RIKEN CGRA:

Unleashing CGRA Potential for HPC

Boma Adhi[†], Emanuele Del Sozzo[†], Xinyuan Wang[§†], Carlos Cortes[†], Tomohiro Ueno[†], Kentaro Sano[†]

[†]Center for Computational Science (R-CCS), RIKEN, Japan, email: {boma.adhi, emanuele.delsozzo, carlos.cortestorres, tomohiro.ueno, kentaro.sano}@riken.jp
[§]University of Toronto, Canada, email: xinyuan.wang@mail.utoronto.ca

Why CGRA for HPC?

CGRA is a reconfigurable computing device traditionally used for accelerators in low-powered embedded devices, akin to an FPGA but **trades the bit-level programmability with ASIC-like Power, Performance, and Area (PPA)**, and is a **promising candidate for computing accelerators in the HPC space**. However, embedded systems and HPC are vastly different domains with different application scales and constraints, thus requiring some design-space exploration and research before one can practically include CGRAs in HPC systems. This poster summarizes our effort to explore the suitable **CGRA architecture for HPC**.

RIKEN CGRA is built around these design philosophies:

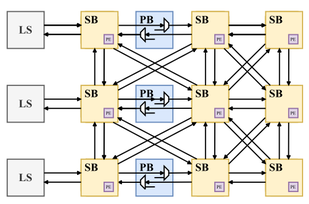
- **Modular** and **scalable** design for high throughput
- **Separation** between computing and memory operations
- **Floating-point** operations capability with **OpenMP** support

Programmable Buffer Tile

Programmable Buffers are introduced to enhance the throughput of LS tiles: LS tiles are commonly connected to memory technologies which provide **optimal performance for sequential memory access only**. On the other hand, iterative stencil loops and similar applications cause **repeated and interleaved** memory accesses. PB tiles **ensure that those are buffered and reordered**, ensuring **sequential memory access** in the LS tile.

PBs can be configured for various applications: For stencil-like computation, they act as a shift register with parallel outputs. Offsets of parallel outputs are configurable and the buffer supports insertion of additional samples at the start and end of data packets.

A large buffer can be formed by chaining multiple, independent PB tiles in a column. To reduce resource overhead, PB columns reuse the normal CGRA interconnect. For the same reason, PB tiles connect in-between two existing SBs instead of containing a SB. When an application does not need PBs, multiplexers allow to transparently bypass them. PB columns are located both close to LS to improve memory throughput, and between PE tiles, to buffer intermediate results.



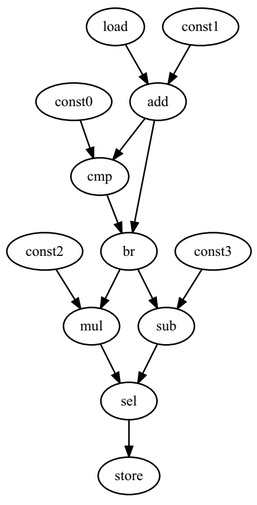
Connection of PBs between CGRA's SBs. Multiplexers allow bypassing PBs.

Predication on CGRA

Predication support is necessary to improve the programability of the CGRA, allowing the CGRA to do conditional branching.

Several design decisions must be made, for example the type of predication, either data-based predication or a true DFG branching. The former is much simpler to implement but might waste a lot of PEs as both side of the branch are always executed. In addition, the type of predication network, either a dedicated predication network or sharing the datapath. A further study on the trade-offs between all these factors is necessary.

Also, we must consider the compiler support. A regular CPU oriented program predications happen on the control-flow side of the program, however branch instruction in CGRA means a diverging data flow path to take different set of instructions. Transforming control flow branch to data path branch is non trivial and requires further research.



This is an example of a DFG containing predication. New instructions for comparison of floating point to generate predicates must be created. Branch instrution will only pass data to the side dictated by the predicates and selet will only pass data if there is a valid token on the taken input path.

Potential Collaboration

Please do not hesitate to send us an email for questions or comments.

We are open for collaborations in various research direction, such as:

- Applications/Benchmarks
- HPC/AI workload acceleration
- Compiler Support
- Silicon based implementation
- Large-scale CGRA Organization
- CPU-CGRA Interconnect
- Memory Architecture
- Mixed precision arithmetic
- and many more!

RIKEN CGRA Architecture

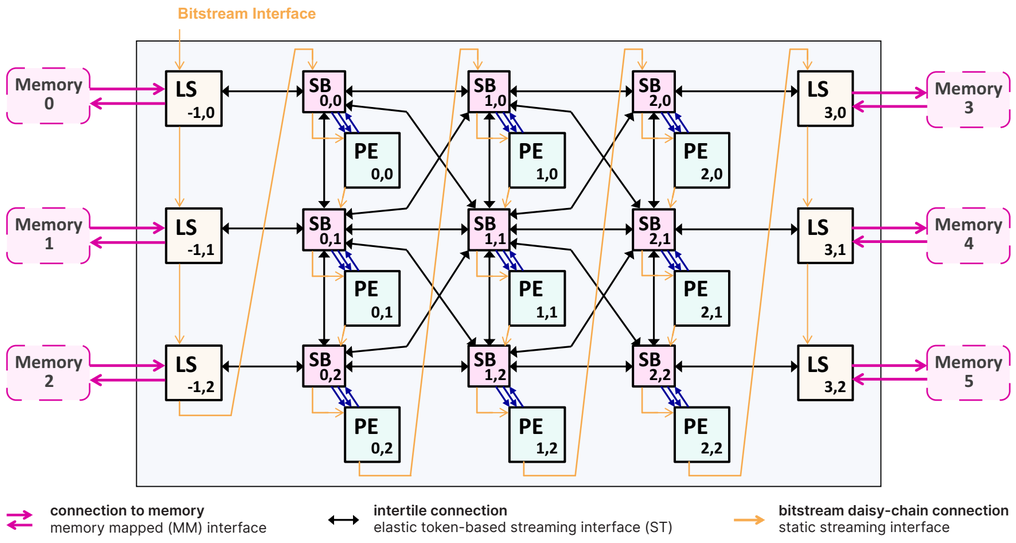
RIKEN CGRA is both a CGRA architecture and its evaluation framework comprising a CGRA generator, RTL simulator, benchmarking tools, CGRA Place & Route (PR) compiler, and FPGA-based emulation on ESPPER FPGA Cluster connected to **Supercomputer Fugaku.**

The CGRA consists of three different tiles connected by elastic token-based streaming interfaces. These modules are:

- **Processing Element (PE) tile,**
- **Load and Store (LS) tile,**
- **Switch Block (SB) tile.**

Different kinds of tiles can be added for future expansion. All tiles in the CGRA are connected with a daisy-chained bitstream interface to program the CGRA.

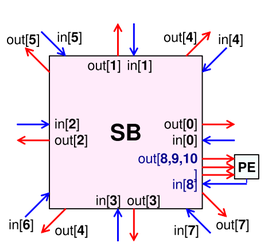
B. Adhi, C. Cortes, Y. Tan, T. Kojima, A. Podobas and K. Sano, "Exploration Framework for Synthesizable CGRAs Targeting HPC: Initial Design and Evaluation," 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Lyon, France, 2022, pp. 639-646, doi: 10.1109/IPDPSW55747.2022.00113.

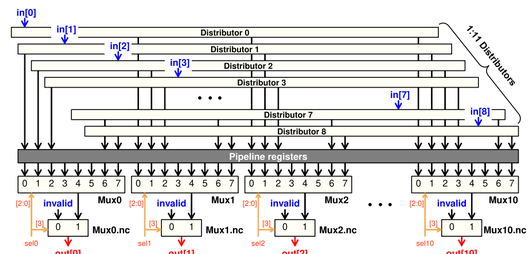


Switch Block Tile (SB) & Interconnect Research

SB tile is responsible for connecting all other tiles forming the whole array. In the baseline configuration, each SB tile is connected to its own PE and all eight neighboring tiles, forming a King-style mesh connection. The SB tile allows static routing with multiple streams concurrently passing through the same SB from multiple inputs to multiple outputs. It is also allowed for a single input stream to be broadcasted to multiple outputs.

Compared to a more traditional ADRES-like CGRA with Embedded Router (CGRA-ER), the SB-based design has a much higher routability albeit with 6.3 times resource usage. At the baseline configuration, our prior research has shown that the interconnect is overprovisioned, thus having a low utilization. Further research to find the balance is necessary.

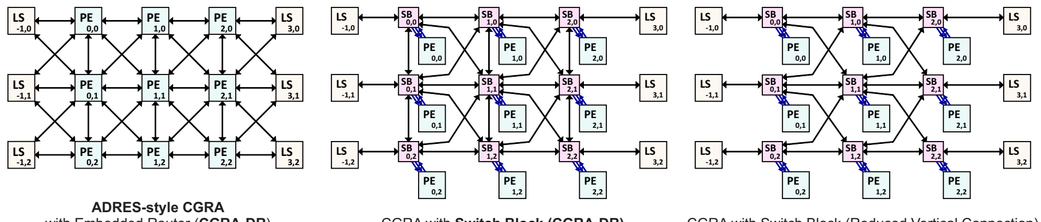




B. Adhi, C. Cortes, Y. Tan, T. Kojima, A. Podobas and K. Sano, "The Cost of Flexibility: Embedded versus Discrete Routers in CGRAs for HPC," 2022 IEEE International Conference on Cluster Computing (CLUSTER), Heidelberg, Germany, 2022, pp. 347-356, doi: 10.1109/CLUSTER51413.2022.00046.

Different inter-tile CGRA interconnect resources usage is shown on the right.

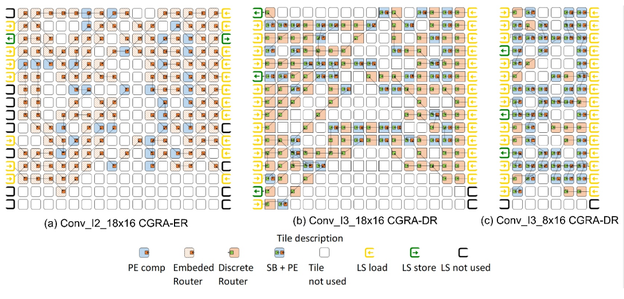
The routing richness of SB based CGRA with Discrete Router (CGRA-DR) design allow a denser mapping. Reducing the connectivity by removing the least used connection does not affect routability.



ADRES-style CGRA
with Embedded Router (**CGRA-DR**)
MUX: **2x 8:1** on PE and **1x 8:1** on LS

CGRA with **Switch Block (CGRA-DR)**
MUX: **10x 8:1** on SB

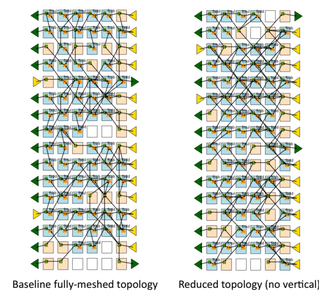
CGRA with Switch Block (Reduced Vertical Connection)
MUX: **8x 8:1** on SB



Left: Not only CGRA-DR allows mapping more kernel unrolls, but SB based design allows denser mapping on smaller CGRA.

Right: CGRA-DR has a much higher resources usage. However, eliminating least usage vertical connections has no significant impact on routability while reducing 30% resources usage.

B. Adhi et al., "Less for More: Reducing Intra-CGRA Connectivity for Higher Performance and Efficiency in HPC," 2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), St. Petersburg, Florida USA, 2023, (accepted)



Systolic Array Style Computation on CGRA

Compared to **Systolic Arrays**, CGRA offers higher flexibility and programmability as users can map arbitrary DFG into the CGRA. RIKEN CGRA originally borrowed some design element of a systolic array to guarantee the performance level required in HPC environbment.

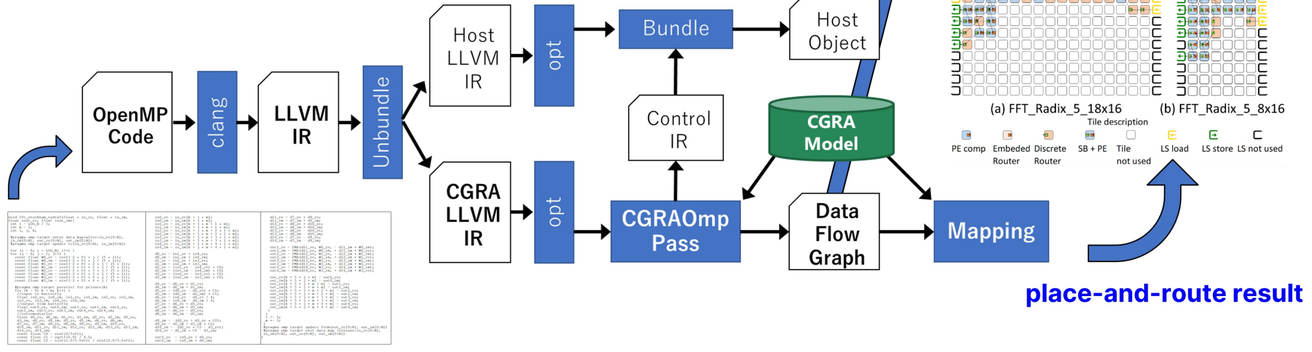
However, in several HPC scenario, the flexibility offered by CGRA costs a higher logic complexity, thus resulting in more area usage in silicon. Moreover, **mapping an arbitrary DFG does not guarantee 100% utilization of the available PE**, reducing the efficiency further compared to a simpler systolic 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, more memory friendly access pattern, modifying intra-CGRA interconnects, and defining a DSL to efficiently construct the mapping primitives that can be placed easily into the CGRA utilizing maximum number of available PEs.

Tradeoffs between those design decision mentioned above shall be explored in future works.

Data Flow Mode: FFT example

- Using GenMap compiler: GA-based multi-objective and general-purpose mapping framework for CGRA
  - ✓ Find solutions in a Pareto-front for mapping a DFG onto CGRA (**place-and-route: P&R**)
  - ✓ Select the best set of solutions as final P&R solution
- LLVM-based front-end for OpenMP

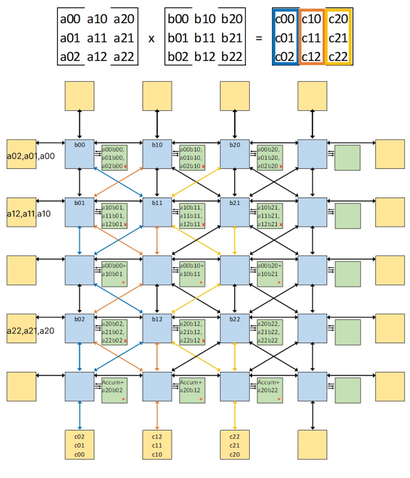


https://github.com/hungalab/GenMap

Systolic Mode: example for GEMM (highly regular)

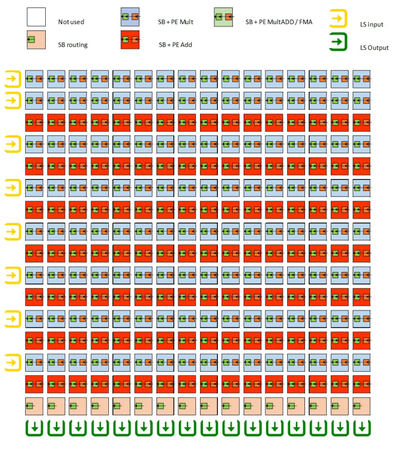
Unmodified RIKEN CGRA (using Elastic interface)

- First stage: 2 PE for initial multiplication and 1 adder.
- The next stage is 1 PE for multiplication and 1 adder for the previous multiplication result and the accumulator.



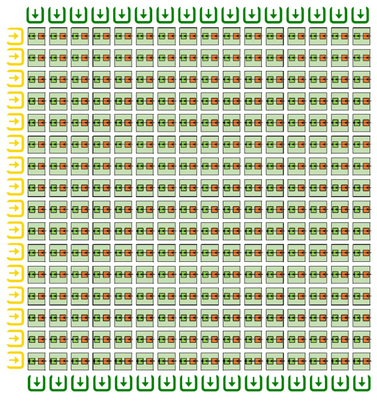
With Baseline PE, MM 8x16

- Matrix A is streamed from the left LS tiles
- Matrix B is stored on IMM register, updated via bitstream interface.
- Accumulation is done vertically. Result on the bottom LS tiles.



With FMA enabled PE, GEMM/MM 16x16

- Matrix A is streamed from the left LS tiles
- Matrix B is streamed from top (or right) onto the systolic buffer on each PE tile.
- Matrix C (optional) is stored on IMM register, updated via bitstream interface.
- Accumulation is done vertically. Result on the bottom LS tiles.

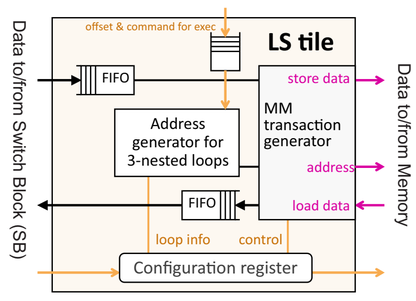


Load & Store Tile (LS)

LS tile performs memory read/write operation and streams the data to/from other tiles.

- Using **Intel AvalonMM memory-mapped interface** with pending reads.
- Address generator logic supports **3-nested deep loop with variable stride size and offsets**.

B. Adhi et al., "Exploring Inter-tile Connectivity for HPC-oriented CGRA with Lower Resource Usage," 2022 International Conference on Field-Programmable Technology (ICFPT), Hong Kong, 2022, pp. 1-4, doi: 10.1109/ICFPT56656.2022.9974525.

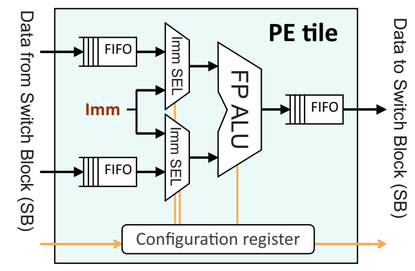


Processing Element Tile (PE)

PE tile is responsible for computing:

- **32-bit floating-point** capability
- **12 opcodes** (including integer and logic operations)
- **2 inputs** and **1 output** (all buffered to absorb slack)
- Single Immediate value register configurable through bitstream register

B. Adhi et al., "Exploring Inter-tile Connectivity for HPC-oriented CGRA with Lower Resource Usage," 2022 International Conference on Field-Programmable Technology (ICFPT), Hong Kong, 2022, pp. 1-4, doi: 10.1109/ICFPT56656.2022.9974525.



Processing Element Tile with Heterogeneous ALUs

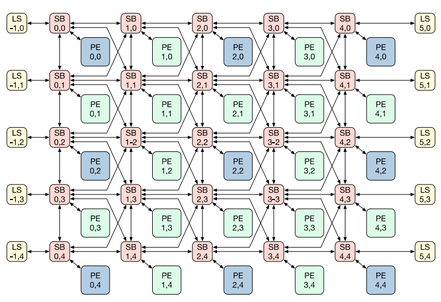
Various HPC kernels often **require transcendental functions**, such as square root and exponential. However, CGRA usually do not directly implement such operations but instead **rely on approximations** (e.g., based on lookup tables or Taylor expansion) that **can negatively impact the result precision or the number of ALU or PE required**. However, we can observe that a slight increase in logic resources, DSPs, and RAM blocks as the ALU becomes more complex. Consequently, instead of using just one type of ALU, we plan to combine them within **a heterogeneous CGRA** and investigate **a proper balance** to reduce resource usage and, potentially, power consumption.

TABLE I
LIST OF OPERATIONS FOR EACH ALU TYPE

ALU Type	Operations
BASIC	Logical: NOP, AND, XOR, OR, NOT Integer: ADD, SUB, MUL, DIV Floating-point: ADD, SUB, MUL
COMPLEX	Floating-point: EXP, LOG, INV, SQRT
FULL	Logical: NOP, AND, XOR, OR, NOT Integer: ADD, SUB, MUL, DIV Floating-point: ADD, SUB, MUL, EXP, LOG, INV, SQRT

TABLE II
RESOURCE USAGE OF A 3 3 CGRA FEATURING DIFFERENT ALU TYPES ON A STRATIX 10 SX FPGA

ALU Type	Resource Usage (%) ALM	Register	RAM	DSP
BASIC	22541 (2.4%)	37686 (1.0%)	0 (0.0%)	59 (1.0%)
COMPLEX	24467 (2.6%)	44419 (1.2%)	189 (1.6%)	180 (3.1%)
FULL	29328 (3.1%)	51033 (1.4%)	405 (3.5%)	221 (3.8%)



PE Basic ALU (16/25)

PE Complex/Full ALU (9/25)

Estimated resources (ALU only):

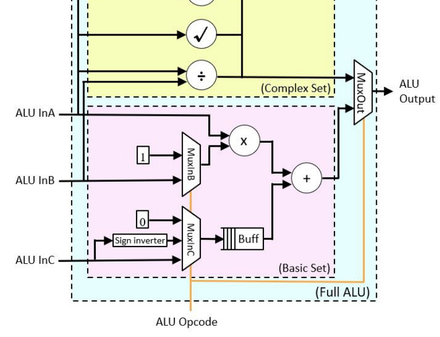
- **Basic + Complex**:
  - ALMs: 1.70 %
  - Registers: 0.91 %
  - RAMs: 1.61 %
  - DSPs: 4.20 %
- **Basic + Full**:
  - ALMs: 2.09 %
  - Registers: 1.01 %
  - RAMs: 3.46 %
  - DSPs: 4.98 %

An example of heterogeneous layout. Different layout & PE composition might be required for different application domain.

For this purpose, a new ALU has to be designed. The new ALU has three inputs and two major parts. The two parts are not necessarily coexist in a single PE.

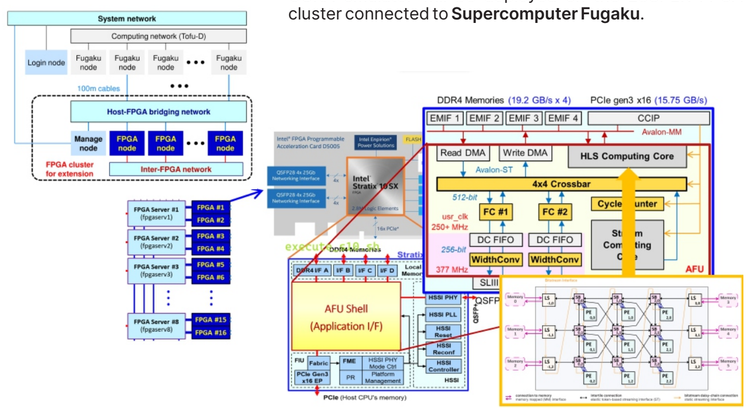
The basic part consist of an adder and a multiplier arranged as shown on the right to allow both used concurrently as an FMA unit.

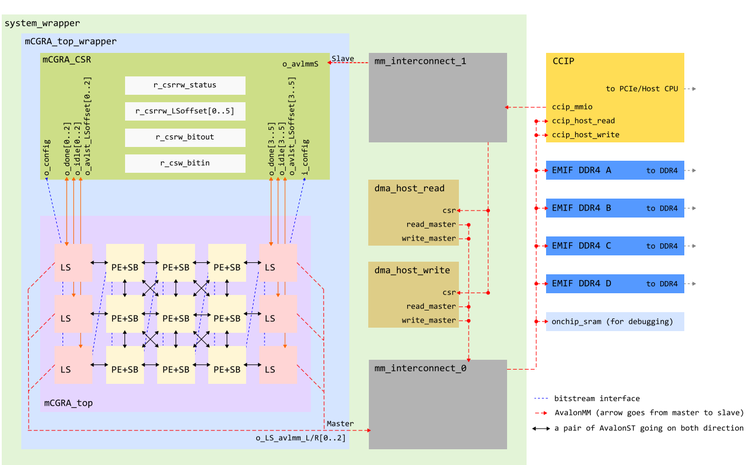
The complex part consists of multiple larger FP unit. In this example square-root, exponent, logarithm and division. This complex part is optional and may not present in all Pes.



FPGA-Based Emulation on Real HPC Environment

For a large-scale evaluation in an HPC environment, we are developing **FPGA-based emulation** of the CGRA. The CGRA emulator will be deployed on the ESSPER FPGA cluster connected to **Supercomputer Fugaku**.





The CGRA, including a CSR Module to control the CGRA operation is embedded into RIKEN AFUshell which is based on Intel AFU Shell. The architecture of the CGRA-based emulator is depicted on above.

The FPGA shell is based on RIKEN AFU shell which is an extension of Intel AFU shell running on top of Intel D5005 Datacenter FGPA Card. The CGRA's LS tiles are connected to the onboard DDR4 memory controller, thus all computations are done to and from the FPGA's onboard memory. A set of DMAs are used to transfer data between host CPU and the CGRA's memory.

A host code running on the host CPU interact with the CGRA core through the CSR interface with the help of AFU Shell Class Library and DMA Library.