

CrypTO CONFERENCE

Politecnico di Torino 1859
Telsy A TIM ENTERPRISE BRAND
DISMA

Politecnico di Torino

Alessandra Dolmeta, PhD student

Exploring Integration Methodologies for Keccak Permutation in RISC-V Architectures

A Practical Overview of Hardware Integration Approaches

Telsy | A TIM ENTERPRISE BRAND

Agenda

01 Why Accelerator for PQC?

Cryptography (PQC), Hardware Devices for PQC, PQC on Embedded/IoT Devices

02 RISC-V

What is RISC-V? How accelerators can be integrated? What are we accelerating?

03 Keccak Accelerators

Design parameters, Loosely coupled, coprocessor and tightly coupled integrations.

04 Integration Methodology

Integration into the X-HEEP System, Loosely-coupled, Coprocessor, and Tightly-coupled Integration.

04 Results

Implementations Results, Conclusions

01

Why Accelerators for PQC?

Post-Quantum Cryptography (PQC), Hardware Devices for PQC, PQC on Embedded/IoT Devices

Telsy | A TIM ENTERPRISE BRAND

Post-Quantum Cryptography

PQC Algorithms and Quantum Threat

- Focuses on algorithms that remain secure against quantum computers.
- Relies on more complex mathematical structures or increased use of hashing
  → results in **heavier computational workloads**

Relevance of Keccak (SHA-3)

- It the sponge-based hash function selected as SHA-3 by NIST.
- It appears in hashing-based signatures and as a building block for other post-quantum protocols.
- It plays a pivotal role in many schemes that must handle significantly larger keys and/or repeated hashing operations.

Telsy | A TIM ENTERPRISE BRAND

Hardware Devices for PQC

Why This Matters for PQC

- It involve more computationally intensive operations than classical crypto.
- The choice of hardware, whether a **specialized accelerator** or a **general-purpose CPU**, directly impacts overall system performance, energy consumption, and feasibility of deploying these algorithms in different contexts.

General-purpose CPUs offer
- Flexibility
- An established software ecosystem,
- **BUT** they can't always meet the performance, or efficiency demands of post-quantum workloads.

Offloading the most demanding cryptographic tasks to a dedicated accelerator, we can:
- boost throughput
- reduce energy consumption

At the cost of additional design effort and hardware resources.

PQC on Embedded/IoT Devices

Specialized hardware accelerators can address PQC bottlenecks more efficiently than general-purpose CPUs alone, especially under strict performance or power constraints.

Performance Bottlenecks

Hashing (e.g., Keccak) and other core operations (like polynomial arithmetic for lattice-based crypto) can be **computationally heavy** if implemented purely in software on a generic CPU. This leads to higher latency and potential throughput limitations, especially for repeated or batched operations.

Area & Power Constraints

Dedicated accelerators can be designed to optimize exactly the operations required by a particular PQC scheme. This fine-tuned approach often uses fewer transistors overall than a CPU handling the same workload, which can reduce both area and power draw, savings battery life and minimizing chip size.

Security

By embedding private keys in on-chip secure enclaves with tamper-resistant design and side-channel protections, custom hardware ensures cryptographic secrets remain isolated and resilient against invasive attacks

Telsy | A TIM ENTERPRISE BRAND

02

RISC-V

What is RISC-V? How accelerators can be integrated? What are we accelerating?

Telsy | A TIM ENTERPRISE BRAND

What is RISC-V?

The Foundation for Flexible Crypto Acceleration.

01

RISC-V is an open-standard Instruction Set Architecture (ISA)

Unlike proprietary ISAs (e.g., x86, ARM), RISC-V is free, modular, and extensible.

Anyone can build, modify, or extend it — no license fees, no vendor lock-in.

02

Designed for Customization

RISC-V offers a base instruction set and allows you to plug in custom extensions for specific domains like cryptography, AI, DSP, etc.

This enables designers to tailor the architecture to application-specific requirements.

03

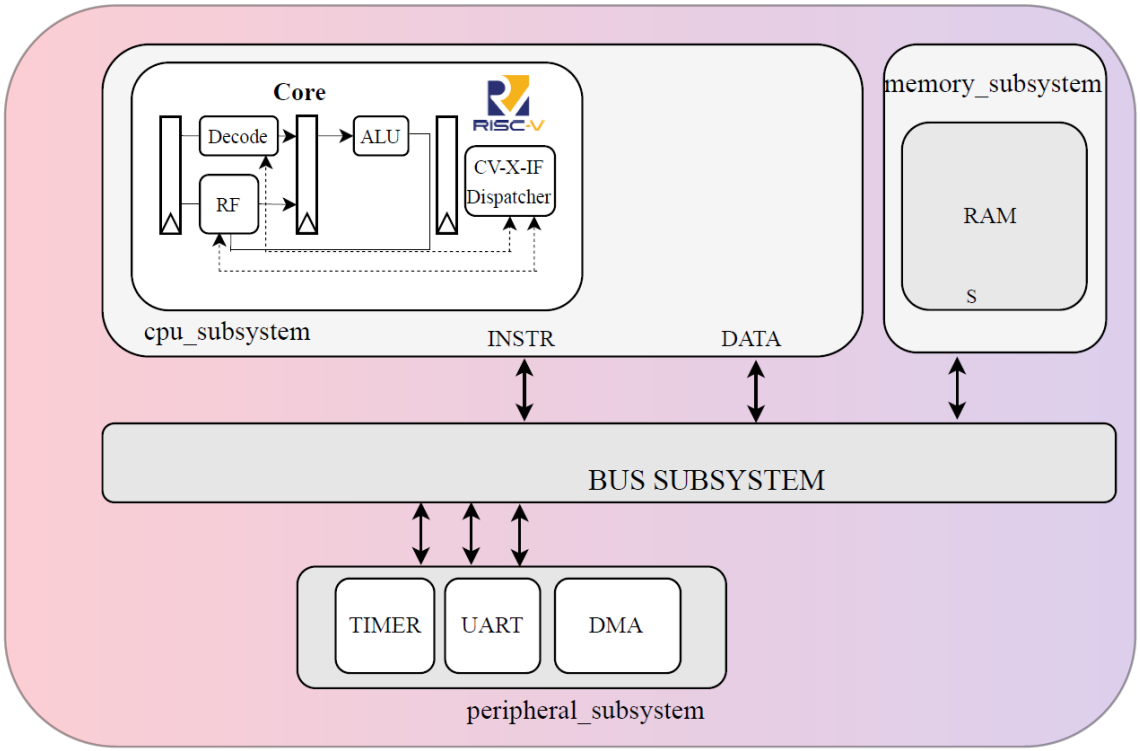
Ideal for Crypto Acceleration

Cryptographic workloads, such as Keccak (SHA-3), benefit from tight hardware/software co-design.

RISC-V's openness makes it easier to integrate hardware accelerators or custom instructions to speed up these tasks, with minimal friction in toolchains.

Telsy A TIM ENTERPRISE BRAND

How accelerators can be integrated?



SoC

A System on Chip is a single-chip system integrating a RISC-V core with on-chip memories, interconnect and peripherals

BUS

- High-speed on-chip interconnect linking the RISC-V core, DMA engine, memories and peripherals

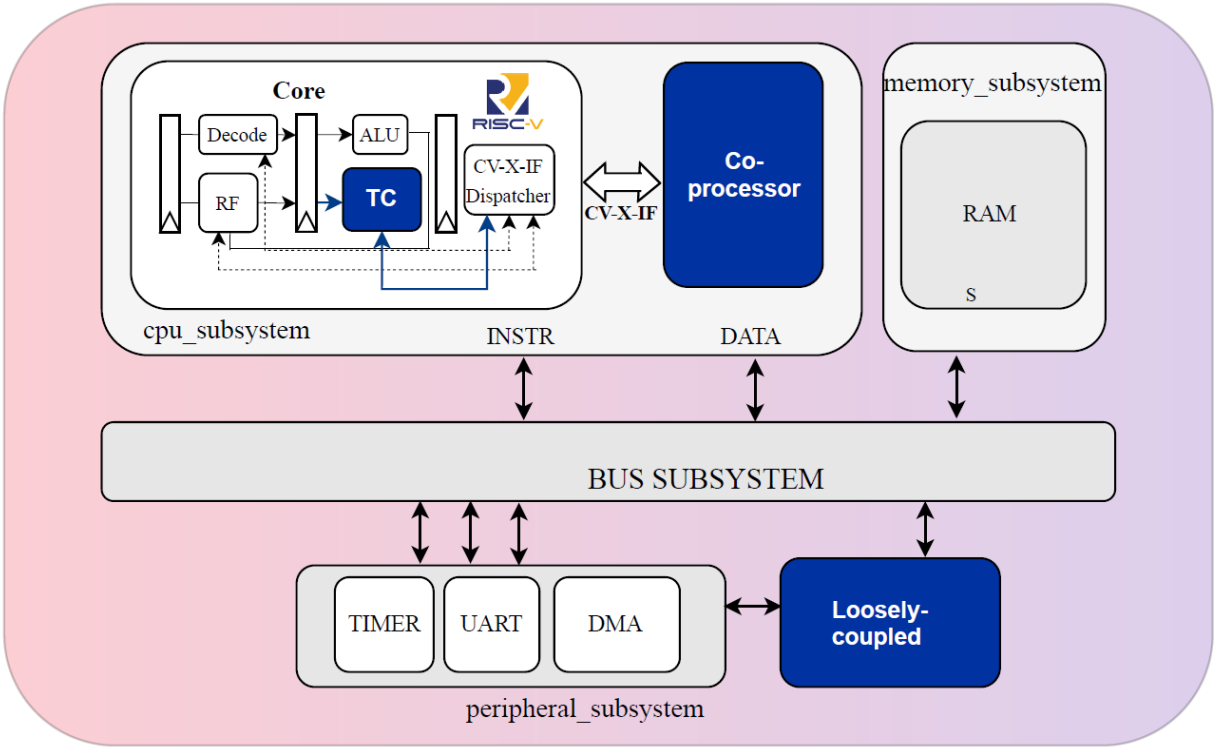
DMA

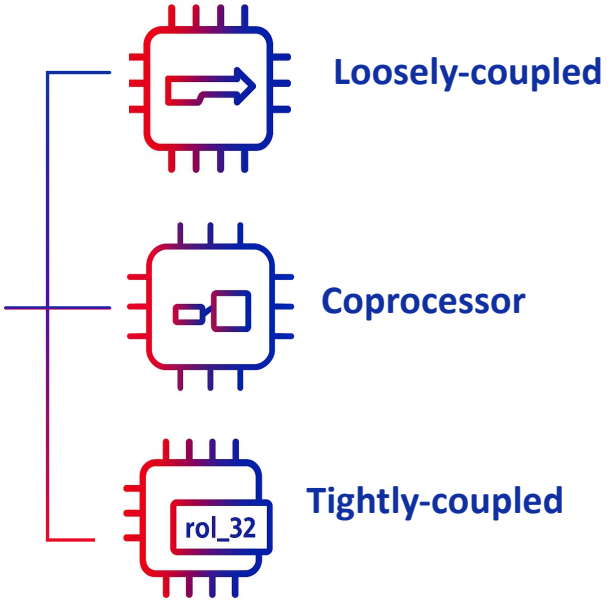
- Dedicated controller that moves data between memory and peripherals without CPU intervention

Memory On-chip

- RAM/ROM storing code and data for the RISC-V core to execute

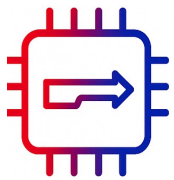
How accelerators can be integrated?

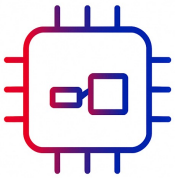




What are we accelerating?

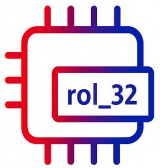
```c
#define NROUNDS 24
#define ROL(a, offset)  (((a) << (offset)) ^ ((a) >> (64 - (offset))))
static const uint64_t RC[NROUNDS] = { /* 24 constants */ };

void KeccakF1600_StatePermute(uint64_t *A) {
    for (int rnd = 0; rnd < NROUNDS; rnd++) {
        // θ step
        uint64_t C[5], D[5];
        for (int x = 0; x < 5; x++)
            C[x] = A[x] ^ A[x + 5] ^ A[x + 10] ^ A[x + 15] ^ A[x + 20];
        for (int x = 0; x < 5; x++)
            D[x] = C[(x + 4) % 5] ^ ROL(C[(x + 1) % 5], 1);
        for (int i = 0; i < 25; i++)
            A[i] ^= D[i % 5];
        // ρ + π + χ + ι steps (simplified view)
        // Each A[i] is rotated (ROL) and permuted in position
        // Chi + round constant applied at the end
        A[0] ^= RC[rnd];  // Example Iota
    }
}
```





Loosely-coupled & Coprocessor

They both accelerate the whole **KeccakF1600_StatePermute** function.



Tightly-coupled

It accelerate a smaller part of KeccakF1600_StatePermute function, which is **ROL**.

03

Keccak Accelerators

Design parameters, Loosely coupled, coprocessor and tightly coupled integrations.

Telsy | A TIM ENTERPRISE BRAND

Design Parameters

A **clock cycle** is the basic time unit in a synchronous design (the interval between two successive rising (or falling) clock edges) during which state updates occur.

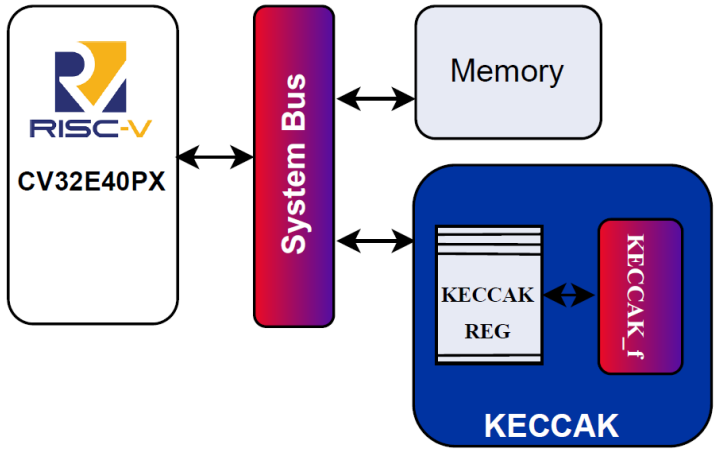
The clock period T_{clk} is the duration of one cycle, so the clock frequency

$$f_{clk} = \frac{1}{T_{clk}}$$

- **Latency**: $total\ time\ per\ operation\ =\ CC \times T_{clk}$
- **Throughput**: $operations\ per\ second\ = \#bit/(CC \times T_{clk})$
- **FPGA Area**: measured in LUTs, FFs, DSPs consumed by the design.

Loosely-coupled integration

Memory-Mapped Accelerator.



Location

Outside the CPU core. Connected via a system bus (e.g., AXI).

Interaction

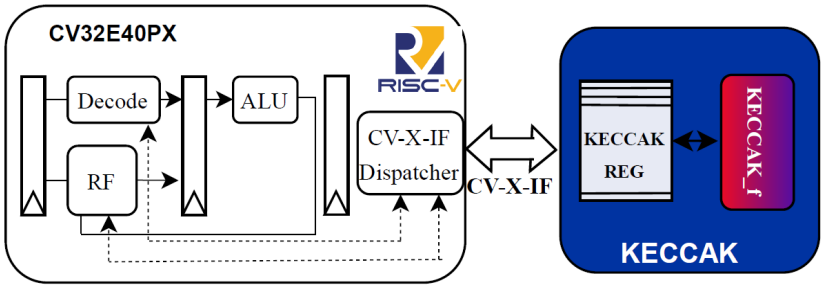
- The CPU must explicitly send data and wait for the result.
- Acts like a peripheral device (e.g., sending data to a GPU or external module).

Implications

- High communication overhead (load/store latency).
- Easiest to implement (no changes to CPU internals).
- Best for reusability and modularity, but not optimal for performance.

Coprocessor integration

CV-X-IF Attached External Unit.



Location

External, but directly connected to the CPU via CV-X-IF.

Interaction

- Special custom instructions to send/load/store data to/from the accelerator.
- Has its own register file (KECCAK_REG) and executes whole Keccak-f operations.

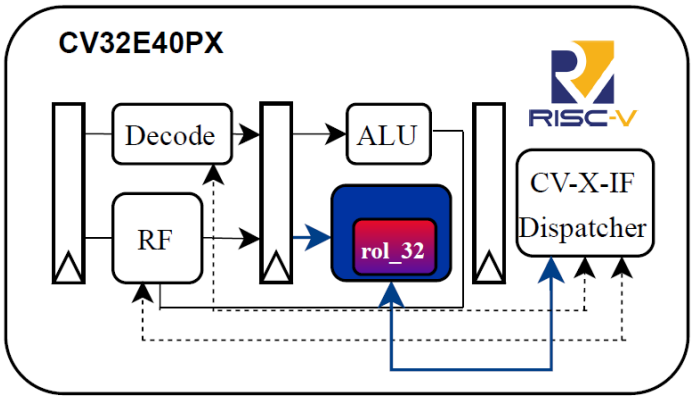
Implications

- Offers a balance between independence and performance.
- Lower latency than memory-mapped.
- Slightly higher area due to the external register file.

Telsy A TIM ENTERPRISE BRAND

Tightly-coupled integration

In-Pipeline Custom Instructions.



Location

Inside the CPU's datapath, integrated into the ALU pipeline via CV-X-IF.

Interaction

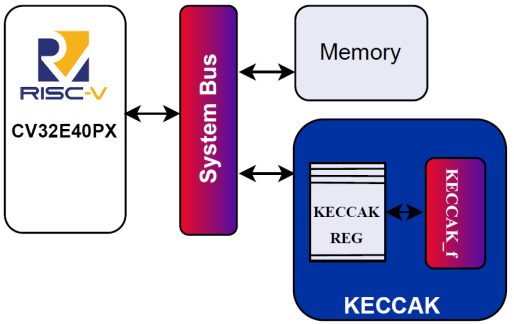
- The accelerator (e.g., rol_32) acts as a custom instruction.
- Direct access to internal registers (RF) and ALU.

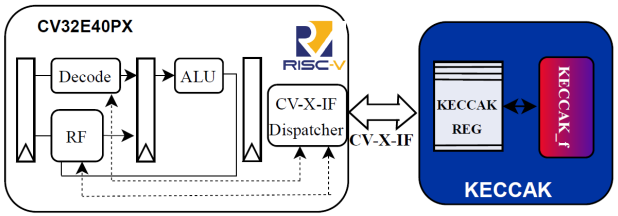
Implications

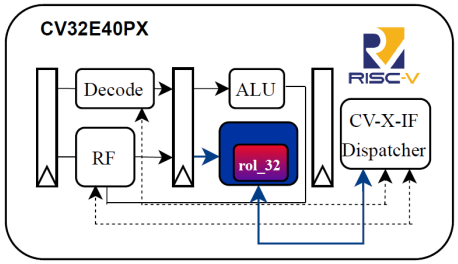
- Lowest latency, as it's like executing a normal instruction.
- Requires modifying the pipeline and supporting custom opcodes.
- Very efficient in area and performance per instruction.

Summary Comparison

Approach	Integration	Latency	Modularity	Area
Loosely-coupled	Memory-mapped	High	High	High
Coprocessor	CV-X-IF	Medium	Medium	High
Tightly-coupled	In-pipeline ALU	Low	Low	Lowest







04

Integration Methodology

Integration into the X-HEEP System, Loosely-coupled, Coprocessor, and Tightly-coupled Integrations

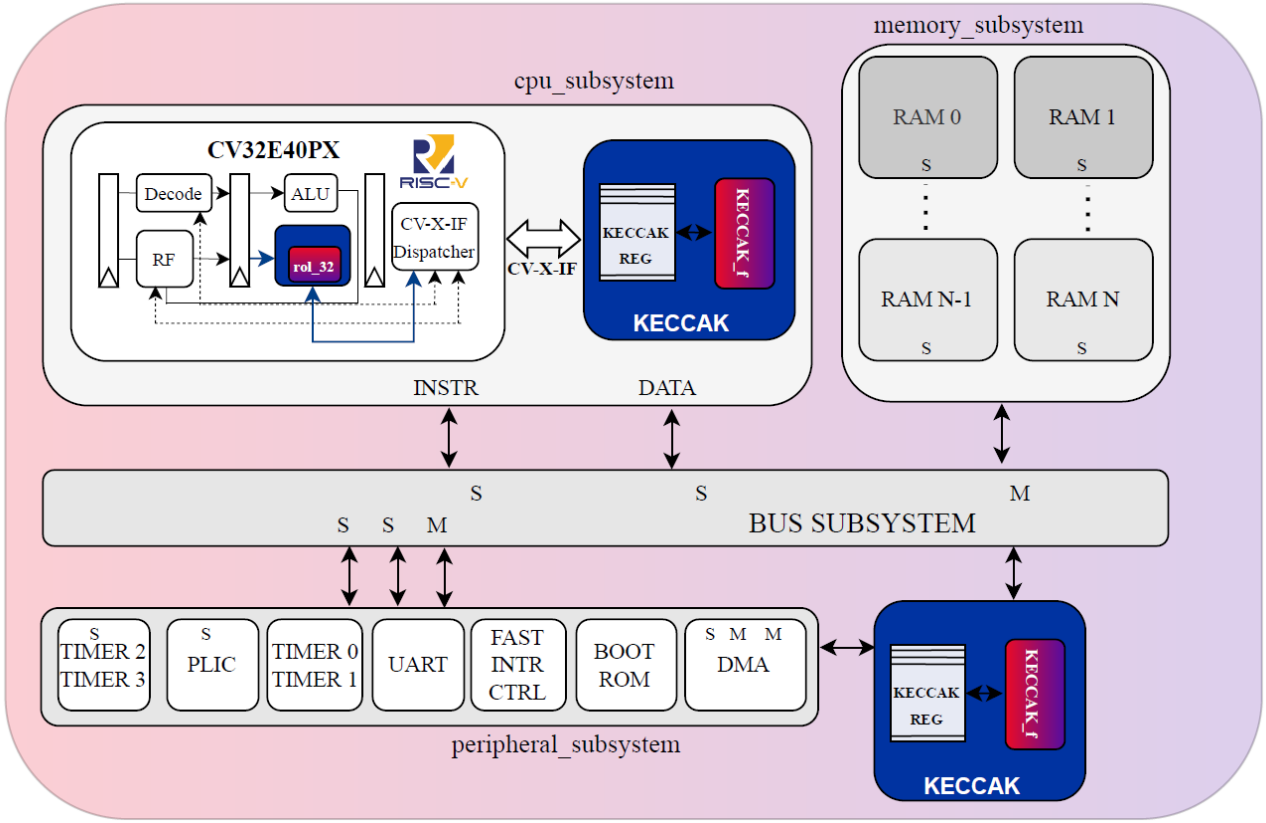
Telsy | A TIM ENTERPRISE BRAND

Integration into the X-HEEP System

System Overview.

The three variants (*representing the different integration strategies*) are evaluated within a realistic SoC environment.

- **Target Platform**: X-HEEP [1], an open-source microcontroller built around RISC-V.
- **Core Used**: All variants are integrated with the CV32E40PX core [2], a RISC-V processor with support for the CV-X-IF interface [3].
- **System-Level Evaluation**: This allows comparison under consistent conditions (*same CPU, memory, bus, and peripheral architecture*).



Loosely-coupled Integration

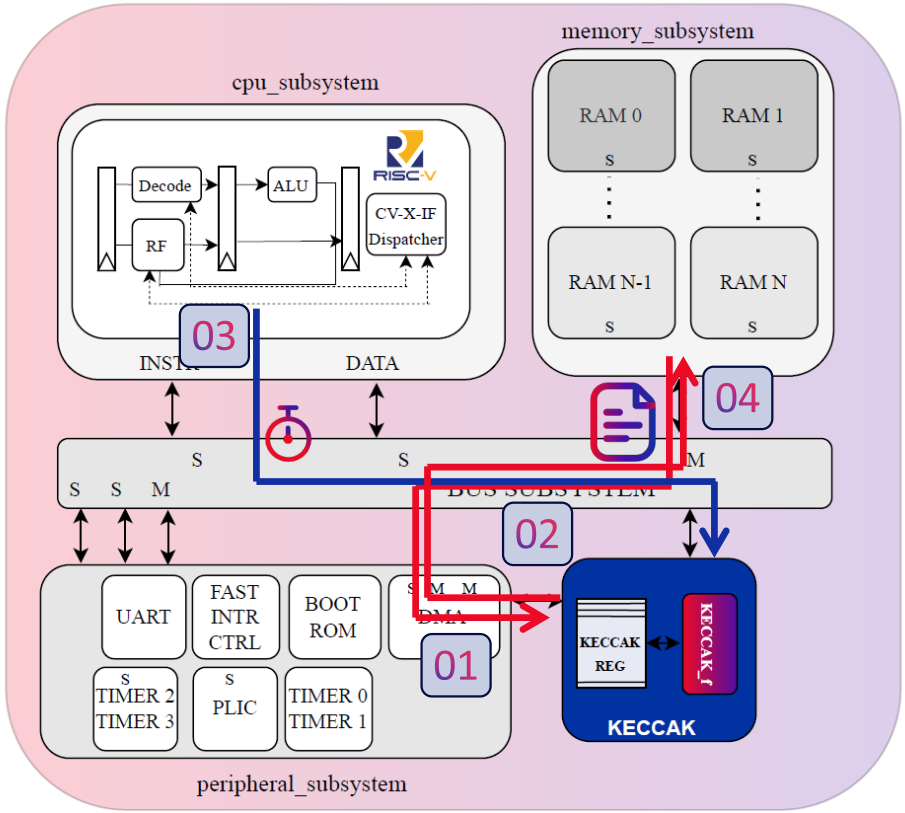
01 **DMA Setup.**
The CPU initializes a Direct Memory Access (DMA) transaction to begin the Keccak permutation. It specifies:
- Source address in RAM (where the input state is stored)
- Destination: the KECCAK register block (KECCAK_REG)
- Length of the transfer (typically 25 × 64-bit words).

02 **State Transfer to KECCAK.**
Once the DMA is triggered:
- The state is automatically fetched from memory,
- And written directly into the KECCAK_REG without CPU intervention. This decouples the CPU from the data movement, saving cycles.

03 **Start the Permutation.**
With the state in place, the CPU sends a control command to the KECCAK module to start the Keccak-f permutation. This is done via a memory-mapped register write (like writing to a command register).

04 **Write Back the Output.**
Once the computation is done:
- The DMA writes the resulting state back to memory,
- Making it available for the CPU or subsequent operations.

The CPU can either poll a “done” register or use an interrupt to be notified.

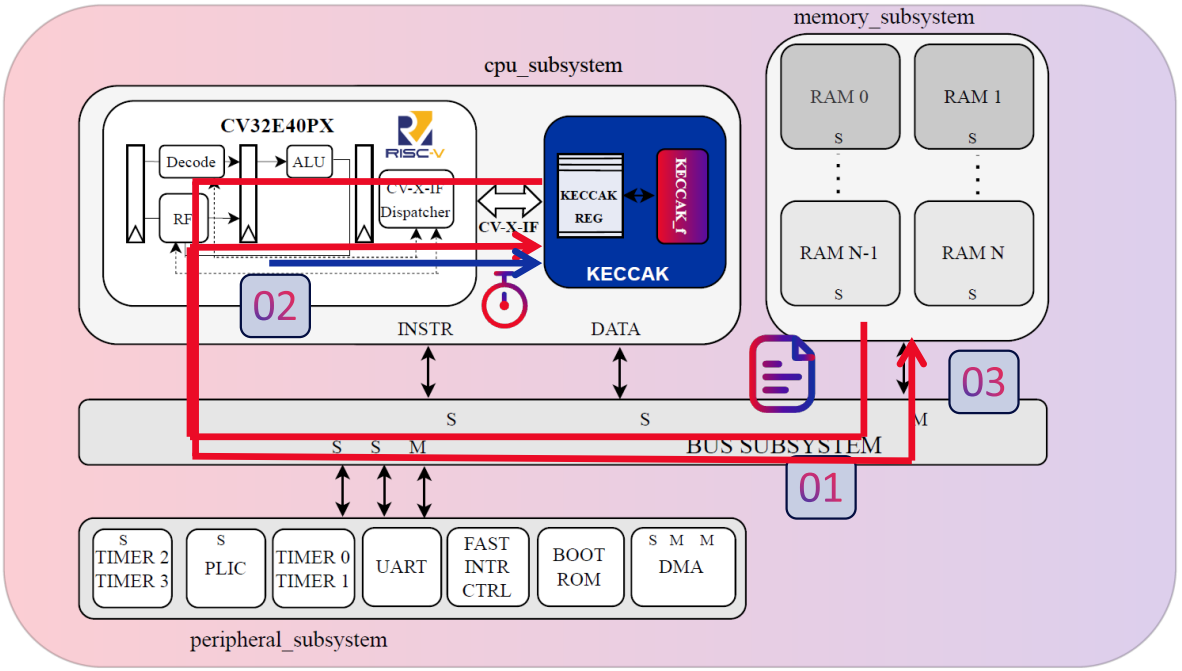


Coprocessor Integration

01 **Load the Keccak State.**
The CPU uses a custom load instruction to move the input state into the KECCAK register block. This instruction transfers the state one 32-bit word at a time from the core register file into KECCAK_REG via the CV-X-IF interface.
These instructions are handled by the CV32E40PX core, extended to dispatch the data to the coprocessor.

02 **Start the Permutation.**
Once the full state is loaded, the CPU triggers a custom start instruction. This command begins the 24-round Keccak-f permutation inside the accelerator.
The computation happens autonomously, without further CPU interaction. The accelerator remains tightly integrated within the CPU subsystem, though not in the pipeline.

03 **Retrieve the Result.**
After completion, the CPU issues a custom store instruction to fetch the result.
The KECCAK module returns the result 32 bits at a time to the CPU register file.



Telsy A TIM ENTERPRISE BRAND

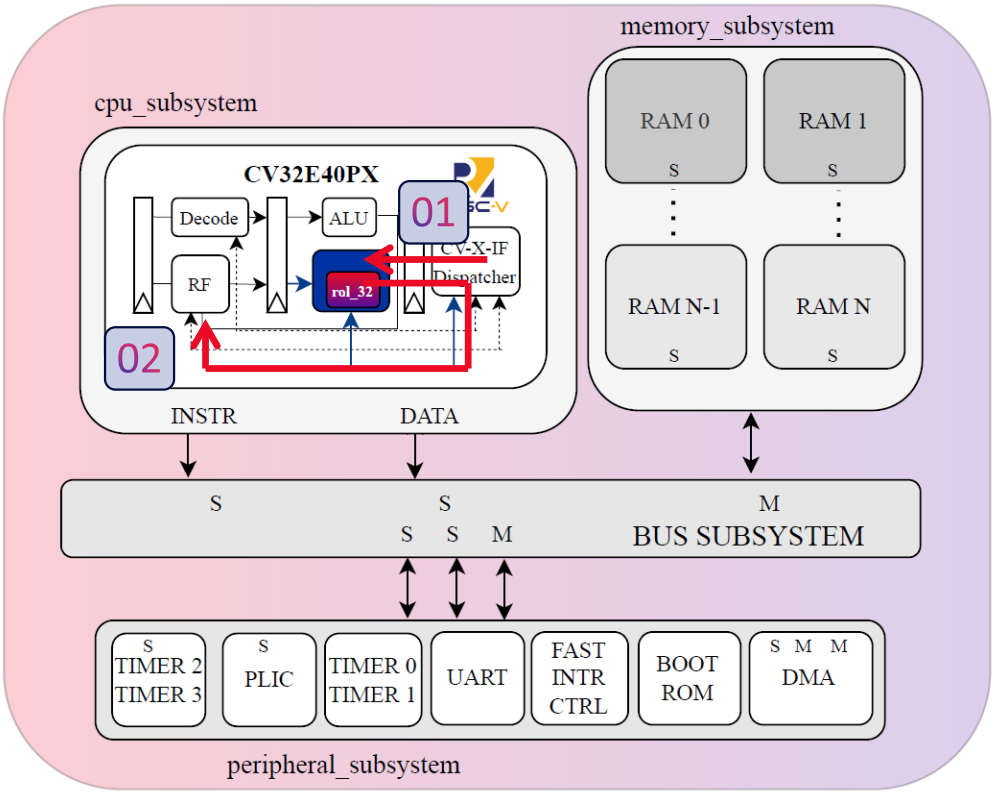
Tightly-coupled Integration

The CPU executes Keccak normally in software, but every time the **ROL** operation is required, it uses the custom **rol_32** instruction.
This is a pipeline-integrated instruction, executed like any ALU operation.

01 **Load the two 32-bit operands.**
The rol_32 instruction takes its two 32-bit operands directly from the CPU's register file (RF).
Since Keccak is 64-bit but the core is 32-bit, two registers are used per operand.

Custom Logic Execution.
The rol_32 instruction performs the 64-bit rotation across the two 32-bit values, using dedicated logic embedded in the ALU stage.
It operates in a single or very few clock cycles, thanks to tight coupling.

02 **Result Storage.**
The result of the rotation is written back to the scalar register file in the following cycle(s), again using standard instruction flow.



Telsy A TIM ENTERPRISE BRAND

05

Results

Telsy | A TIM ENTERPRISE BRAND

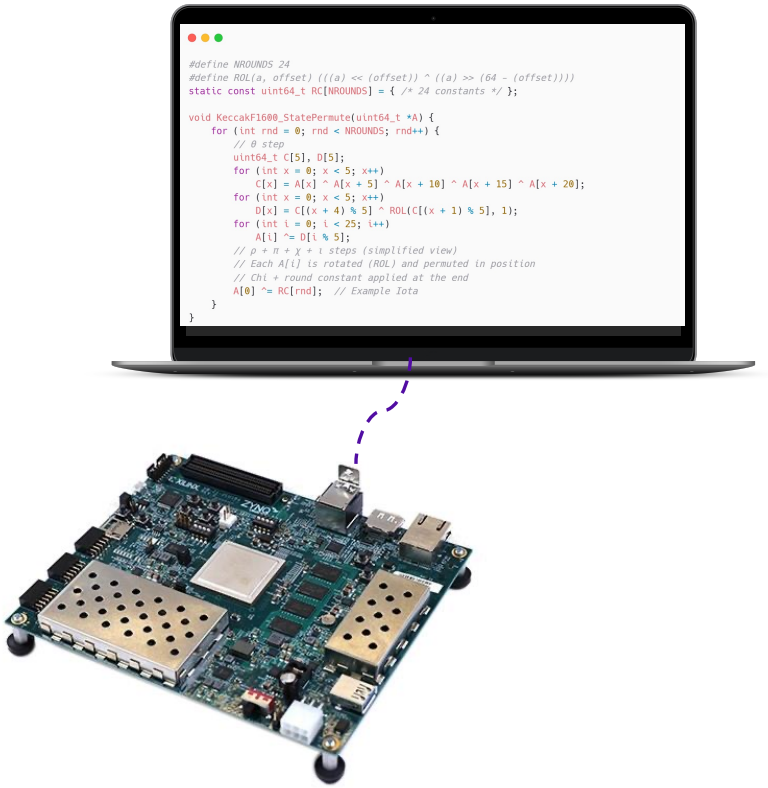
Implementations Results

We implemented on the **Zynq UltraScale+ ZCU104 board**.

The synthesis and implementation are performed using Xilinx Vivado, with a global clock of 50 MHz.

FPGA Performances

Method	Reference [Clock cycles]	Accelerated [Clock Cycles]	Speed-up	Throughput [Mb/s]
Loosely	56,529	4,169	13.56×	4.61
Coprocessor		7,533	7.48×	2.56
Tightly		31,527	1.79×	0.61



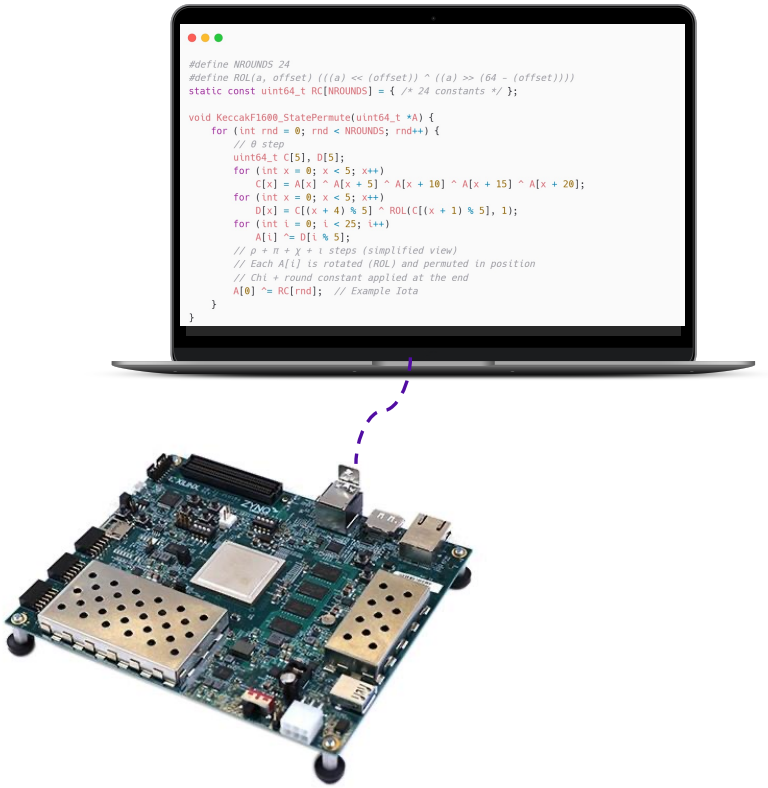
Implementations Results

We implemented on the **Zynq UltraScale+ ZCU104 board.**

The synthesis and implementation are performed using Xilinx Vivado, with a global clock of 50 MHz.

FPGA Area

Method	LUT	Register
Loosely	4,915	3,252
Coprocessor	6,591	3,372
Tightly	569	129



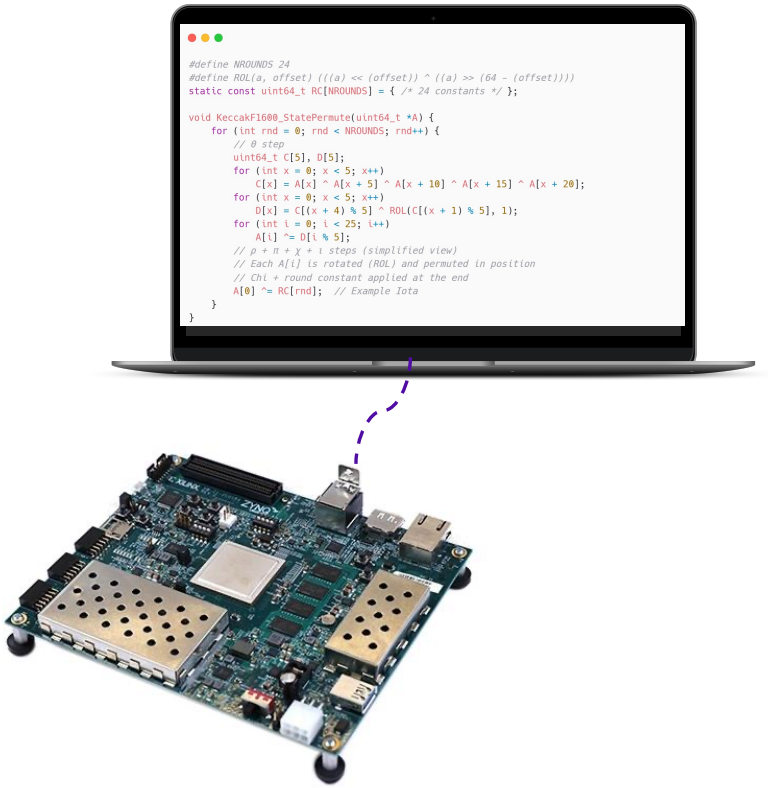
Telsy A TIM ENTERPRISE BRAND

Implementations Results

Method	LUT	Throughput [Mb/s]	Throughput/Area [kb/(s·LUTs)]
Loosely	4,915	4.61	0.937
Coprocessor	6,591	2.56	0.386
Tightly	569	0.61	1.070

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, respectively, almost 9 and 11×.



Telsy A TIM ENTERPRISE BRAND

Conclusions

Design Trade-Offs Matter

- The choice of the integration methodology should be guided by system constraints:
  - Area
  - Performance
  - Power consumption
  - Design complexity

  Each approach brings a different balance of modularity, latency, and throughput.
- There is no one-size-fits-all strategy — the architecture must be tailored to the target application and environment.

Accelerating PQC is Feasible and Flexible

- Post-Quantum Cryptography, while often computationally intensive, can be efficiently accelerated in embedded and general-purpose systems.
- RISC-V's openness and extensibility allow seamless integration of PQC accelerators at various levels of coupling.
- Whether targeting IoT devices, edge platforms, or secure microcontrollers, hardware support can make PQC practical today — not just in the future.

Telsy | A TIM ENTERPRISE BRAND

References

[1] Pasquale Davide Schiavone, Simone Machetti, Miguel Peón-Quirós, Jose Miranda, Benoît Denkinger, Thomas Christoph Müller, Ruben Rodríguez, Saverio Nasturzio, and David Atienza Alonso. 2023. X-HEEP: An Open-Source, Configurable and Extendible RISC-V Microcontroller. In Proceedings of the 20th ACM International Conference on Computing Frontiers (CF '23). Association for Computing Machinery, New York, NY, USA, 379–380. https://doi.org/10.1145/3587135.3591431

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

[3] https://docs.openhwgroup.org/projects/openhw-group-core-v-xif/en/latest/intro.html

Thanks for the attention!


Alessandra Dolmeta


alessandra.dolmeta@polito.it


Crypto Conference 23 / 5 / 25

Telsy A TIM ENTERPRISE BRAND

GruppoTIM.it