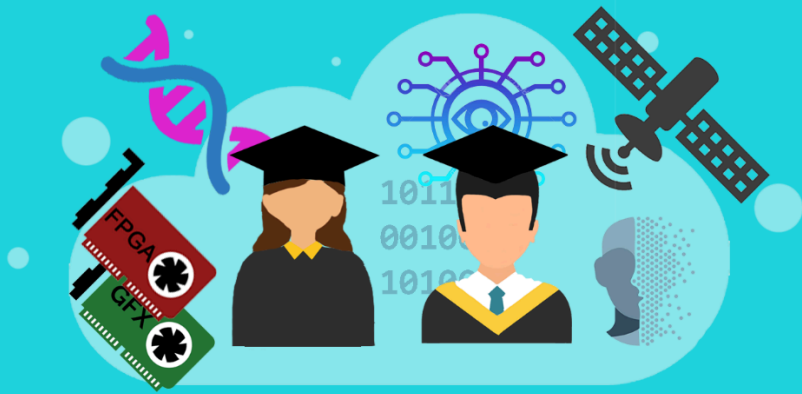




Adaptive Multi-Region CGRA Architectures

Motivation

Rapid adoption of modern DNN and LLM-based solutions increased the need for hardware parallelism, with a great demand of speed with low energy consumption growing each day. Following those demands, the variety of affinities found in certain ML kernels is also increasing the need for reconfigurable hardware to provide efficient and performant inference tasks.

Coarse-Grained Reconfigurable Architectures provide a balanced trade-off between programmability, energy consumption, and performance, being more programmable than task-specific ASIC accelerators while maintaining more performance and efficiency compared to FPGA accelerators.

Although CGRA solutions offer an efficient trade-off, most state-of-the-art solutions still rely on modulo scheduling approaches and fail to co-explore data-level parallelism with task-level parallelism (independent parts of the program's execution graph). Furthermore, constant communication through buses with higher memory hierarchy levels, as well as constant interaction with a host CPU (re-program + control), reduces the efficiency of larger, more complex kernel execution and task-level pipelining.

Trying to approach those limitations, region-based CGRAs expose opportunities for both performance and energy optimization: ChordMap [1] increases kernel concurrency, DynPaC [2] dynamically reallocates resources to relieve pipeline bottlenecks, and ICED [3] exploits voltage–frequency islands to reduce energy consumption. As a specialized case, Alexandris et al. [4] combine approximate computing with reduced-voltage islands for further power savings under accuracy constraints. Across these approaches, the optimization challenge shifts to mapping and resource allocation: assigning computations to regions while balancing parallelism, communication, timing, and power—and, where applicable, accuracy.

Regional flexibility therefore creates the opportunity; effective mapping realizes the gains.

Scope

CGRA Design

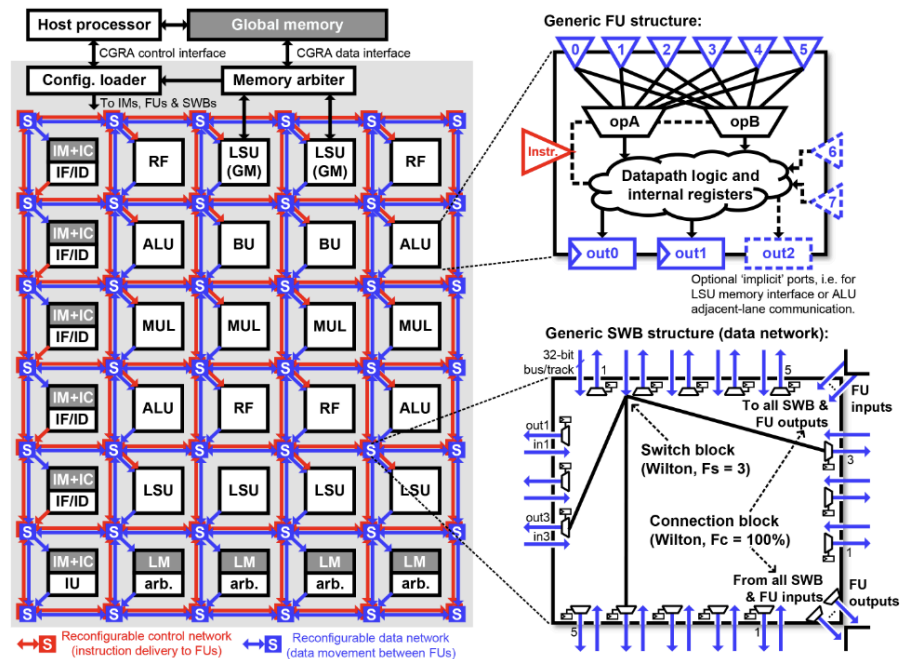


Figure 1: R-Blocks Architecture

As our target CGRA accelerator will use the R-Blocks CGRA [5], a state-of-the-art heterogeneous CGRA approach with each processing element providing a subset of the total ISA provided, transforming the classing CGRA paradigm into a “vectorized CPU” accelerator, with custom Instruction Decode units per computation unit (scalar mode) or N units (SIMD mode with N lines). Two switchbox networks connect all the CGRA elements together, one carrying data and one carrying instruction information. Memory-wise, R-Blocks provide Local Memories for faster access and better locality, and are also configured to be accessed in SIMD mode, as well as external access to external memory with the Load Store Unit (LSU). The Accumulation and Branching Unit (ABU) is responsible for controlling the execution of the program mapped on the CGRA accelerator, sharing a common program counter with all the Instruction Decode units, as well as necessary stalling signals (e.g. when a load from the external memory is performed).

Thesis Targets

Based on the R-Blocks processor-like CGRA paradigm, we can produce multiple “controllable” regions with one interconnected grid. This will allow us, if the correct resources are available (e.g., the right number of IDs to drive the instructions), to provide both variable SIMD/scalar configurations.

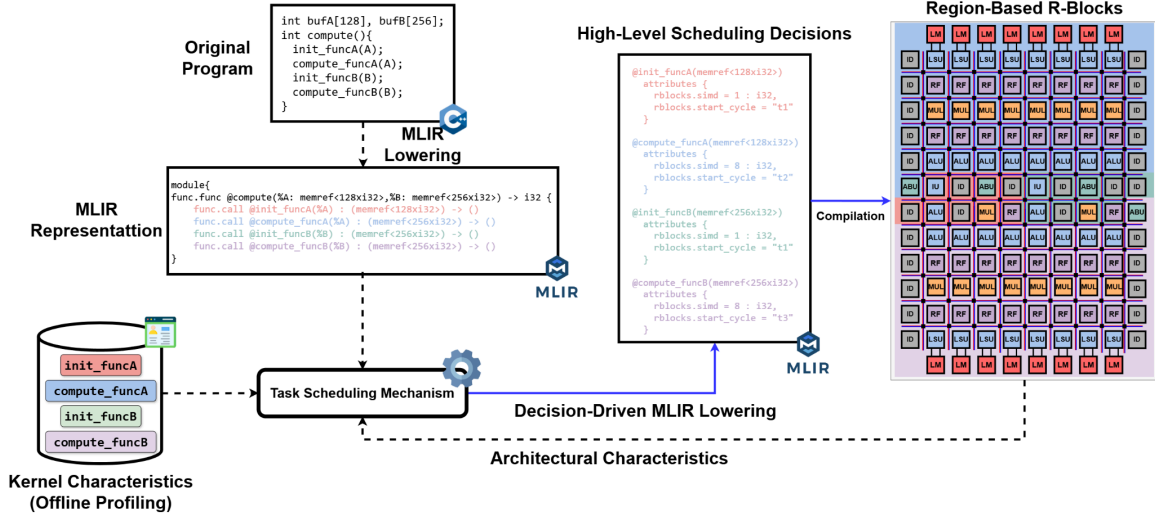


Figure 2: Proposed Thesis Flow (Abstraction)

Based on the proposed flow in Figure 2, this thesis will be based on the MLIR [6] compilation flow, producing several abstraction representations of the to-be-mapped kernel. Based on this abstraction, we will provide a characterization of several kernels, which can be profiled offline (e.g. how the same matrix-matrix multiplication will behave under different parallelization, loop optimizations, and memory availability), and, along with the available resources of our CGRA accelerator (type and quantity of functional units available; e.g., in Figure 2 we have 18 Multiplication Units), we will construct our scheduling mechanism, deciding the best utilization fit of our sub-kernel (defining the “region” that we will occupy), as well as the scheduling timing (defining for “how long” we will occupy that region). Then the final compilation pass will provide the bitstream, including the programming interface for all the regions available.

To simplify this flow based on the same Figure 2 example: The `init_funcA` and `init_funcB` functions are independent of each other, so they can be scheduled at the same time if resources are available. At the same time, those functions only initialize certain points of the program, not performing any compute-intensive operation, so extra parallelism will be “wasted” on those functions. At the same time, `compute_funcA` and `compute_funcB` are compute-intensive but also dependent on the `init_funcA` and `init_funcB` functions, respectively, so they cannot start

at the same time as them, but there is also a chance that they can start before init functions complete (“fused” operations).

Furthermore, providing 4 ABU units, we give the opportunity of “breaking” the programmability also on 4 different regions, with the control being independent among ABUs. This independence gives us a dynamically configurable CGRA fabric (aka some regions can be reorganized and reprogrammed without reprogramming the CGRA).

Finally, this thesis will also be accompanied by certain hardware optimizations, mainly focusing on:

- The interconnection: Long-distance connections will provide unoptimized data routing, so intermediate points will be necessary, buffering the data and breaking long-distance relations, exploring routing efficiency with latency trade-offs.
- Memory partitioning: Local Memories need to be correctly mapped both in the scheduling flow and in the hardware description, being able to “adapt” to the region-based paradigm.

Concluding, this thesis will provide an optimal hardware/software co-design flow, providing the undergraduate student with great knowledge of state-of-the art compilation and mapping strategies and also working on novel ML kernel mapping, down to design optimizations, providing a holistic knowledge base.

RELATED MATERIAL:

- [1] Z. Li, D. Wijerathne, X. Chen, A. Pathania and T. Mitra, "ChordMap: Automated Mapping of Streaming Applications Onto CGRA," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 2, pp. 306-319, Feb. 2022, access: <https://ieeexplore.ieee.org/abstract/document/9351547>
- [2] Q. Zhu, Y. Cao, Y. Qiu, X. Gao, W. Yin and L. Wang, "A Dynamic Partial Reconfigurable CGRA Framework for Multi-Kernel Applications," 2023 International Conference on Field Programmable Technology (ICFPT), Yokohama, Japan, 2023, pp. 298-299, access: <https://ieeexplore.ieee.org/document/10416089>
- [3] C. Tan et al., "ICED: An Integrated CGRA Framework Enabling DVFS-Aware Acceleration," 2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO), Austin, TX, USA, 2024, pp. 1338-1352, access: <https://ieeexplore.ieee.org/document/10764442>

[4] G. Alexandris et al., "Voltage Aware Approximate CGRA Synthesis for Energy Efficient DNN Inference," 2026 Design, Automation & Test in Europe Conference (DATE), Verona, Italy, 2026, pp. 1-7, access: <https://ieeexplore.ieee.org/document/11539365>

[5] Barry de Bruin, Kanishkan Vadivel, Mark Wijtvliet, Pekka Jääskeläinen, and Henk Corporaal. 2024. R-Blocks: an Energy-Efficient, Flexible, and Programmable CGRA. ACM Trans. Reconfigurable Technol. Syst. 17, 2, Article 34 (June 2024), 34 pages. <https://doi.org/10.1145/3656642>

[6] <https://mlir.llvm.org/>

CONTACT INFORMATION:

- Georgios Alexandris, Ph.D Student, NTUA: (galexandris@microlab.ntua.gr)
- Panagiotis Chaidos, Ph.D Student, NTUA: (pchaidos@microlab.ntua.gr)
- Alexis Maras, Ph.D Student, NTUA: (amaras@microlab.ntua.gr)
- Georgios Mytilinaios, Ph.D Student, NTUA: (gmytilinaios@microlab.ntua.gr)
- Assistant Professor Sotirios Xydis, NTUA: (sxydis@microlab.ntua.gr)