RIKEN CGRA: Reconfigurable Data-Driven Architecture for Future HPC

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

Carlos Cortes
carlos.cortestorres@riken.jp
RIKEN Center for Computational Science
Kobe, Hyogo, Japan

Yiyu Tan
tan.yiyu@riken.jp
RIKEN Center for Computational Science
Kobe, Hyogo, Japan

Takuya Kojima
tkojima@hal.ipc.i.u-tokyo.ac.jp
University of Tokyo
Tokyo, Japan

Artur Podobas
podobas@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

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

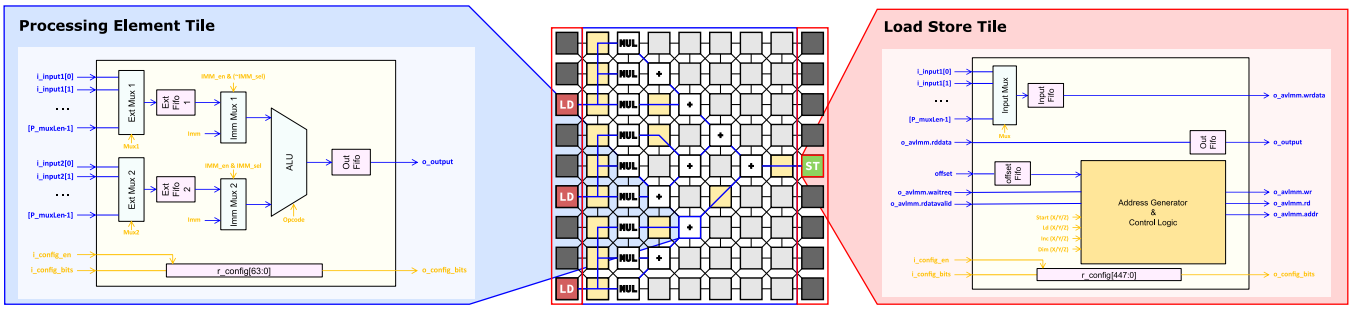

Figure 1: RIKEN Coarse Grained Reconfigurable Array

1 INTRODUCTION

CGRAs were traditionally used in embedded-devices to provide the CPU with extra computing power with low energy consumption. In order to understand how CGRA can be used in an HPC environment, we previously conducted a literature survey on CGRA[2] and developed a framework for a design-space exploration of CGRAs in the HPC environment[3]. Based on this result, we proposed a CGRA architecture which allows testing a much larger CGRA or multiple CGRA islands running on an FPGA cluster connected to Fugaku supercomputer, which enable us to explore research questions such as the heterogeneous composition of tiles, their functionality, and the (inter-tile) connectivity; exploration of memory subsystem, such as, the design or the load-store unit, and how the load and store unit talks to multiple memory controllers.

2 RIKEN CGRA

The CGRA is implemented in SystemVerilog targeting Intel FPGA in our FPGA cluster. As depicted in Figure 1, the current design has two tiles: Load Store (LS) tile is responsible for memory access and calculating memory address based on the bitstream provided by the CPU; Processing Element (PE) tiles performs an arithmetic calculation based on the configured opcode taking two streams of data from adjacent tiles. 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[1] 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.

ACKNOWLEDGMENTS

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.

REFERENCES

[1] Takuya Kojima, Nguyen Anh Vu Doan, and Hideharu Amano. 2020. GenMap: A Genetic Algorithmic Approach for Optimizing Spatial Mapping of Coarse-Grained Reconfigurable Architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, 11 (2020), 2383–2396.

[2] Artur Podobas, Kentaro Sano, and Satoshi Matsuoka. [n.d.]. A Survey on Coarse-Grained Reconfigurable Architectures From a Performance Perspective. ([n. d.]). https://doi.org/10.1109/ACCESS.2020.3012084

[3] Artur Podobas, Kentaro Sano, and Satoshi Matsuoka. 2020. A Template-based Framework for Exploring Coarse-Grained Reconfigurable Architectures. In *2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. 1–8. https://doi.org/10.1109/ASAP49362.2020.00010