Bridging the Gap Between Fine-Grained Architectures and Large Systolic Arrays

Andrea Belano, Francesco Conti
University of Bologna, Bologna, Italy

MOTIVATION

The challenge in accelerating modern deep neural networks lies not only in the immense scale of state-of-the-art models but also in the heterogeneity of their computational profiles.

While some architectures, such as traditional Convolutional Neural Networks, generally use smaller kernels and relatively simple activation functions, transformer-based models heavily rely on very large matrix multiplications and generally complex dataflows, in part due to the multi-head attention mechanism.

This wide variety of architectures and their rapid evolution make evident the need for an architecture which is performant with large matrices but at the same time capable of parallelizing smaller, distinct tasks.

WHY NOT JUST A BIG TENSOR ENGINE?

At the same FMA/cycle budget, a single large tensor unit is generally more efficient than multiple smaller ones thanks to its better data reuse and it efficient pipeline.

However, this advantage is lost when, instead of a single big workload, we have many small tasks that taken individually do not completely fill the array.

It is in these cases that a fleet of smaller units has the edge, allowing us to issue multiple different operations in parallel and achieve a much higher percentage of the peak theoretical performance.

Can we have the flexibility of smaller units while keeping the benefits of a large engine?

NAUSICAA AT A GLANCE

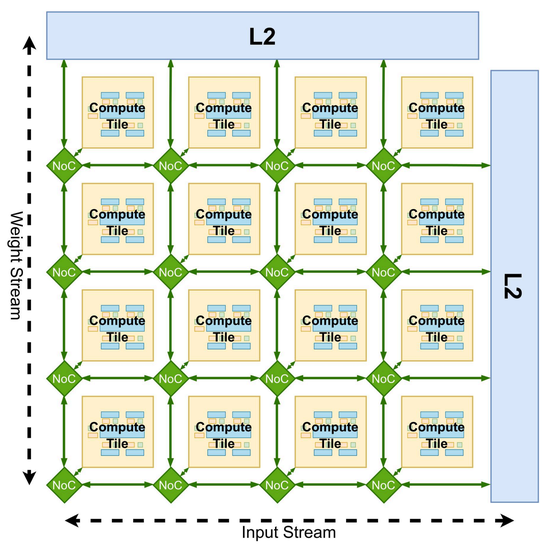
To accelerate the increasingly heterogeneous workloads found in modern DNNs, we propose NAUSICAA (Neural Acceleration Unit for Scalable Integration and Configurable Adaptive Architecture), a multi-tile architecture comprised of multiple identical compute tiles organized in a mesh connected through a Network-on-Chip.

Each tile features its own shared scratchpad memory, specialized units for matrix multiplications and non-linear functions, and a DMA.

THE MESH ARCHITECTURE

The tiles are connected using a Network-on-Chip based on the FlooNoC architecture, featuring a wide 512-bit channel for data transfers and narrow 64-bit channel for inter-tile communications.

To optimally use the vertical and horizontal NoC channels, the L2 memory banks are distributed along the right and top edges of the mesh. The banks on the right store inputs, and the banks on the top store weights. By doing this, we avoid the network contention that would result from storing both inputs and weights along the same dimension.



THE COMPUTE TILES

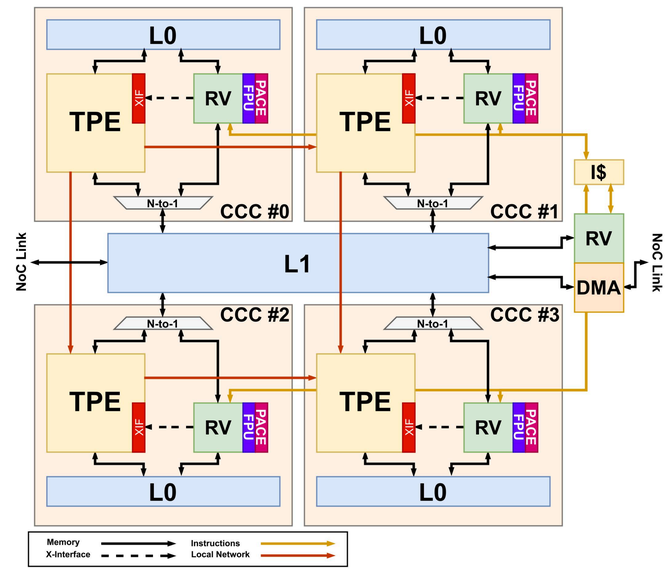
Each compute tile features 128 KB of shared L1 scratchpad memory and a RISC-V RV32IAMC control core based on the Snitch architecture. This core manages memory operations, synchronizes operations across tiles, and acts as the interface for inter-tile communication.

Actual computations are carried out by four Compute-Core-Complexes (CCCs) operating in a Single-Program-Multiple-Data paradigm.

Each CCC features a RISC-V RV32IMCF core, also based on the Snitch architecture and capable of 4-way SIMD.

The core ISA has been also extended to support the approximation of non-linear functions through piecewise polynomial approximations using the PACE architecture.

To accelerate linear layers, each CCC features a dedicated 8x8 FMA tensor processing engine based on the RedMulE architecture and supporting context pipelining.



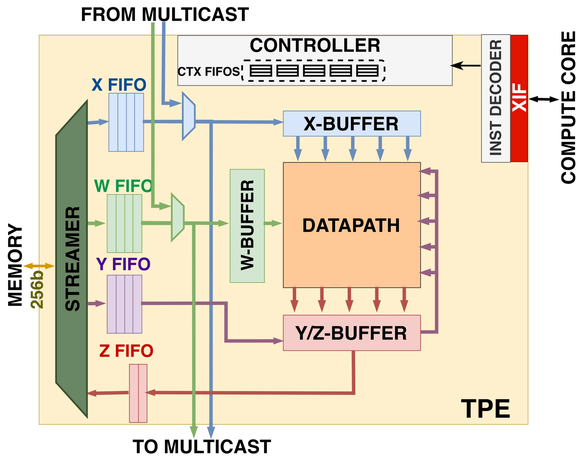
THE TENSOR PROCESSING ENGINE

RedMulE is an input-stationary, systolic array accelerator supporting the FP16 and both FP8 formats.

In this work it has been extended with a V1.0-compliant eXtension InterFace to allow fast programming and synchronization with the compute core through custom instructions.

Another improvement is the support for context pipelining to allow the seamless concatenation of different operations and a full utilization regardless of the tile size as long as new operations are continuously pushed.

It also features ports to the local multicast network to share input and weights when the accelerators are used in parallel (detailed later).



THE CONTROL CORE

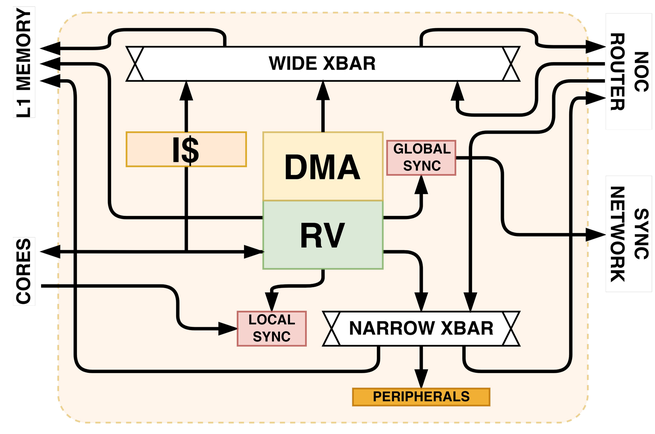
The control core is a simple RV32IAMC based on the snitch architecture and capable of controlling the tile DMA through an ISA extension.

It is the only core capable of interacting with the other tiles and the L2 memory thanks to the connection to the NoC's narrow channel.

The core is connected to a local synchronization network, which allows the synchronization between the control cores and the compute core complexes in one cycle, as well as to an inter-tile synchronization network, based on FractalSync, which enables synchronization across the tiles with complex patterns.

The control core's ISA has been further extended with two custom instructions:

- FSSYNC, which handles up to 6 synchronization patterns in parallel using 3 source registers; useful for complex kernels.
- FSSYNCI, for a single fast synchronization using an immediate.

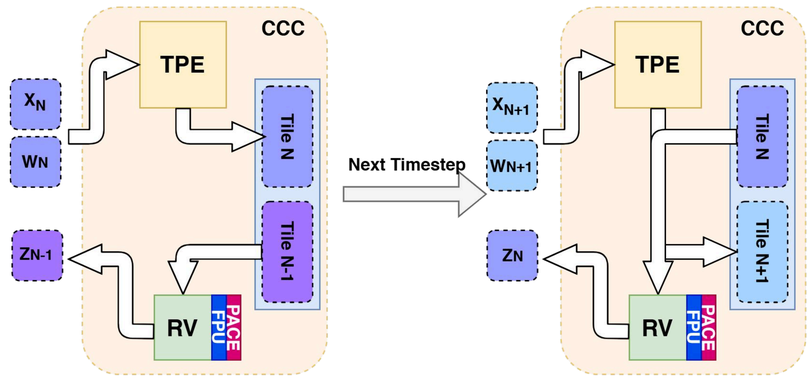


MEMORY LAYOUT

To enable the tensor processing engine and the core to operate in parallel, simply increasing the number scratchpad memory banks would complicate the interconnect too much.

Instead, each CCC has 32KB of private L0 scratchpad memory, only accessible by the core and the tensor processing engine.

This design choice stems from the observation that nonlinear functions are almost exclusively applied to the output of a linear layer. By letting the tensor processing engine place the intermediate results into the L0 scratchpad, the system achieves a pipelined execution: when the engine is computing the next tile, the core will apply the required nonlinearity on the output of the previous linear computations and store the result back in L1 memory.

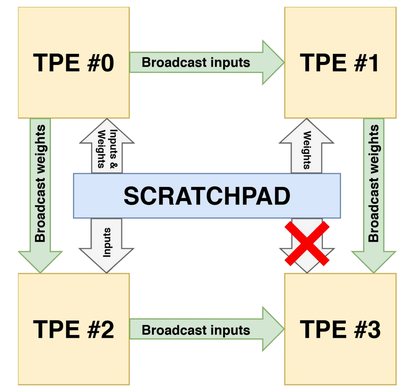


KEEPING BANDWIDTH AT BAY

The problem of using 4 8x8 tensor processing units is that they require double the memory bandwidth compared to a single 16x16 instance.

To solve this, each tile features a local multicast network. The tensor processing engine normally work independently, but when performing a large matmul, inputs and weights are broadcast across CCCs, ensuring each tile is only fetched once.

With this optimization, the total bandwidth required by the array of tensor processing engines when performing a matrix multiplications is the exact same as the one of a larger monolithic engine.

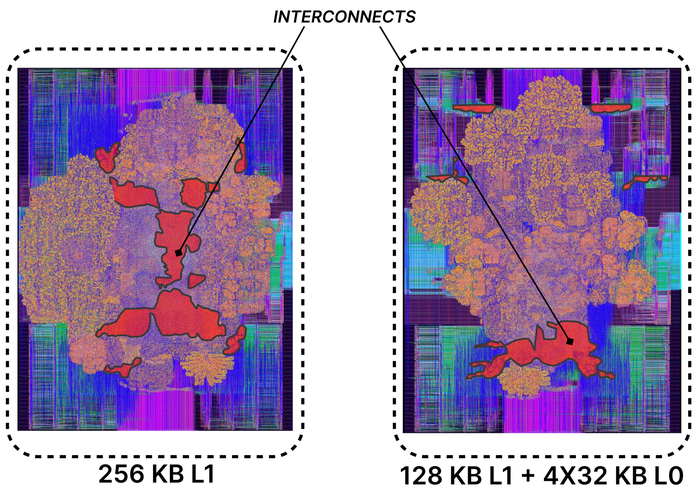


EFFECTS OF PARTITIONING

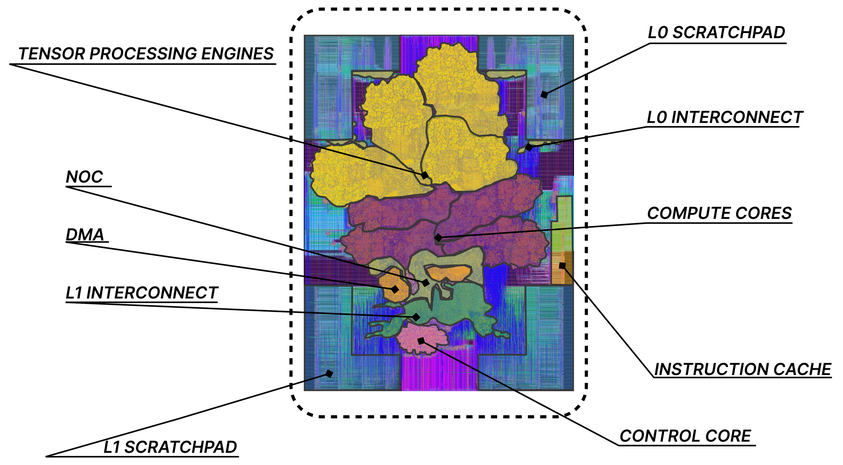
Partitioning the memory into two separate levels has a major impact on the tile's interconnect.

Compared to a cluster with only a 256 KB L1 memory and the same total number of banks:

- The interconnect area is reduced by 50%
- The number of violations at the end of the implementation process is reduced to 63 from 17552, making the design practically implementable



COMPUTE TILE LAYOUT



Corresponding author: andrea.belano2@unibo.it

Isachi, Victor, et al. "FractalSync: Lightweight Scalable Global Synchronization of Massive Bulk Synchronous Parallel AI Accelerators." Proceedings of the 22nd ACM International Conference on Computing Frontiers. 2025.
Fischer, Tim, et al. "FlooNoC: A 645-Gb/s/link 0.15-pJ/B/hop open-source NoC with wide physical links and end-to-end AXI4 parallel multistream support." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 33.4 (2025): 1094-1107.
Tortorella, Yvan, et al. "RedMulE: A compact FP16 matrix-multiplication accelerator for adaptive deep learning on RISC-V-based ultra-low-power SoCs." 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2022.
Prasad, Arpan Suravi, et al. "PACE: An optimal piecewise polynomial approximation unit for flexible and efficient transformer non-linearity acceleration." 2025 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). Vol. 1. IEEE, 2025.
Zaruba, Florian, et al. "Snitch: A tiny pseudo dual-issue processor for area and energy efficient execution of floating-point intensive workloads." IEEE Transactions on Computers 70.11 (2020): 1845-1860.