A Deep Dive into Integration Methodologies in RISC-V

Alessandra Dolmeta[1], Valeria Piscopo[1], Maurizio Martina[1], Guido Masera[1]

1 Politecnico di Torino, DET - Dipartimento di Elettronica e Telecomunicazioni , Turin, Italy

Finanziato dall'Unione europea

Introduction

- In embedded systems design, the growing computational demands have driven the integration of specialized accelerators. The integration methodology can significantly affect their performance:
  - **Loosely coupled** accelerators are generally memory-mapped and operate independently, interfacing with the CPU via a system bus.
  - **Tightly coupled** accelerators are integrated within the CPU microarchitecture and directly access internal registers.
  - **Coprocessors** differ by utilizing external register files.

- RISC-V's open-source ecosystem has further advanced accelerator research. The Core-V eXtension InterFace (**CV-X-IF**) [1] facilitates tightly coupled accelerator integration without modifying the CPU toolchain and pipeline.
- The analysis of different integration methodologies is applied to Keccak, a pivotal hashing standard in Post-Quantum Cryptography (PQC): we present **KRONOS**, a **Keccak RISC-V Optimized eNgine for haShing**.

CV-X-IF

- The **CV-X-IF interface** introduces a **dispatcher** that enables the integration of custom instructions **without modifying the existing core components**, preserving the original decoder, ALU, and RF (*Figure 1*).

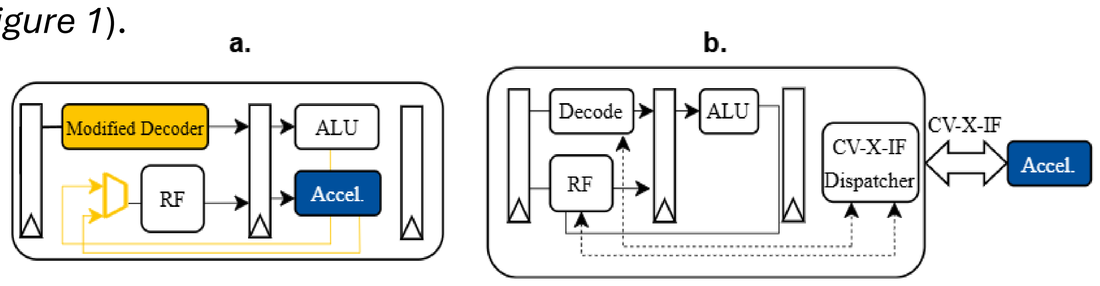

Figure 1. *Common tightly coupled approach (a) vs CV-X-IF (b).*

Integration methodologies

- Three versions of KRONOS are presented. They are integrated into X-HEEP [2], a RISC-V based microcontroller, using the CV32E40PX core [3].

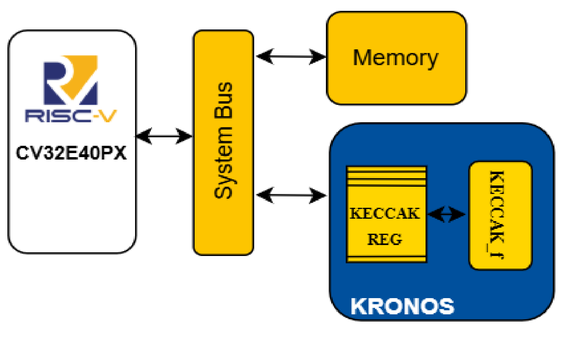

Figure 2. *Loosely coupled approach.*

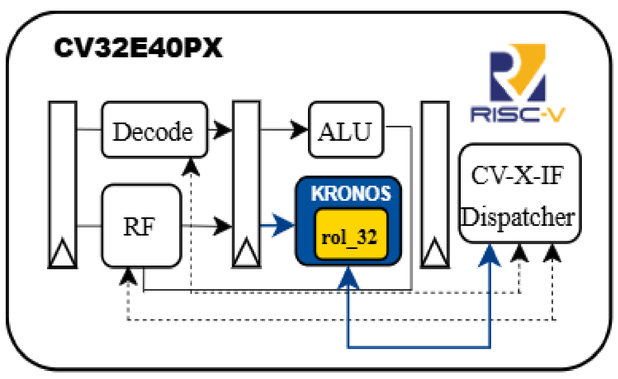

Figure 3. *TIghtly coupled approach.*

- **Loosely coupled** *(Figure 2)*:
  - Memory-mapped component, accelerating the complete Keccak-*f* permutation.
  - Fifty 32-bit registers to store the state.
  - KRONOS called by a dedicated driver.

- **Tightly coupled** *(Figure 3)*:
  - Leverages the CV-X-IF.
  - RISC-V-compliant custom instruction *rol_32*, accelerating the bitwise rotation operation. Instruction called through inline assembly.
  - No additional register file is used.

- **Coprocessor** *(Figure 4)*:
  - Hybrid between loosely and tightly approaches: leverages CV-X-IF but accelerates the complete Keccak-*f* permutation.
  - Three R-type custom instructions to store/load the state into the dedicated Keccak register file and to start the permutation. Instructions called via inline assembly.
  - Not fully RISC-V compliant.

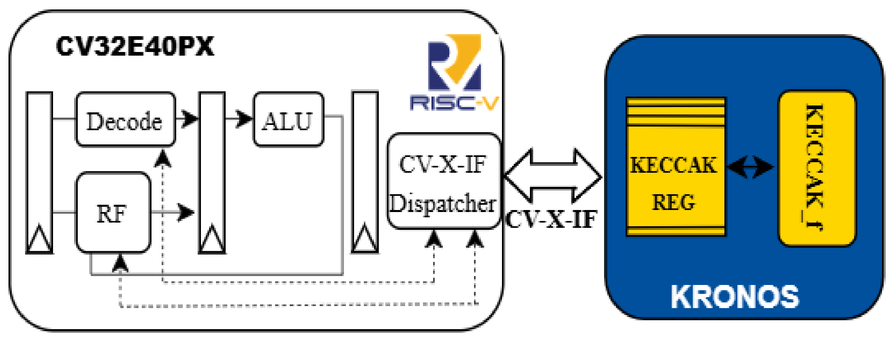

Figure 4. *Coprocessor approach.*

Results

- We take **SHA3-384** function as a case of study to test the three versions of KRONOS. SHA3-384 is part of the SHA-3 family, which is based on the **Keccak** cryptographic permutation: evaluating SHA3-384 directly tests the performance of KRONOS.
- *Table 1* illustrates the **cycle counts** for executing the SHA3-384 function on X-HEEP platform using the three different methods, and their relative speed-ups and throughput. The -O2 flag is used for reference and accelerated simulations.

Table 1. *Cycle counts – SHA3-384.*

Method	Ref.	Accel.	Speed-up factor	Thr. [Mb/s]
Loosely	56,529	4,169	13.56×	4.61
Tightly		31,527	1.79×	0.61
Coproc		7,553	7.48×	2.54

- *Table 2* presents a comparison of the FPGA **area** results for the three versions, implemented on the Zynq UltraScale+ ZCU104 board (xczu7ev-ffvc1156-2-e). The synthesis and implementation are performed using Xilinx Vivado, with a global clock of 50 MHz.

Table 2. *Resource utilization (ZCU104, 50 MHz).*

Method	LUT	Registers	Thr./Area [kb/(s·LUTs)]
Loosely			
◇ KRONOS	4,915	3,252	0.937
○ Controller	7	35	
○ Keccak	4,908	1,617	
○ Register File	0	1,600	
Tightly			
◇ KRONOS	569	139	1.070
○ Controller	569	102	
○ rol_32	0	32	
Coproc			
◇ KRONOS	6,591	3,372	0.386
○ Controller	1,181	125	
○ Keccak	5,409	1,615	
○ Register File	0	1,600	

- **The optimal trade-off between area and throughput is provided by the tightly coupled version.** The absence of a dedicated Keccak register file results in a substantially more compact solution when compared to the other integration methodologies.

Reference

[1] https://github.com/openhwgroup/core-v-xif/tree/main
[2] https://github.com/esl-epfl/x-heep
[3] https://github.com/esl-epfl/cv32e40px