# Abstract

Artificial Intelligence (AI) and Machine Learning (ML) are increasingly central to automotive perception and decision-making, where cameras, radars, lidars, and other on-board sensors generate large amounts of data that must be processed under strict constraints on latency, power, cost, reliability, and implementation complexity. In this context, the computational workload is intrinsically heterogeneous: conventional digital signal processing remains essential in radar and lidar front-ends, while ML models are increasingly adopted for perception, classification, sensor fusion, and higher-level reasoning. As a result, efficient automotive computing cannot be addressed by optimizing a single accelerator in isolation, but requires a broader architectural perspective that considers workload structure, hardware specialization, host–accelerator coupling, memory organization, data movement, and system-level integration.

This dissertation investigates hardware architectures for automotive sensor processing and ML across multiple levels of specialization and integration, using RISC-V-based platforms as an open and extensible substrate for architectural exploration. The work first frames automotive perception as a heterogeneous computing problem, analyzing the computational patterns that arise from radar, lidar, and camera-based processing pipelines. These include dense linear algebra, nonlinear functions, reductions, FFT-based signal processing, filtering, thresholding, data movement, and irregular control-oriented stages. This kernel-oriented view motivates the exploration of different architectural points rather than a single monolithic acceleration strategy.

The first part of the architectural contribution focuses on close-to-the-core acceleration. Two tightly coupled coprocessors are presented as complementary case studies. TOXOS is a configurable coprocessor based on the unified CORDIC algorithm, designed to accelerate nonlinear functions such as trigonometric, hy-

perbolic, exponential, and related operators. By adopting a global floating-point strategy around a fixed-point CORDIC core, TOXOS provides a compact implementation while preserving a floating-point interface. Its integration through a RISC-V custom extension and the CV-X-IF interface demonstrates how narrow but recurrent computational bottlenecks can benefit from strong specialization when invocation overhead is kept under control.

RACE explores a different point in the close-to-core design space. Instead of targeting a single operator family, it implements a compact reconfigurable arithmetic fabric exposed as an instruction-visible coprocessor. The design investigates how a small CGRA-like structure can accelerate short arithmetic dataflows typical of embedded ML-oriented kernels. The results highlight that reconfigurability alone is not sufficient: practical speedup depends strongly on the efficiency of the host interface, operand delivery, result collection, and configuration overhead. In this sense, TOXOS and RACE jointly show that the choice between specialization and reconfigurability depends on the stability and breadth of the target workload, as well as on the coupling mechanism used to expose the accelerator to software.

The second part of the dissertation moves from the processor core to system-level and far-from-core architectural solutions. In this context, a study on scalability of near-memory computing architectures investigates how multi-bank compute-capable memories scale in low-power SoCs. The results indicate that area overhead, critical path, energy, and performance all depend heavily on memory capacity, lane count, bank organization, and how data orchestration is handled by software. Moving to the system level, the integration of multiple near-memory instances shows that meaningful acceleration requires the software stack to explicitly manage tiling, buffering, and parallelism across instances.

The dissertation then introduces a parametric CGRA model for design-space exploration. The model includes a configurable array of processing elements, a banked internal memory, row-level DMA engines, configurable routing, and a host-controlled execution model. This framework is used to study how array dimensions, processing-element latency, memory organization, routing flexibility, and data-movement mechanisms affect kernel execution. Building on this model, an automotive-oriented CGRA instance is considered through the mapping of representative kernels such as matrix multiplication and FFT. The case study

shows how concrete workload requirements can guide architectural choices such as feed-forward connectivity, streaming support, and hardware-loop execution.

In conclusion, HEEPatia and polHEEPo are presented as two silicon prototypes of RISC-V based systems on chip integrating heterogeneous accelerators.

Overall, the thesis argues that efficient embedded automotive computing is best addressed through heterogeneous architectures rather than through a single dominant accelerator paradigm. The main conclusion is that the appropriate architectural solution depends on the target kernel, on the cost of data movement, on the required degree of programmability, and on the surrounding system. Specialized coprocessors, reconfigurable fabrics, near-memory computing, and system-level accelerators are thus not rivals in any absolute sense. They are complementary design options, each of which delivers its best when matched to the right computational and integration context.