Converging AI and HPC: Design and Optimization of a Coarse-Grained Reconfigurable Architecture

Boma Adhi, Chenlin Shi, Jiaheng Liu, Kentaro Sano
RIKEN Center for Computational Science (R-CCS), Kobe, Japan
{boma.adhi, chenlin.shi, jiaheng.liu, kentaro.sano}@riken.jp

***Abstract*—Coarse-grained reconfigurable arrays (CGRAs) are promising post-Moore accelerators for scaling performance in High-Performance Computing (HPC) and Artificial Intelligence (AI). However, fully understanding and realizing the benefits CGRAs bring to these demanding workloads is an open research question. This paper highlights our past and future design-space explorations to optimize our CGRA architecture specifically for HPC and AI applications.**

***Index Terms*—CGRA, HPC, AI, Architecture, Dataflow, Systolic Array**

I. Introduction

Reconfigurable computing offers a middle ground between the fixed efficiency of ASICs and the programmability of general-purpose processors such as CPUs and GPUs. It spans a range of architectures defined by the granularity of reconfiguration, from fine-grained FPGAs to coarser-grained CGRAs, which are closer to specialized hardware.

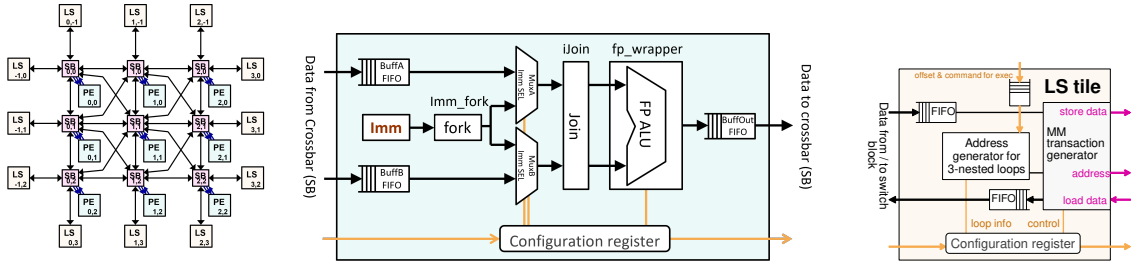
CGRA-like architecture is not new; it has been around since the late 90s and early 2000s, but it was overshadowed by the rapid performance growth of general-purpose CPUs thanks to Moore's law and Denard's scaling. The breakdown of Dennard scaling in the mid-2000s led to the "power wall," limiting further clock speed improvements in general-purpose CPUs. Concurrently, emerging workloads in AI, machine learning, and edge computing demand high parallelism and energy efficiency beyond the capabilities of traditional processors. CGRAs have emerged as promising domain-specific accelerators.

Classical CGRAs were typically targeted for low-power edge applications, but the rise of data-intensive workloads requires a reassessment of CGRA architecture. The growth of artificial intelligence has generated significant demand for processing large datasets with specific computational patterns, particularly dense matrix multiplications and convolutions, as well as complex non-linear and sparse computations. In parallel, traditional high-performance computing applications in scientific fields require a more general-purpose computing capability with higher floating-point performance (often FP64 precision) and the capacity to handle vast data volumes. Modern HPC applications also increasingly incorporate AI techniques, thus demanding more versatile architectures. Identifying optimal design-space strategies for CGRA deployment in modern HPC plus AI environments remains an open research challenge. This paper highlights our past and future design-space explorations to optimize our CGRA architecture specifically for HPC and AI applications.

RIKEN CGRA is a research CGRA implemented with the CGRA-ME framework. It is designed as a modern HPC accelerator with mixed AI and HPC usage in mind. There are 3 key design characteristics:

- **Elastic stream computing model:** The stream computing model is a class of dataflow computing paradigm that strictly separates computation and memory access. This paradigm prevents stalls in memory access from directly affecting computations, keeping the pipeline full. Furthermore, the elastic CGRA paradigm or dynamic dataflow execution allows the CGRA to seamlessly handle operations with variable latencies (like IO or memory accesses) and applications with complex control flow or synchronization requirements, common in large-scale HPC environments.
- **Floating-point and multi-precision with SIMD:** High-precision floating-point operations support is a must in HPC environments. In contrast, AI workloads heavily utilize mixed precision (BF16, FP16, INT8, or FP8) for performance/efficiency and only occasionally FP32 for stability. Innovative PE design with mixed precision support is highly desired.
- **Modularity and scalability:** RIKEN CGRA has a parameterized modular design with scalability in mind. Custom tiles can be easily added. Intra-CGRA datapath can be easily converted into AXI4-Stream or Avalon-ST with minimal glue logic. All the external interfaces are compatible with Avalon-MM or AXI4-Lite.

The CGRA mainly comprises of 3 types of tiles, i.e., Processing Element (PE) tiles for computing, Load-Store (LS) tiles for memory access, and Switch Block (SB) tiles for routing, as shown in Figure 1. New type of tiles may be added for future extension. Each tiles is typically connected in a king-style fashion to its nearest-neighbor tiles. A daisy-chained bitstream interface is also employed for configuration. This architecture is defined in C++ with CGRA-ME [1], thus it is synthesizable, highly parameterizable and can be easily extended for design space exploration.



a. Simplified 3 × 3 RIKEN CGRA overview.

b. Processing Element Tile.

c. Load-Store Tile.

Fig. 1. RIKEN CGRA Architecture

II. CGRA Design-space Optimization for HPC & AI

A. Intra-CGRA Interconnect Optimization

An appropriately designed intra-CGRA interconnect provides enough routability to map common HPC workload kernels while maintaining low hardware costs. Typical AI workloads often satisfied with simple systolic array-like interconnect. We explored ADRES-style integrated mux in each PE vs. HyCube-style interconnect with a discrete router. The latter is far more routable, albeit with 6.3× higher resource usage [2]; however, it is often underutilized at less than 30% connection usage under a typical HPC workload [3]. To find the balance, we evaluate various reduced connection topologies by removing the least used connections and higher compute density topologies [4].

B. Heterogeneous ALUs

Various HPC kernels often require transcendental functions, such as square root and exponential. However, Coarse-Grained Reconfigurable Arrays (CGRAs) usually do not directly implement such operations but instead rely on approximations (e.g., based on lookup tables or Taylor expansion) that can negatively impact the result precision or the number of Arithmetic Logical Units (ALUs)/Processing Elements (PEs) required. For this reason, we provide our CGRA with a heterogeneous set of ALUs, namely Basic, Complex, and Full. The first ALU implements basic logical, integer, and floating-point operations, while the second features transcendental functions only. Finally, the last ALU groups the previous ones. As the number and complexity of operations increase, the required hardware resources also increase. We synthesized three 4×4 CGRAs on 3-nm FinFET, each implementing a different type of ALU. The results indicate a slight increase in resources usage as the ALU becomes more complex. Consequently, instead of using just one type of ALU, we combined them within a heterogeneous CGRA and investigate a proper balance to reduce resource usage and, potentially, power consumption [5].

C. Systolic-Array Style Execution on CGRA

AI workload is dominated by GEMM operations, which involve many floating-point calculations and memory accesses, creating a significant computational burden on general-purpose processors. Specialized hardware accelerators have been developed to address this challenge, with systolic arrays (SAs) emerging as a prominent architecture for GEMM acceleration. Compared to SAs, CGRA offers higher flexibility and programmability as users can map arbitrary DFG into the CGRA. However, the flexibility costs a higher logic complexity, thus resulting in more area usage in silicon. Optimization target for AI often differs from that for HPC. Moreover, mapping an arbitrary DFG does not guarantee 100% utilization of the available PEs, reducing the efficiency further compared to a simple systolic array. In our latest effort, we proposed several changes to allow systolic array style execution on CGRA, i.e., extra systolic style registers for local data storage on each PE, which is commonly found in systolic array style computation, and defining a DSL to efficiently contruct the mapping primitives that can be placed easily into the CGRA utilizing maximum number of available PEs [6].

References

[1] J. Anderson, B. Adhi, C. Cortes, E. D. Sozzo, O. Ragheb, and K. Sano, "Exploration of compute vs. interconnect tradeoffs in CGRAs for HPC," in *Proceedings of the 13th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*, ser. HEART '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 59–68. [Online]. Available: https://doi.org/10.1145/3597031.3597055

[2] B. Adhi, C. Cortes, Y. Tan, T. Kojima, A. Podobas, and K. Sano, "The cost of flexibility: Embedded versus discrete routers in CGRAs for HPC," in *Proceedings of the IEEE Cluster conference*, 2022, pp. 347–356.

[3] B. Adhi, C. Cortes, E. D. Sozzo, T. Ueno, Y. Tan, T. Kojima, A. Podobas, and K. Sano, "Less for more: Reducing intra-CGRA connectivity for higher performance and efficiency in HPC," in *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2023, pp. 452–459.

[4] B. Adhi, C. Cortes, T. Ueno, Y. Tan, T. Kojima, A. Podobas, and K. Sano, "Exploring inter-tile connectivity for HPC-oriented CGRA with lower resource usage," in *2022 International Conference on Field-Programmable Technology (ICFPT)*, 2022, pp. 1–4.

[5] E. Del Sozzo, X. Wang, B. Adhi, C. Cortes, J. Anderson, and K. Sano, "Exploration of trade-offs between general-purpose and specialized processing elements in HPC-oriented CGRA," in *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2024, pp. 668–680.

[6] C. Shi, B. Adhi, S. Miwa, and K. Sano, "Enabling systolic computing on elastic coarse- grained reconfigurable array for HPC and AI," in *2025 62th ACM/IEEE Design Automation Conference (DAC)*, 2025.