RIKEN CGRA: Data-Driven Architecture as an Extension of Multicore CPU for Future HPC

Boma Adhi
Processor Research Team
RIKEN Center for Computational Science
Kobe, Japan
boma.adhi@riken.jp

Takuya Kojima
Department of Information and Computer Science
Keio University
Tokyo, Japan
tkojima@am.ics.keio.ac.jp

Yiyu Tan
Next Generation High Performance Architecture Research Team
RIKEN Center for Computational Science
Kobe, Japan
tan.yiyu@riken.jp

Artur Podobas
Department of Computational Science and Technology
KTH Royal Institute of Technology
Stockholm, Sweden
podobas@kth.se

Kentaro Sano
Processor Research Team
RIKEN Center for Computational Science
Kobe, Japan
kentaro.sano@riken.jp

***Abstract*—A Coarse-Grained Reconfigurable Array (CGRA) is currently gaining momentum to enter the HPC market as Deep Learning accelerators due to its potential for performance scalability and energy efficiency. However, traditionally the usage of CGRA was limited in embedded systems to provide extra computing performance with low power consumption. This poster describes our research journey to find what it takes for CGRA to become an ideal HPC accelerator.**

***Index Terms*—Reconfigurable Computing, CGRA, Data-driven, Multicores, Compilers**

I. Introduction

In the forthcoming post-Moore era, many promising architectures for High-Performance Computing (HPC) have been proposed [1]. Like quantum and neuromorphic computing, those newly proposed architectures often force users to abandon the currently available and mature software ecosystem. Meanwhile, other approaches, like reconfigurable computing, are more practical yet still impactful by alleviating problems associated with current generation architecture. While the current multicore CPU based on von Neumann architecture is currently popular, it has some inherent inefficiencies [2], including Low ALU density, complicated memory hierarchy, and inefficient data movement among cores, to name a few of the problems.

Reconfigurable data-driven computing allows efficient and scalable hardware tailored for target problems, providing a higher compute density than a general-purpose multicore CPU [3]. It also enables a more DRAM-friendly memory access pattern, thus ensuring more efficient memory bandwidth utilization. One of the most popular reconfigurable systems in HPC is Field-Programmable Gate Arrays (FPGA). It has been proven capable of providing a better performance and power consumption over conventional CPU-based systems. However, FPGA's flexibility with fine-grain reconfigurability has drawbacks, such as a lower clock frequency, a larger area, higher power consumption, and time-consuming compilation. By trading off some of FPGA's flexibility, another class of reconfigurable device, the Coarse-Grained Reconfigurable Array (CGRA), gains more performance and efficiency with a less complicated configuration process.

This work is partially supported by the New Energy and Industrial Technology Development Organization (NEDO) Project JPNP16007 and Japan Society for the Promotion of Science (JSPS) KAKENHI Grant JP20H00593 and JP21H04869.

II. CGRA in HPC

CGRAs are traditionally used in embedded-devices to assist the CPU with extra computing power while keeping the energy consumption low [4]. However, recently, CGRA-like architectures are also getting momentum in HPC as a deep-learning accelerator with multiple companies offering such a product. In order to understand how CGRA can be used in an HPC environment, we previously conducted a literature survey on CGRA and made several key findings [4]:

- Most CGRA researches focus on small-sized CGRA with no floating-point support, which is not suitable for HPC.
- Most proposed CGRAs only exploit instruction-level parallelism, which is insufficient for larger CGRA with a more complex memory hierarchy.
- The exploration of the memory subsystem is often overlooked, ignoring its impact on memory-bounded applications commonly found in the HPC environment.

Driven by the survey result, we previously developed a framework for a design-space exploration of CGRAs in an HPC environment [5]. We observed the impact of different memory technologies and several architectural parameters, like SIMD, on-chip buffers, and parallelism levels with seven hand-written benchmark applications on our CGRA simulator. However, we quickly realize that this is inadequate to understand how CGRA behaves in a real HPC environment.

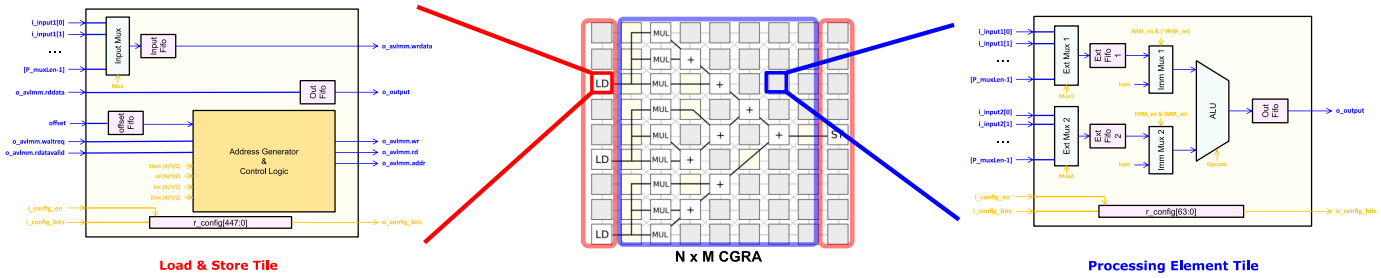

Fig. 1. RIKEN Coarse Grained Reconfigurable Array 2

CGRA in the HPC environment tends to be larger than in embedded systems. There are several research questions as we scale the size of the CGRA. Depending on the target application domain, heterogeneous tiles may become necessary, opening up research questions such as the composition of tiles, their functionality, and the (inter-tile) connectivity. The memory subsystem should be properly considered, such as, the design or the load-store unit, and how the load and store unit talks to multiple memory controllers. Also, how the CGRA accesses the host CPU memory and vice versa. On the compiler side, the Place-and-Route algorithm can be implemented in multiple ways, and each of them has many different parameters to adjust.

In this work, we would like to explore those research questions above, therefore we decided to port our simulator to run on top of our FPGA cluster [6] attached to an HPC system. This approach allows us to test much larger CGRA sizes or multiple CGRA islands running on different FPGAs across the cluster. We also introduced a Genetic-Algorithm-based compiler to generate CGRA's bitstream from C codes annotated with OpenMP directives. Moreover, we are exploring many different options of CPU-CGRA interconnect, which affect the performance and, most importantly, the programming model. Integrating CGRA as part of the CPU core complex itself may provide benefits like lower latency communication with the CPU core, but it may have some restrictions such as logic size limitations and sharing the CPU's memory bandwidth. On the other hand, placing the CGRA further from the CPU, for example, over a PCIe bus, allows a larger design with higher power consumption and thermal dissipation. However, this approach might complicate the programming model as the CGRA has to access the host CPU memory through the PCIe bus.

III. RIKEN CGRA2

The new CGRA is still based on the same design principle of the earlier Riken High-Performance CGRA. The CGRA is reimplemented in SystemVerilog targeting Intel FPGA in our FPGA cluster. As depicted in Figure 1, the current design has two tiles: LD tile is responsible for memory access and calculating memory address based on the bitstream provided by the CPU; PE tile takes two streams from adjacent tiles and performs an arithmetic calculation based on the configured opcode. We use industry-standard AvalonMM for memory interface and AvalonST for interconnection between tiles. The usage of industry-standard interconnect allows easier integration of other kinds of tiles, like on-chip buffer tiles for data reuse, custom computes tiles, or even third-party IPs. It also allows us to easily utilize hardened features of the FPGA like External Memory interface, PCIe interface, and networking for inter-FPGA communications.

The compiler for our CGRA [7] consists of two parts, the LLVM based front-end, which takes OpenMP annotated C code then compiles it into Data-Flow Graph (DFG), and the backend compiler, which performs a Place-and-Route (PR) to map the DFG to the CGRA based on various constraints and criteria.

IV. Conclusion

It is challenging to find the best practice of designing a CGRA suitable for future HPC. This work in progress is far from over. We are sharing what we have learned and what we can explore in the near future.

References

[1] J. S. Vetter, E. P. DeBenedictis, and T. M. Conte, "Architectures for the Post-Moore Era," *IEEE Micro*, vol. 37, no. 4, pp. 6–8, 2017.

[2] N. Jouppi, C. Young, N. Patil, and D. Patterson, "Motivation for and evaluation of the first tensor processing unit," *IEEE Micro*, vol. 38, no. 3, pp. 10–19, 2018.

[3] R. Hartenstein, "A decade of reconfigurable computing: a visionary retrospective," in *Proceedings Design, Automation and Test in Europe. Conference and Exhibition 2001*, 2001, pp. 642–649.

[4] A. Podobas, K. Sano, and S. Matsuoka, "A Survey on Coarse-Grained Reconfigurable Architectures From a Performance Perspective."

[5] ——, "A Template-based Framework for Exploring Coarse-Grained Reconfigurable Architectures," in *2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2020, pp. 1–8.

[6] K. Sano, T. Ueno, T. Miyajima, H. Jens, and A. Koshiba, "ESSPER: Development of a Scalable and Flexible FPGA Cluster System for High Performance Computing (in Japanese)," Tech. Rep., 2021.

[7] T. Kojima, N. A. V. Doan, and H. Amano, "Genmap: A genetic algorithmic approach for optimizing spatial mapping of coarse-grained reconfigurable architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 11, pp. 2383–2396, 2020.