# Comparing CGRAs with VPUs as RISC-V Coprocessors

CEI UPM
Centro de Electrónica Industrial

RISC-V Summit Europe 2024

*Daniel Vázquez, Alfonso Rodríguez, Andrés Otero*
*daniel.vazquez@upm.es*

## Coarse-Grained Reconfigurable Architectures

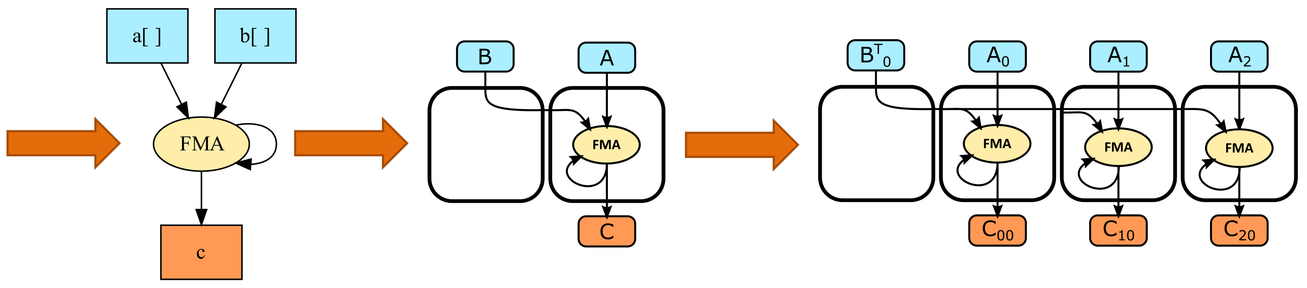
- CGRAs are domain-specific hardware accelerators that can process a wide range of applications with higher energy efficiency and lower execution times when compared to CPUs
- The application offloading consists on the generation of a data-flow graph and its mapping into the architecture

**CGRA Specifications**

- Elastic spatially-configured CGRA
- Architecture made of equal Processing Elements with inputs in the north PEs and outputs in the south PEs
- **32-bit floating-point** operations: *add*, *sub*, *mul* and *FMA* **[HardFloat IPs]**
- The CGRA configuration can be changed between executions

```c
#define LOOP_SIZE 1000

void mac(int a[], int b[], int c) {
    for (int i = 0; i < LOOP_SIZE; i++) {
        c += a[i] * b[i];
    }
}
```




## Integration of the CGRA into Chipyard as a RISC-V coprocessor

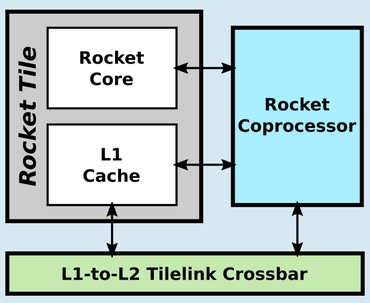
- We use dedicated Direct Memory Access nodes for each north border input and south border output of the CGRA to load and store the required data.
- We treat the CGRA as an extension of the processor datapath, using custom instructions for configuring the memory nodes to load and store the data operands of the operations performed in the CGRA, and load the CGRA configuration from main memory.

**A comparison with the HWACHA Vector Processor Unit (VPU) can be done!**

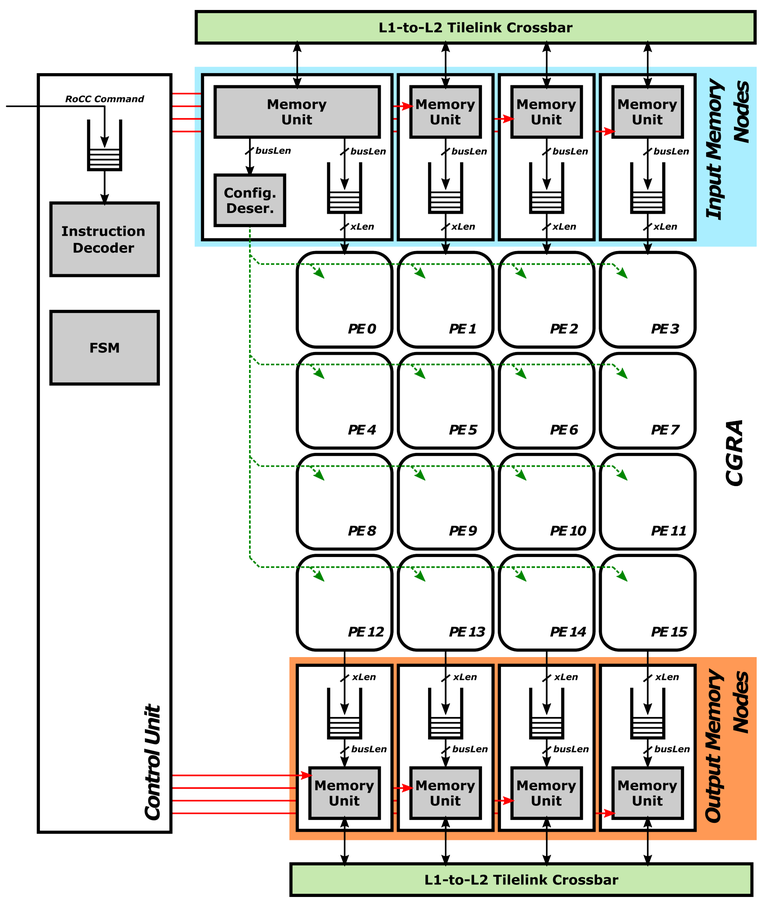
1. Both RISC-V coprocessors are integrated into the same platform: **Chipyard.**
2. They use the same interface to receive the **custom instructions** and to perform **memory operations**: **RoCC Interface.**
3. Both accelerators can perform **32-bit floating** point operations.
4. HWACHA available benchmarks can be run into the CGRA.

**RocketChip [1] and Chipyard [2]:**

- Extend the RISC-V ISA through the RoCC interface
- Direct connection to L1 and L2 caches (high-bandwith connections), and FPU




[1] Asanović et al. "The Rocket Chip Generator". Tech. rep. UCB/EECS-2016-17. EECS Department, University of California, Berkeley, Apr. 2016
[2] Alon Amid et al. "Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs". In: IEEE Micro 40.4 (2020)




## Results and conclusions

- **EXPERIMENTAL SETUP:** A System on Chip (SoC) with a Big Rocket core (**RV64IMAFDC**), 64 KB data cache, 64 KB instruction cache, and 4 GB of external DDR3 RAM memory implemented in the VC709 AMD-Xilinx board (xc7vx690t-2ffg1761c FPGA) is used as the baseline for the experimental setup. Three implementations are generated to compare the hardware accelerators: a plain SoC, a SoC with **Hwacha (default config.)**, and a SoC with the **CGRA Accelerator (1x8 PE array)**. The PMBus interface of the evaluation board is used to measure the FPGA core power consumption.

- **BENCHMARKS:** Three single-precision floating-point vector tests of *esp-tests* are used for the performance comparison, which are the only official benchmarks to use Hwacha.

- **RESULTS:** FPGA hardware utilization results, speedup results of both coprocessors compared with the CPU execution, and power consumption metrics.

**Table 1:** *FPGA resource utilization.*

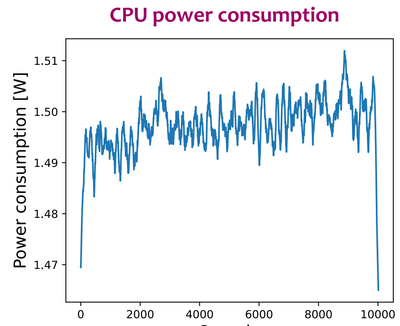
| | LUTs | FFs | BRAMs | DSPs |
|---|---|---|---|---|
| CPU | 5,209 | 2,024 | 0 | 13 |
| CGRA | 26,867 | 13,309 | 32 | 16 |
| Hwacha | 93,475 | 25,987 | 34.5 | 173 |

**Table 2:** *Performance results.*

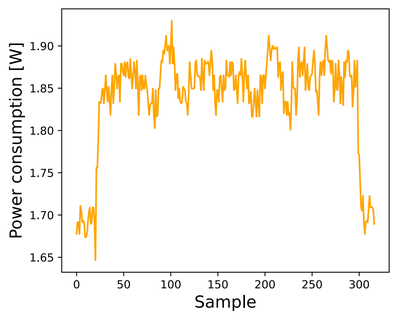
| Benchmark | Hwacha | CGRA |
|---|---|---|
| *vec_vvadd* | 11.30x | 11.39x |
| *vec_saxpy* | 11.45x | 13.64x |
| *vec_sgemm_naive* | 84.35x | 62.04x |
| *vec_sgemm_opt* | 101.03x | - |

**Speed-up vs. CPU*

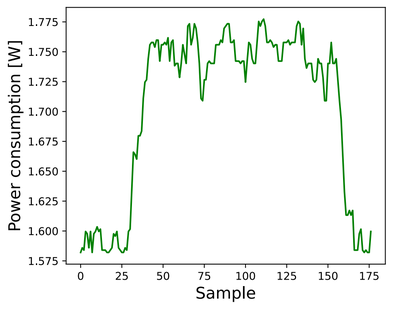
CPU power consumption




VPU power consumption




CGRA power consumption




- **CONCLUSIONS:**

- **The CGRA SoC is enhanced with spatially-distributed computing capabilities to accelerate computing-intensive sections of code.**
- **By extending the RISC-V ISA the control overhead is minimized.**

- **Comparison: Hwacha uses more HW resources than the CGRA because it supports more floating-point operations, uses big vector register files, and it has an instruction cache. Regarding performance, both coprocessors achieve similar metrics. The execution in Hwacha is the one that consumes the most power, attributed to the use of more FPGA resources and more complex design.**

UNIVERSIDAD POLITÉCNICA DE MADRID

This work has been funded by the MCIN/AEI/10.13039/501100011033 and the European Union NextGenerationEU/PRTR (ref. 101096658) under the project AIQ-Ready (ref. PCI2022-135077-2)

MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES
Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
AGENCIA ESTATAL DE INVESTIGACIÓN