Doctoral Dissertation
Doctoral Program in Electrical, Electronics and Communications Engineering (36thcycle)

From Control to Compute: Modular CPU design and Programmable Near-Memory Solutions for Edge Applications

By

Michele CAON
******

Supervisor(s):
Prof. Maurizio MARTINA, Supervisor
Prof. Guido MASERA, Co-Supervisor

Politecnico di Torino
2024

Summary

Introduction

This thesis addresses the challenges of modern edge computing, driven by the rise of data-centric applications like Artificial Intelligence (AI) and Internet of Things (IoT), which expose the inefficiencies of traditional von Neumann architectures. To overcome these limitations, it proposes innovative solutions: a programmable near-memory computing architecture, NM-Carus, that bridges flexibility and energy efficiency and a dynamically scheduled RISC-V Central Processing Unit (CPU) core, LEN5, optimized for instruction-level parallelism and coprocessor integration. Together, these contributions enhance computational throughput and resource utilization, laying the foundation for scalable, energy-efficient edge devices.

NM-Carus: A Near-Memory Computing Architecture for Edge Applications

This chapter introduces NM-Carus, a novel Near-Memory Computing (NMC) architecture, depicted in Fig. 1, designed to address the inefficiencies of traditional Compute-In-Memory (CIM) systems and enhance performance, energy efficiency, and programmability in edge devices. By adopting a fully digital design leveraging standard Static Random-Access Memory (SRAM) macros and integrating a programmable RISC-V-based controller with a scalable Vector Processing Unit (VPU), NM-Carus overcomes the limitations of existing In-Memory Computing (IMC) and NMC solutions.

Its key contributions include:

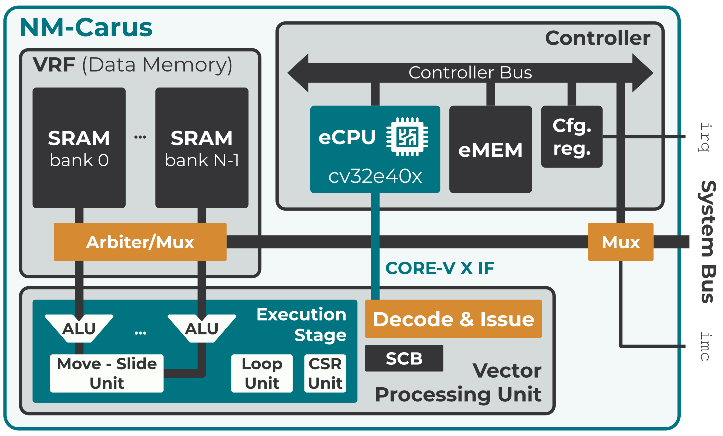
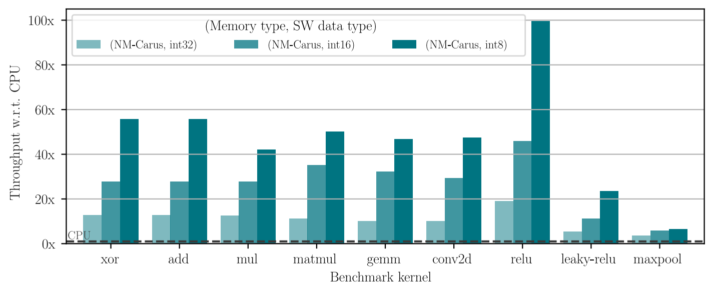


Figure 1: Top-level architecture of the NM-Carus NMC system, highlighting its SRAM-based Vector Register File (VRF), RISC-V-based embedded controller, and VPU.

- Seamless integration into existing Microcontroller Unit (MCU) systems as a direct replacement for conventional SRAM banks, enabling easy adoption without major redesigns.
- High energy efficiency and throughput for data-intensive workloads, demonstrated through extensive post-layout simulations and benchmarking.
- Enhanced programmability via a custom `xvnmc` vector instruction set extension, supporting a wide range of operations and bitwidths with minimal overhead.

Figure 2 shows the throughput improvements of the NMC-enhanced X-HEEP MCU compared with its CPU-only version.



(a) Throughput improvement.

Figure 2: Throughput improvement (a) of the NMC-enhanced X-HEEP MCU compared with its CPU-only version.

Summary of the LEN5 Microprocessor Contribution

The LEN5 microprocessor addresses the limitations of in-order CPUs in edge computing by leveraging Out-of-Order (OoO) execution to enhance performance and resource utilization. Its modular and scalable design enables efficient integration of tightly coupled coprocessors, making it suitable for diverse applications.

Key Features:

- **Dynamic Scheduling:** An enhanced Tomasulo's algorithm enables out-of-order dispatch and execution using reservation stations (RSs), a reorder buffer (ROB), and a common data bus (CDB).
- **Speculative Execution:** A *gshare* branch predictor, branch target buffer (BTB), and return address stack (RAS) reduce control hazards.
- **Latency Hiding:** Supports tightly coupled coprocessors, with store-to-load forwarding and out-of-order commit to maximize throughput.
- **Configurability:** Provides adjustable ROB, RSs, and execution units to balance performance and area.

Physical Implementation: LEN5 was implemented in TSMC 65 nm low-power CMOS technology, with three configurations: (1) **Max Performance:** 32-entry ROB, pipelined multiplier. (2) **Balanced Performance:** Smaller data structures, no divider. (3) **Minimal Area:** Omitted multiplier/divider for area efficiency. Compared to CV32E40P, LEN5 shows a 11.3 % area overhead in X-HEEP MCU while offering a 20 % higher clock frequency and significantly higher Instructions Per Cycle (IPC) when coupled with long-latency coprocessors. Figure 3 shows the IPC comparison between LEN5 and CV32E40P over the Embench suite.

Future Work: The ARCANE In-Cache Computing System

The Adaptive RISC-V Cache Architecture for Near-Memory Extensions (ARCANE) system, shown in Fig. 4, represents the evolution of near-memory computing by integrating NM-Carus instances directly into the data cache of a system, transforming it into a tightly coupled compute engine. This approach eliminates explicit memory management by offloading synchronization and data movement to an augmented cache controller, enabling efficient instruction-based programming for complex operations.

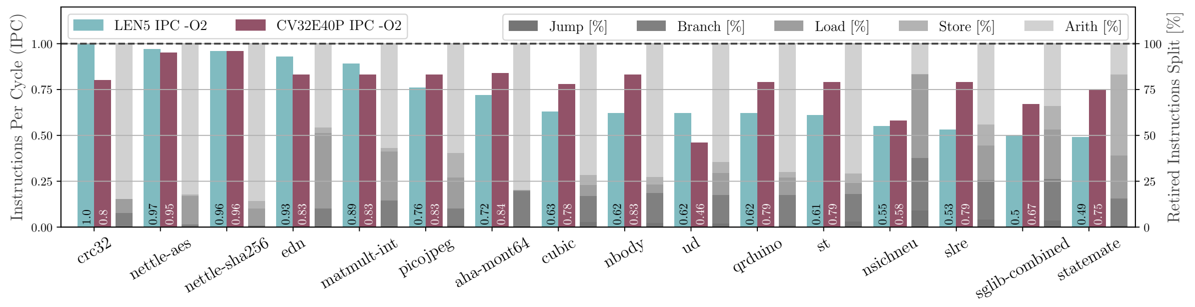


Figure 3: IPC comparison with CV32E40P over the Embench suite (colored bars) and executed instruction composition (grayscale).

Key Features:

- **Programmable Cache Controller:** Manages synchronization, handles vector instruction offloading, and reduces data movement.
- **Matrix-Based Instruction Set Architecture (ISA):** Implements arbitrary software-defined instructions to execute complex workloads on the NMC-based cache array. The `xmr` instructions, together with a hardware Address Table (AT), handle hazards between bus transactions and offloaded instructions, ensuring implicit synchronization with the host CPU.
- **Broadcasting and Parallelism:** Supports tiling and distributing vector instructions across multiple NM-Carus instances, improving scalability for large operations.

Integration with LEN5: Future integration of the LEN5 out-of-order CPU with ARCANE aims to exploit the Instruction-Level Parallelism (ILP) opportunities exposed by the long-latency offloaded instructions, allowing concurrent execution of scalar instructions and in-cache computing operations.

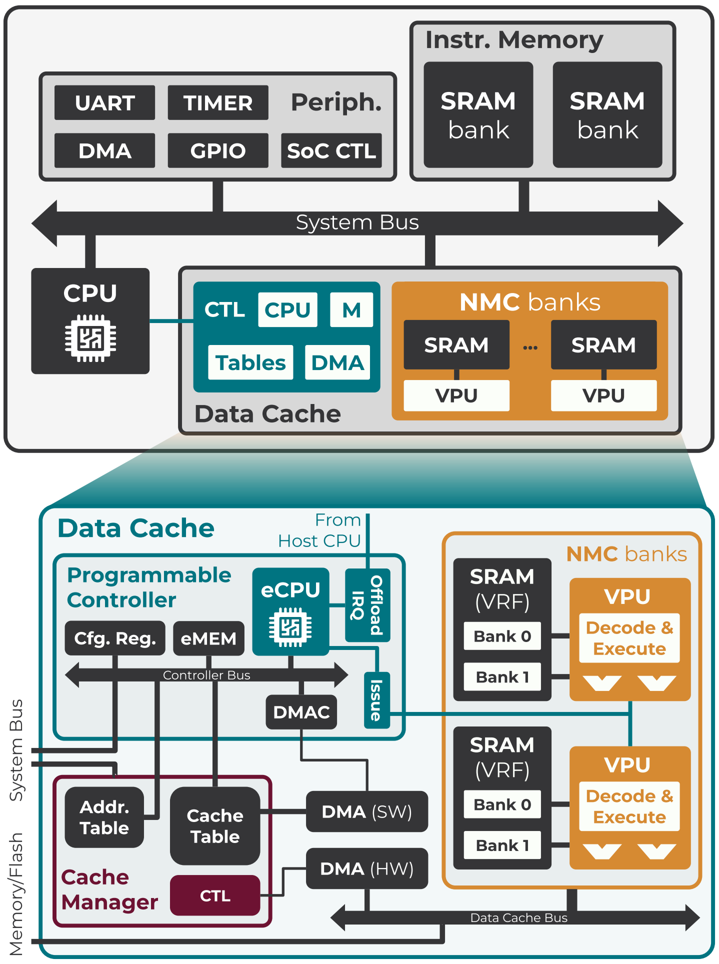


Figure 4: Top-level block diagram of the ARCANE system.