# Integration of a CGRA Accelerator with a CVA6 RISC-V Core for the Cloud-edge

CEI UPM
Centro de Electrónica Industrial

***Juan Granja, Daniel Vázquez, Alfonso Rodríguez, Andrés Otero***

**RISC-V Summit Europe 2025**

***juan.granja@upm.es***

In the context of the development of adaptable nodes for the cloud-edge continuum, this work integrates a Coarse-Grain Reconfigurable Array (CGRA) accelerator with an application-class RISC-V processor on a System on Chip. The resulting platform is deployed on an FPGA, and its performance is evaluated when accelerating a set of relevant tasks, both in a bare-metal environment and under a Linux operating system.

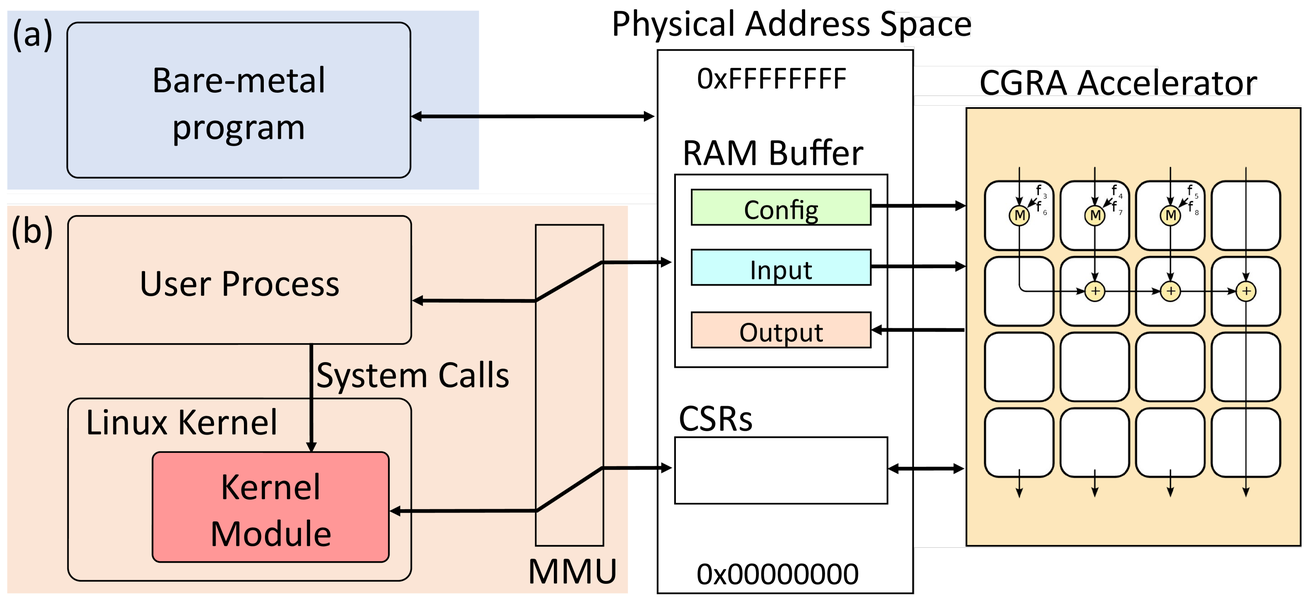
## CGRA Operation

• STRELA CGRA: 4x4 grid of processing elements capable of integer arithmetic, logical & comparison operations on 32-bit words.

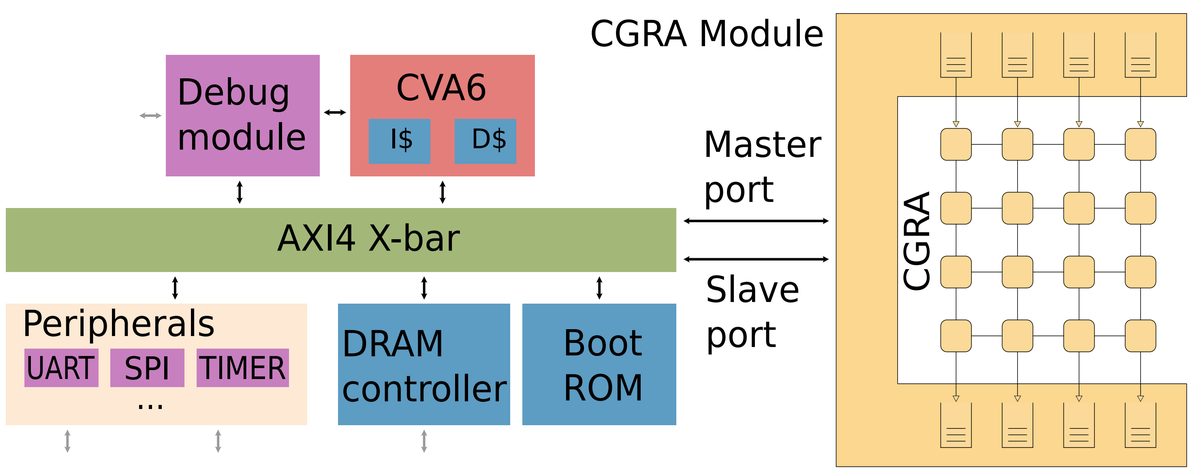
• A code fragment is mapped to the CGRA, either by hand or with a compiler.

• Configuration and input data are placed in memory, after processing, output data is written back to memory.

• Usable under Linux with a custom kernel module.




## CGRA Integration in the CVA6 SoC




• CGRA module as a wrapper with an AXI-4 interface, includes hardware for data provisioning and memory-mapped registers for configuration and control.

• Single-beat DMA interface with multiple outstanding transactions allows non-contiguous accesses to high-latency DRAM.

• Deployed in FPGA emulation of OpenHW Group's CVA6 Development Platform on the KC705 board.

## Results

• Evaluated speedup with respect to software solution with two benchmarks: ReLU (32 KiB of 32-bit words) and 2D Convolution (64x64 images, 3x3 kernel).

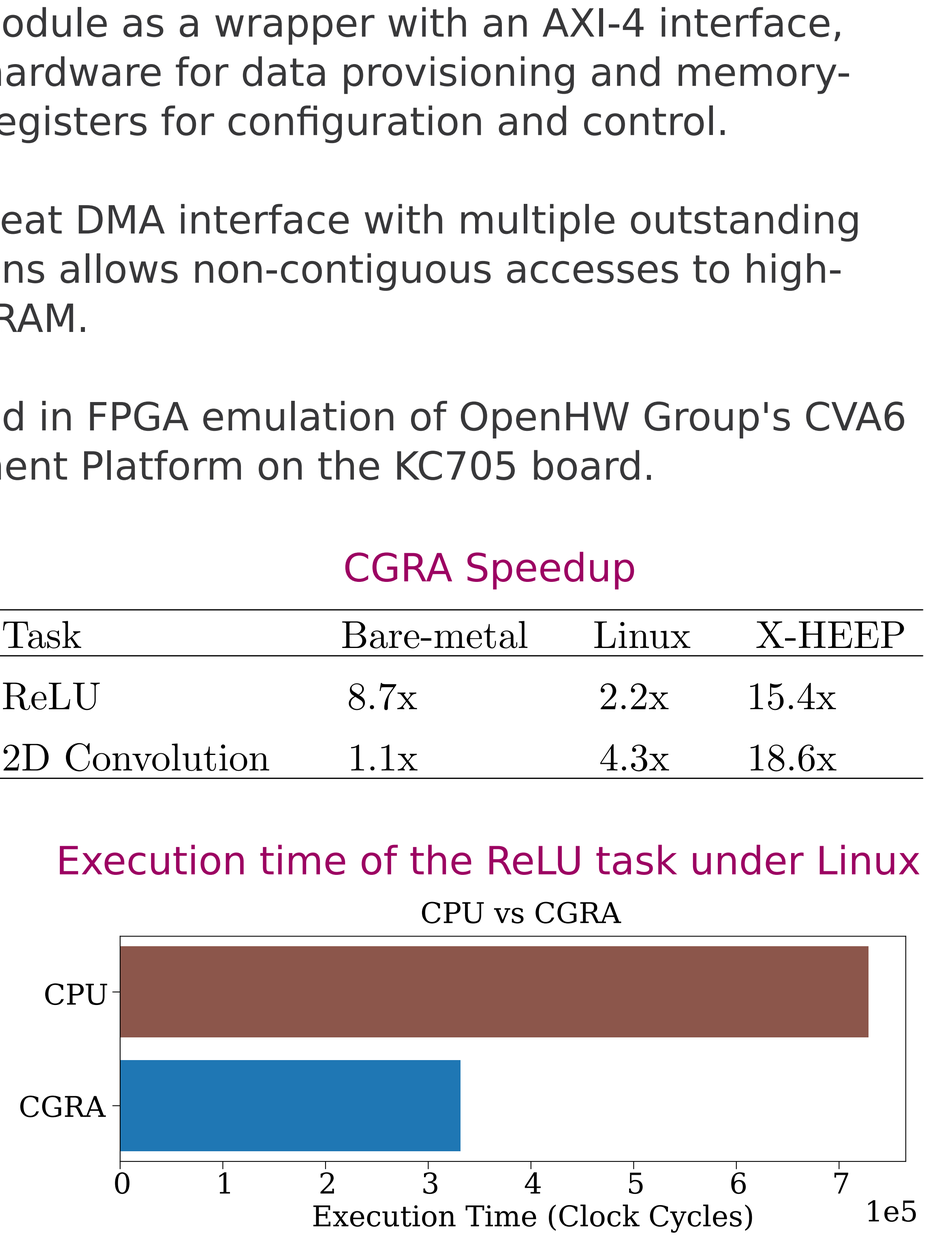
• Single master port for DMA results in less speedup than in previous CGRA integration in the X-HEEP microcontroller platform, with 8 way interleaved on-chip memory and multiple master ports.

• Significant overhead under Linux: 5x slower in software, CGRA execution dominated by data copy to shared memory region.

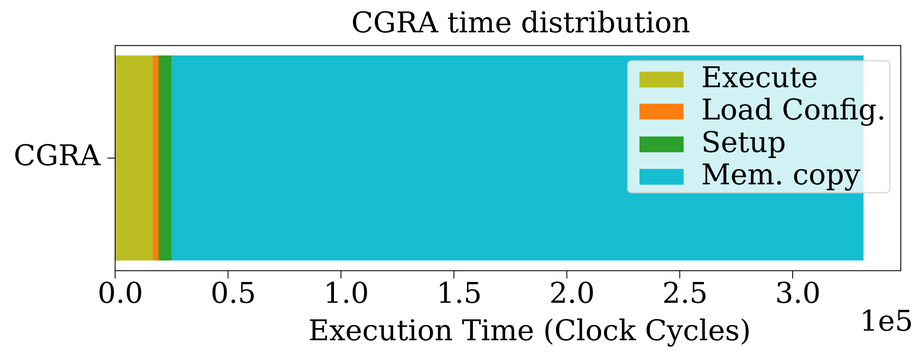
CGRA Speedup

| Task | Bare-metal | Linux | X-HEEP |
|---|---|---|---|
| ReLU | 8.7x | 2.2x | 15.4x |
| 2D Convolution | 1.1x | 4.3x | 18.6x |

Execution time of the ReLU task under Linux




CGRA time distribution




## Future Work

• CGRA configuration with custom RISC-V instructions.

• Interleaved accelerator cache to speed up memory access.

GitHub Repository

UNIVERSIDAD POLITÉCNICA DE MADRID

Funded by the European Union

Grant No. 101135183

MYRTUS