Synthara

ComputeRAM®

Benchmarking application note #1

Post-physical synthesis gate-level benchmarking of matrix-vector primitive using Arm Cortex-M0 + ComputeRAM

Version 1.0
March 27, 2024

About Synthara

Synthara is a Swiss company founded in 2019 by Dr. Manu V Nair and Dr. Alessandro Aimar, based on their work at the Institute of Neuroinformatics, ETH Zürich and University of Zürich. It has raised over USD 15 million in funding from EU and Swiss grants as well as venture capital funds such as HTGF, Excellis, and DeepIE. Synthara's mission is to enable a future where artificial intelligence is ubiquitous, by developing next-generation processing systems that are orders of magnitude faster and energy efficient while ensuring they are easy to use and integrate for its customers.

Copyright notice and proprietary information

Disclaimer

Synthara AG is committed to upholding the highest engineering standards and reserves the right to update reported metrics with more accurate ones to comply with its quality commitment. Synthara AG and its licensors make no warranty of any kind, express or implied, with regard to this material, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Trademarks

Synthara, ComputeRAM, and other Synthara product names are trademarks of Synthara AG. All other product or company names may be trademarks of their respective owners.

Third-party links

Any links to third-party websites included in this document are for your convenience only. Synthara does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Contact

Synthara AG
www.synthara.ai
business@synthara.ai

Table of contents

Revision history

Date	Version	Changes
27-Mar-2024	1.0	Initial Release

Related documents

- ComputeRAM Product Brief
- ComputeRAM Datasheet
- Arm Cortex-M0 Processor Datasheet

1. Introduction

Synthara's ComputeRAM is an SRAM macro with integrated computing capabilities, which unlocks improvements of up to 130x in speed and 150x in energy efficiency for linear algebra workloads in MCU-based edge applications. ComputeRAM is designed as a drop-in SRAM replacement with only a small area overhead. By making general-purpose chips faster and more efficient, ComputeRAM eliminates the need for dedicated AI and DSP accelerators in edge applications that demand flexibility, low power consumption, and high performance.

ComputeRAM is accompanied by a dedicated SDK (software development kit) which allows its users to seamlessly develop and deploy software on ComputeRAM-based systems. It includes a collection of low-level linear algebra primitives, optimized neural network layers and models, as well as compilation, simulation, and emulation tools. At a fundamental level, the ComputeRAM SDK libraries break down linear algebra operations into a series of primitives, whose computation is accelerated by ComputeRAM. While this occurs, overall control and coordination are managed by the CPU. This approach gives programmers complete flexibility to develop new models while also benefiting from the acceleration capabilities of ComputeRAM.

2. Application note purpose

This application note is the first of a series on the benchmarking of ComputeRAM. It explores the matrix-vector multiplication (MVM) operation, which is the lowest-level linear algebra primitive accelerated by ComputeRAM. Understanding the performance gains unlocked for this operation is important, as it forms the basis for numerous other algorithms, ranging from signal processing to AI and machine learning. Thanks to ComputeRAM's design flexibility, these benefits can be easily quantified by simply replacing the on-chip memory of a reference MCU (microcontroller unit) with ComputeRAM. This application note describes the test MCU's architecture, the adopted simulation and measurement methodology, and the obtained results.

3. Methodology

3.1. System architecture

Benchmarking was conducted using a series of post-physical synthesis gate-level simulations for a reference MCU hardware architecture (Figure 1). This is composed of an Arm Cortex-M0 processor, an AHB interconnect provided by Arm, and a memory array. The Arm Cortex-M0 was chosen as it is widely used across edge devices and embedded systems. The memory array consisted of 256 KB, partitioned into 128 banks comprising 512 entries of 32-bit width. ComputeRAM was integrated into the architecture as a drop-in replacement for the SRAM.

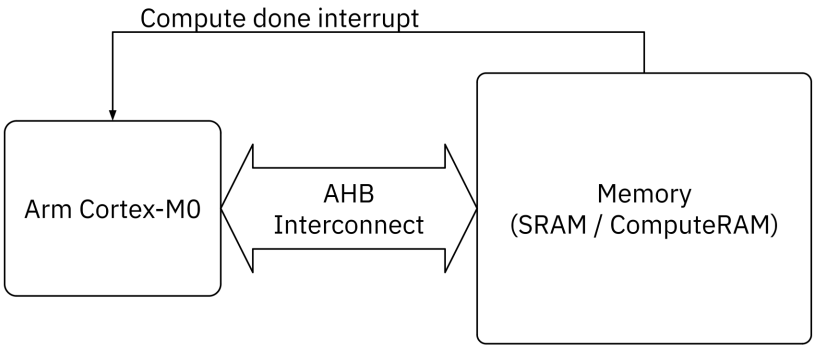

***Figure 1: Testing MCU hardware configuration**, composed of an Arm Cortex-M0, the AHB interconnect and a memory array.*

3.2. Hardware assumptions

The 32-cycle iterative multiplier configuration of the Arm-Cortex M0, as defined in the *Arm Cortex-M0 Processor Datasheet*, was used in this work. The ComputeRAM configuration definition and key performance indicators (KPIs) are taken from the *ComputeRAM Datasheet.* Figure 2 summarizes the latency and energy efficiency associated with the compute operations performed by ComputeRAM for different matrix sizes at INT8 precision. The latency scales linearly with the number of matrix rows, while the compute efficiency is maximized when operating with larger matrices.

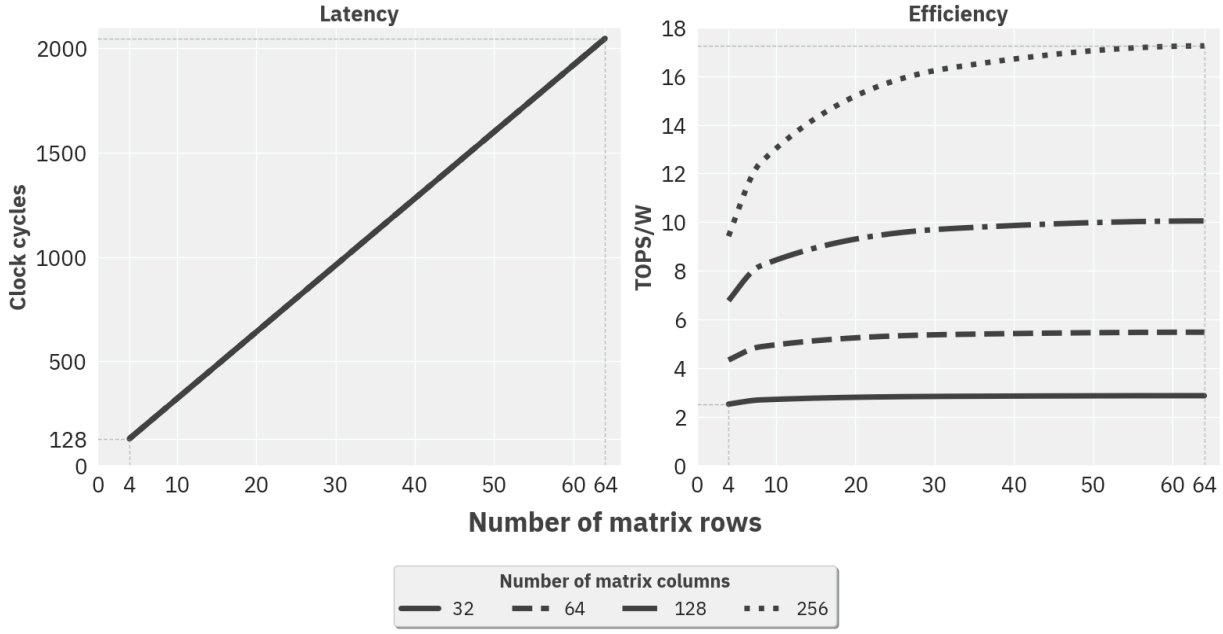

***Figure 2: Latency and energy efficiency** corresponding to ComputeRAM's compute operations, assuming 100% input bus utilization and INT8 input and weight precision. The latency is constant across the number of matrix columns as ComputeRAM can process up to 256 columns in parallel.*

3.3 Simulation setup

The design, including the components of ComputeRAM, was synthesized using the GF22FDX process, utilizing 7.5T libraries with 0.65V cells. The timing analysis confirmed a stable operation at 250 MHz under worst-case corner conditions at 125°C and 0.59V. Power estimates were performed by the synthesis tool using post-synthesis switching activity in typical conditions at 25°C.

Testing was split into two phases:

1. **Data Load Phase:** In the SRAM configuration, matrix data is loaded from the external memory into the CPU, followed by the writing of the result to memory once the computations are complete. In the ComputeRAM configuration, data is written from memory to the correct address space used for computations by ComputeRAM. All described operations were carried out using standard read/write instructions, and have non-negligeable energy and latency costs.
2. **Compute Phase:** In the SRAM configuration, the CPU ALU carries out the workload. In the ComputeRAM configuration, the workload is carried out by ComputeRAM instead of the CPU ALU.

The simulation produced latency and energy consumption data. The latency was calculated as the number of clock cycles that the processor required to complete the operation. The energy consumption, including leakage, was measured using a post-synthesis netlist by employing the following steps:

1. A physical-aware synthesis was performed using Cadence Genus, incorporating an early floorplan and clock tree estimation;
2. The Verilog netlist and SDF for subsequent gate-level simulations were extracted;
3. A gate-level simulation was then performed, followed by the saving of the waveforms into a VCD file which captures the activity of all nodes in the system over time;
4. The VCD was reloaded into Genus to produce detailed power reports;
5. The total energy from the power reports was extracted;

During this process, Genus was configured for maximum power optimization starting from the GlobalFoundries reference flow, including the provision of RTL-level switching activity at synthesis start for power-driven synthesis.

3.4. Benchmark description

Several conditions were tested, with matrix dimensions being varied to accommodate the full spectrum of matrices that can fit into ComputeRAM. The number of rows varied between 4 and 64, while the column sizes varied between 32 and 256 elements. Matrices and vectors were uniformly randomized to ensure a realistic switching activity within the system registers. The test was carried out using INT8 data, with the accumulation performed in full precision (32 bits). The result was then shifted and truncated back to 8-bits before being stored back into SRAM. This is representative of the inner product kernel typically found in neural network inference workloads.

4. Results

The system-level improvement factors attributed to the use of ComputeRAM are presented in Figure 3. When ComputeRAM is used instead of a traditional SRAM, improvements between 11.74x and 139.72x can be achieved in terms of latency, and between 12.23x and 158.7x in terms of energy efficiency.

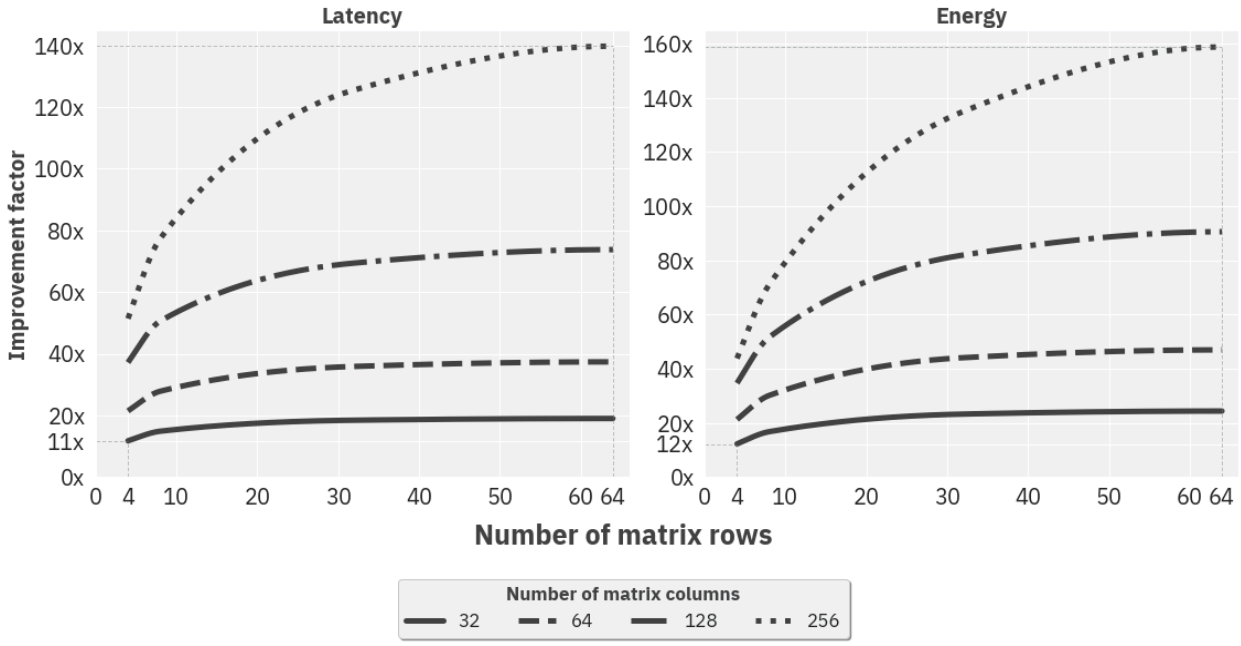

Figure 3: Gate-level simulation MVM improvement factors *achieved by the Arm Cortex-M0 with ComputeRAM over the SRAM configuration, calculated as the ratios between the two.*

5. Discussion

This application note showcased how, by replacing the on-chip memory of a simple reference MCU with ComputeRAM, important performance improvements can be unlocked. The 140x gain in speed would enable even a basic processor, like the Arm Cortex-M0, to handle heavy workloads that would otherwise take an order of magnitude longer to process. Meanwhile, the 159x reduction in energy consumption leads to not only lower operational costs, but also expands the variety of applications that can be efficiently deployed on resource-limited devices. These observations establish ComputeRAM as a potential solution that eliminates the reliance on dedicated AI or DSP accelerators for edge applications.

While the peak ComputeRAM efficiency at a macro level is 17.25 TOPS/W, achieving such performance at system level depends on the system architecture and workload partitioning algorithms. Future application notes will explore these topics in greater detail in applications ranging from complex signal processing to AI models.