ComputeRAM in AI accelerators: An LLM case study

Contact : business@synthara.ai

Date	Version number	Description
15 Mar 2025	0.1.0	Document Creation.
19 Mar 2025	0.2.0	First Complete Draft
26 M6 2025	1.0.0	First release

Table of Contents

Executive Summary

This document is a study on the integration of Synthara's ComputeRAM (CxR) technology into a reference AI accelerator, leading to an estimated 3.7x increase in compute efficiency (TOP/s/W) and a 20% reduction in chip area without any loss in throughput. We also show that the integration of ComputeRAM is seamless and non-disruptive, making its adoption straightforward.

The reference architecture is modeled to represent state-of-the-art high-performance engines such as Nvidia Tensor Cores, Tenstorrent Tensix Cores, Graphcore IPU units, or AMD Matrix Core Units. Additionally, we estimate how ComputeRAM's benefits scale across technology process nodes, showcasing its adaptability and continued performance advantages. This system-level analysis underscores the transformative potential of ComputeRAM to dramatically enhance modern AI accelerator designs.

Introduction

This document evaluates the integration of ComputeRAM (CxR) technology into large AI-focused accelerators or Systems-on-Chip (SoCs), estimating the system-level performance gains from substituting traditional SRAM memory with Synthara's ComputeRAM. The analysis begins with defining a baseline architecture featuring multiple independent cores, each equipped with local memory and dedicated MAC arrays, reflecting common modern designs in AI acceleration. When appropriate, the document also provides references to state-of-the-art systems at the time of writing this paper.

The document demonstrates that CxR improves all key performance metrics, such as silicon area, cost, and energy efficiency. These improvements arise from embedding computation directly into memory, significantly reducing the overhead associated with data transfers between the local memories and compute units.

The metrics presented are based on a generic model to provide a clear and realistic evaluation. For an analysis specifically tailored to your architecture or workload, please contact us at business@synthara.ai

Use-case under analysis

The starting point of our analysis is a traditional, non-ComputeRAM-based architecture similar to what can be found on the market today. This section outlines the model we have chosen as the baseline, highlighting its key metrics and parameters. We use GF22 to perform this analysis, the process node at which we are sampling our product today. We are also working on porting CxR to new process nodes, including TSMC FinFETs, and our expected scaling performance is detailed in a later section.

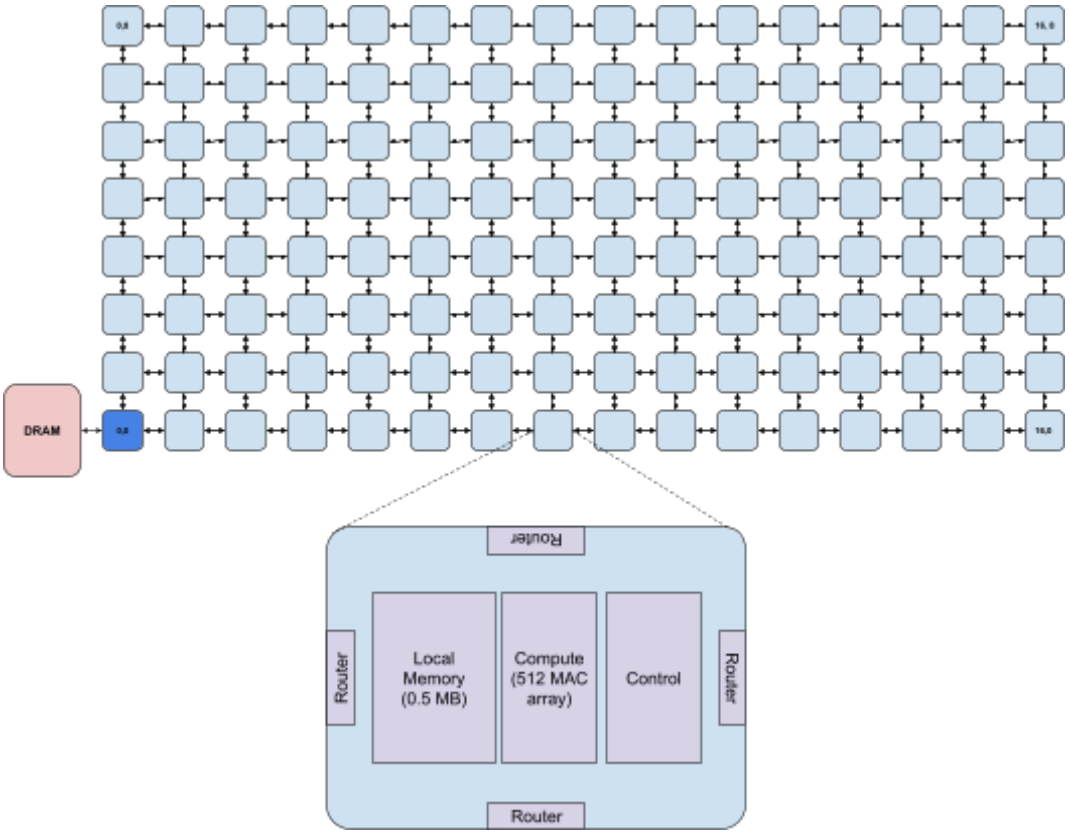
Baseline System Description

Architecture

We will start our analysis with an AI accelerator composed of 128 independent processing units with 0.5 MB of local memory available for computing. Each processing unit is equipped with an engine for accelerating matrix operations, operates autonomously, and is capable of communicating efficiently with others to distribute workloads and exchange intermediate results. The architecture mimics[1] modern multi-channel bus interconnects, NoC architectures, or shared memory structures where tensor-processing units or SIMD

[1] Examples: Nvidia Tensor Cores, Tenstorrent Tensix Cores, Graphcore IPU units, or AMD Matrix Core Units

cores compute independently and can simultaneously share data with their neighbors. The figure below illustrates the target architecture.



The main system parameters are:

- Process node: GF22
- System frequency: 1 GHz
- 128 independent processing cores with
  - 0.5 MB SRAM each, for a total of 64MB split over 512 banks (4 banks/core)
  - 512 MAC each, for a total of 65K MAC
  - The cores are arranged in an 8x16 grid where each core can communicate with its neighbors.
- Each core can receive and transmit (in parallel) 64-bit per cycle for an effective bandwidth of 8 GB/s
- A bandwidth with external DRAM memory of 8 GB/s, connected with the node in position (0,0) in the grid

Baseline Area

Given that 128 KB memory in GF22 is about 220k μm^2 and a simple MAC is about 1.5k μm^2, we obtain

$$512\ banks\ \times\ 220k\ \mu m^2\ =\ 112.7\ mm^2\ SRAM\ Area$$
$$65k\ MAC\ \times\ 1.5k\ \mu m^2\ =\ 98.3\ mm^2\ MAC\ Area$$

For routing, control, and other logic in the design, we assume it to be approximately of the same size as the MAC engine itself, leading to a final area breakdown of

$$112.7\ mm^2\ SRAM\ Area$$
$$98.3\ mm^2\ MAC\ Area$$
$$98.3\ mm^2\ logic\ Area$$

$$112.7\ +\ 98.3\ +\ 98.3\ =\ 309.3\ mm^2\ total\ area$$

Baseline Energy Efficiency

To evaluate the energy efficiency of the system, we estimate an energy cost of ~0.57 pJ/op with the following breakdown:

- Compute Cores (incl. memory): 0.50 pJ/op
- NoC Interconnect: 0.0197 pJ/op
- Control Logic: 0.0455 pJ/op

The cost for compute cores is estimated from Synthara's placed and routed designs and verified as compatible with data published in scientific literature[2]. For control and interconnect, we computed the contribution based on the workload (as defined in a later section), and the detailed derivations of these values are provided in the appendix. In this process node, we model leakage as roughly 6.5 nW/µm² at 25°C. This value is based on the process datasheets, simulations and our empirical observations from past designs.

It must be noted that the performance metrics listed earlier in this section are typically hard to achieve and require careful optimization of the system architecture and the physical implementation of the SoC. This sets a high bar for Synthara to improve upon.

[2] Jokic, P., Azarkhish, E., Bonetti, A., Pons, M., Emery, S., & Benini, L. (2021). A construction kit for efficient low power neural network accelerator designs https://arxiv.org/abs/2106.12810

Schuiki, F., Schaffner, M., & Benini, L. (2018). NTX: An energy-efficient streaming accelerator for floating-point generalized reduction workloads in 22nm FD-SOI https://arxiv.org/abs/1812.00182

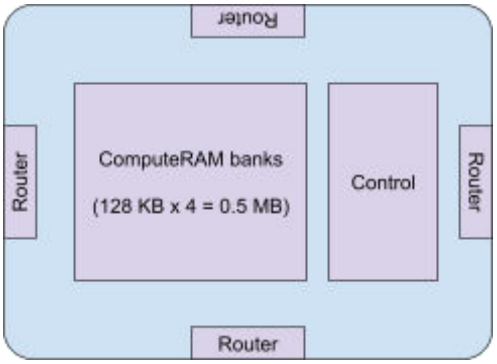
Aimar, A. (2021). Energy-efficient convolutional neural network accelerators for edge intelligence (Doctoral dissertation, University of Zurich). https://www.zora.uzh.ch/id/eprint/209482/

ComputeRAM variant

To insert ComputeRAM in the system described above, we have to decide which parameterization to use, as ComputeRAM is a flexible IP with several degrees of freedom in terms of customization. For this specific analysis, we have chosen the following:

- 128KB instances
- Operating frequency at 1 GHz
- 64-bit interface
- 8 datapaths with 16 multipliers each
- Datapath latency of 1 (1 vector slice/cycle)[3]
- 294k μm^2 per macro (GF22)

The resulting system is architecturally similar to the one described in the baseline, with the only difference being inside each core. Note that the number of MACs per block and the amount of local memory are entirely unchanged. The only difference between the two systems is that we fuse the local memory with the computational engines with ComputeRAM. We engineered the baseline and ComputeRAM variant to make the change transparent to the software. The resulting core is conceptually shown in the following drawing.



Workload Definition

The workloads under analysis are based on standard LLMs, such as Llama3-8B or Llama-3.1-70B. These models are based on large matrix multiplication, for example, for performing the query part of the attention layers or for the feed-forward to be fully connected afterward. The operation we model is a (4096, 8192) x (8192, 8192) matrix multiplication, similar, for example, to a Llama3-70B Query layer with a sequence length of 4096 tokens. In the document, we will use the following notation:

[3] See Synthara's ComputeRAM technical documentation for more information on the parametrization options of the ComputeRAM macro.

- D will refer to the matrix of size (4096, 8192)
- S will refer to the matrix of size (8192, 8192)
- A will refer to the result matrix of size (4096, 8192)

For all three matrices, we are going to use an INT8 datatype. The operation requires:

$$2 \times 4096 \times 8192 \times 8192 = 549.7\ GOPs$$

Dataflow

In our example, we load the matrix S into the processing cores and then stream the D matrix rows to perform the computation. There are 512 SRAM banks in the system, which we replace with 512 CxR instances (for a total of 512 × 8 × 16 = 65k multipliers). Each instance stores 8 columns of S with 8192 entries each. During computing, each of the 4096 vectors composing D is streamed in and broadcasted using the interconnect to all the cores and in the cores to all the CxR instances.

Performance Estimate

Area

When replacing the traditional SRAM and the matrix-multiplication engine with ComputeRAM, we obtain:

$$512 \times 294k\ \mu m^2 = 150.5\ mm^2\ Area$$
$$0\ mm^2\ MAC\ Area\ (integrated\ in\ )$$
$$98.3\ mm^2\ logic\ Area\ (unchanged)$$

$$150.5 + 0 + 98.3 = 248.8\ mm^2\ total\ area$$

Compared to the baseline value of 210.7 mm^2, we obtained a ~20% **area reduction** at the system level despite having the same number of MAC and memory in the system.

Throughput

The baseline system we described is memory-bounded when implemented with or without CxR. For this reason, we assume that the compute time is dominated by the data transfer time from the DRAM to the system. We need to transfer the S and D matrices and then read

out the result (assuming that eventual internal delays for flushing the pipelines are negligible). Given the matrix sizes in the workload section, we have:

- Matrix S: 64 MB
- Matrix D: 32 MB
- Matrix A: 32 MB

Given the INT8 datatype and the reference 8 GB/s interface, we obtain

$$\frac{64\ MB + 32\ MB + 32MB}{8\ GB/s} = 15.62\ ms$$

$$\frac{549.7\ GOPs}{15.62\ ms} = 35.18\ TOP/s$$

Energy Efficiency

The energy cost of the NoC interconnect and control remain constant between the baseline and the CxR-based implementation, while the compute cost and the leakage change. Given the calculated area and computing time, we estimate the leakage for the two systems:

$$6.5\ nW/\mu m^2\ *\ 309.3\ mm^2 = 2.01\ W\ baseline\ leakage$$
$$6.5\ nW/\mu m^2\ *\ 248.8\ mm^2 = 1.61\ W\ CxR\ leakage$$

This shows **a reduction of 17% in leakage** for the CxR-based implementation. For the 16.1 ms compute time, the pJ contribution per operation is

$$\frac{15.62\ ms * 2.01\ W}{549.7G\ ops} = 0.057\ pJ/op\ baseline\ leakage\ contribution$$

$$\frac{15.62\ ms * 1.61\ W}{549.7G\ ops} = 0.046\ pJ/op\ CxR\ leakage\ contribution$$

The energy efficiency of ComputeRAM for S = 4096x8192 and D = 8192 is reported in the following table for GF22. As sparse, ReLU-based LLMs are gaining traction, and we report the numbers for both conventional (0% sparsity) models and sparse models (50%, 75%, 95%).[4]:

[4] While sparsity is not the focus of the present document, Synthara has a large range of technologies to exploit it. Please get in touch with your Synthara representative or with business@synthara.ai for more information.

D Sparsity	Energy Efficiency of ComputeRAM Operations	pJ/op
0%	17.48	0.057
50%	25.90	0.039
75%	33.97	0.029
95%	49.68	0.020

The system-level efficiency becomes, for the baseline with 0% sparsity:

Cost of MAC + Interconnect + Control + Leakage = Total energy consumption

$$0.5 + 0.0197 + 0.0455 + 0.058 = 0.622\,pJ/OP = 1.607\ TOP/s/W$$

$$0.057 + 0.0197 + 0.0455 + 0.047 = 0.168\,pJ/OP = 5.949\,TOP/s/W$$

These observations translate into **370% higher energy efficiency at a chip level.** In cases where sparsity on one of the two matrices can be exploited, the energy efficiency jumps to 6.61, 7.08, and 7.56 TOP/s/W, respectively. As the system is now limited by the efficiency of the interconnect and control, the impact of sparsity is less significant than before. In these cases, ComputeRAM can be configured to support sparse data storage and data transfer, reducing the energy consumption of the other design components.

Overview and Analysis

	Baseline Design	CxR Design	Variation
Memory	64 MB		0%
#Multipliers	64K		0%
Area	309.1 mm2	248.8 mm2	**-20%**
Throughput	35.18 TOP/s		0%
Latency	15.62 ms		0%
Leakage	2.01 W	1.61 W	**-17%**
Energy Efficiency	1.607 TOP/s/W	5.949 TOP/s/W	**+370%**

When using an IP such as ComputeRAM, our customers value system performance metrics significantly over the macro's raw performance. This document is written to demonstrate and quantify these benefits at a conceptual level. We showcase that the integration of ComputeRAM results in considerable area savings, reducing the total area requirement from 309.1 mm² in the baseline to 248.3 mm², equivalent to an approximate 20% reduction in silicon area. This reduction is primarily achieved by embedding computational capabilities directly into the memory array. Such integration eliminates separate MAC unit areas, streamlining the system architecture and freeing space for additional functionalities or further enhancements in chip design.

The most significant improvement is, however, in energy efficiency. By replacing traditional SRAM-based memory and separate MAC arrays with ComputeRAM, the system achieves an overall 370% improvement in energy efficiency, increasing from a baseline of approximately 1.59 TOP/s/W to over 5.75 TOP/s/W. This is possible thanks to the effectiveness of ComputeRAM in minimizing data movement between local memories and MAC units, one of the primary sources of energy consumption in traditional architectures.

Technology Scaling

Our analysis has been carried out using GF22 as the reference foundry node. The choice comes from the availability of data, both internal and external, including silicon data for ComputeRAM. Modern AI accelerators are, however, implemented in more advanced nodes, such as N12/16 or N6/7[5] by TSMC. Synthara engineers regularly work with both technologies and with porting efforts already underway, it is possible to estimate how the system's performance will scale. This is described in the table below, using a similar analysis to that performed above for GF22:

	GF 22		**TSMC 12/16**		**TSMC 6/7**	
	Baseline	**CxR**	**Baseline**	**CxR**	**Baseline**	**CxR**
Area [mm2]	309.3	248.8	206.2	165.8	103.1	82.9
Energy Efficiency [TOP/s/W]	1.607	5.949	2.410[6]	8.923	4.821	17.847

[5]Taiwan Semiconductor Manufacturing Company Limited. (n.d.). N7/N6. Retrieved March 26, 2025, from https://www.tsmc.com/english/dedicatedFoundry/technology/platform_DCE_N7_N6

[6] As validation for our methodology, this number is very close to the reported figure for the Alibaba Hanguang 800 taped out in 12nm. See - **Jonathan Hui.** (2020, October 25). *AI Chips Technology Trends & Landscape (Mobile SoC, Intel, Asian AI Chips, Low-Power Inference Chips)*. Medium. Retrieved from https://jonathan-hui.medium.com/ai-chips-technology-trends-landscape-mobile-soc-intel-asian-ai-chips-low-power-inference-4db701dbe85d

The table gives a first-order approximation of how the two variants would scale with the technology nodes.

DRAM and System-Level Considerations

In our model, data is stored in the external DRAM. In the following table, we report the energy efficiency of the accelerator plus the access cost for the DRAM for our reference workload. We used the following energy cost for the analysis:

- HBM3: 5 pJ/bit
- GDDR7: 10 pJ/bit
- LPDDR5X: 3 pJ/bit

		GF 22		TSMC 6/7	
		Baseline Design	**CxR Design**	**Baseline Design**	**CxR Design**
Energy Efficiency [TOP/s/W]	**No DRAM**	1.607	5.949	4.821	17.847
	HBM3	1.582	5.622	4.604	15.198
	GDDR7	1.558	5.330	4.406	13.234
	LPDDR5X	1.592	5.749	4.689	16.157

As expected, the efficiency dropped in all the use cases. However, the relative impact is very different for the baseline and the CxR variant. For the baseline, the type of memory does not impact the energy efficiency: the energy consumption of the core is so high that any efficiency advantage on the memory side is neglected, and between a power-demanding GDDR7 and an LPDDR5X, the difference is from 3% to 7% depending on the technology node - making the “low power” of LPDDR5X irrelevant. However, thanks to the CxR energy efficiency, the energy efficiency of the memory has a material impact: for example, in the case of TSMC N6, the difference between GDDR7 and LPDDR5X becomes ~20%. As system architects find themselves needing to balance bandwidth, power, and cost, the performance benefit of CxR provides an extra degree of freedom, allowing for more application-specific architectures and enabling workloads, such as LLMs, on low-power devices.

Appendix I: Interconnect Energy Cost

The cost of the interconnect is a major contributor to modern ASICs. Estimating its energy consumption is, however, a non-trivial task, with a vast design space to explore potentially. For this modeling exercise, we decided to start from the state-of-the-art available in the literature and adapt it to our model. The first step is to compute how much data we need to transfer and then normalize it for the number of operations to compute.

For the data transfer, we need to compute the number of hops (jump between NoC nodes) for both the S and the D matrix. In the case of the S matrix, we need to transfer 8192×8192 entries, and each entry goes to a specific compute core. Given our 8x16 grid, on average, the number of hops for each entry is the average Manhattan distance of the system:

$$\frac{8+16-2}{2} = 11\ average\ Manhattan\ distance$$
$$8192 \times 8192 \times 11 = 738M\ number\ of\ hops\ for\ S$$

For matrix D, we need to transfer the 4096x8192 entries, broadcasted to all cores. We assume that each core can receive and forward the data, which leads to 128 hops per matrix entry (each core has to receive the data at least once).

$$128 \times 4096 \times 8192 = 4\,294M\ number\ of\ hops\ for\ S$$

Then, for reading out the result, A:

$$4096 \times 8192 \times 11 = 369M\ number\ of\ hops\ for\ A$$

The number of operations to compute for the analysis is, as specified in the workload section, 550 GOPs, which leads to:

$$\frac{738M + 4\,294M + 369M}{550G} = 0.0098\ hop/op$$

Now that we have the hop/op ratio, we must determine the pJ per hop. From literature[7], for process nodes between 32nm and 12nm, values range between 2.0 pJ/byte/hop and 0.15 pJ/byte/hop, leading to a range between 0.02 pJ/op and 0.0015 pJ/op. For this estimate, we are picking a value of 2.0 pJ/byte/hop, leading to 0.0197 pJ/op.

[7]Fischer, T., Rogenmoser, M., Benz, T., Gürkaynak, F. K., & Benini, L. (2024). FlooNoC: A 645 Gbps/link 0.15 pJ/B/hop open-source NoC with wide physical links and end-to-end AXI4 parallel multi-stream support. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. arXiv:2409.17606. https://doi.org/10.48550/arXiv.2409.17606

Appendix II: Control Energy Cost

In modern ASIC design, the control logic implementation can vary from highly flexible multicore RISC-V systems to highly efficient (but not flexible) FSMs. In our estimate, we model the control cost as a RISC-V core per Compute Core, active all the time. An average RISC-V microcontroller implemented in GF22 consumes around 12.5 pJ/instruction[8], which, for simplicity, we are translating to 12.5 pJ/cycle to avoid modeling effective IPC, multicore solution, or other real-world variables going beyond the purpose of this analysis. Given the compute time extracted in the previous sections and the clock frequency, we can extrapolate that the total energy consumption for the control is

$$15.6\,ms\ *\ 1\,GHz\ *\ 12.5\,pJ/cycle\ *\ 128\,cores\ =\ 25\,760M\,pJ/op$$

$$\frac{25\,760M\,pJ/op}{549.7G\,op} =\ 0.0455\,pJ/op$$

[8]The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-ready 1.7GHz 64bit RISC-V Core in 22nm FDSOI Technology https://arxiv.org/pdf/1904.05442