

Multi-Partner Project: CeCaS Accelerator Design for Efficient Supercomputing in Automotive Systems

Annina Gutermann¹, Alexey Serdyuk¹, Fabian Lesniak¹, Julian Hofer¹, Hella Toto Kiesa¹,
 Tanja Harbaum¹, Juergen Becker¹, Brian Pachideh^{1,2}, Sven Nitzsche^{1,2}, Moritz Neher^{1,2,3},
 Carmen Weigelt^{1,3}, Jann Krausse^{1,3}, Victor Pazmino², Klaus Knobloch³,
 Lukas Groth⁴, Andrija Nešković⁴, Saleh Mulhem⁴, Mladen Berekovic⁴

¹Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Email: {gutermann, alexey.serdyuk, lesniak, hella.kiesa, harbaum, becker}@kit.edu

²FZI Research Center for Information Technology, Karlsruhe, Germany

Email: {pachideh, nitzsche, neher, pazmino}@fzi.de

³Infineon Technologies, Dresden, Germany

Email: {carmen.weigelt, jann.krausse, klaus.knobloch}@infineon.com

⁴Institute of Computer Engineering, Universität zu Lübeck, Lübeck, Germany

Email: {l.groth, andrija.neskovic, saleh.mulhem, mladen.berekovic}@uni-luebeck.de

Abstract—Modern vehicles integrate an increasing amount of computational functionality, driven by the growing complexity of in-vehicle applications. At the same time, automotive system architectures are becoming more centralized, requiring powerful HPC platforms at the core. These platforms must deliver the performance needed for ADAS, AI, and autonomous driving, while also meeting stringent energy efficiency and safety requirements.

The CeCaS project addresses these challenges across a wide range of topics and domains of expertise, including processor design in advanced FinFET technology, the transformation of the E/E architecture, and advanced packaging for automotive supercomputing platforms. Within CeCaS, our work focuses on application-specific accelerator design to enable efficient processing of compute-intensive workloads. In this paper, we present our contributions in this area, including the design of hardware accelerators for both conventional and neuromorphic AI workloads, the development and evaluation of representative AI benchmarks, and the use of virtual platforms for early design-space exploration and hardware/software co-design.

Index Terms—Hardware Accelerators, Hardware Software Co-Design, Automotive, Benchmarking, Neuromorphic

I. INTRODUCTION

Modern vehicles are becoming increasingly automated and connected. They must process large volumes of sensor data and execute complex algorithms, many of which involve Artificial Intelligence (AI) workloads. To manage these demands and the growing complexity of software and wiring, the industry is moving toward centralized architectures built around a powerful High-Performance Computer (HPC). Such platforms need to deliver near-supercomputer performance while meeting strict constraints on energy efficiency, reliability, and automotive qualification.

The nationally funded project *Central Car Server* (CeCaS) [1] addresses these challenges by developing an automotive supercomputing platform based on new automotive-qualified high-performance processors in FinFET technology. The platform is supported by application-specific accelerators

and an adaptive automotive software stack for automated and connected vehicles. The project thus covers a wide range of research topics, including processor and accelerator design, hardware/software co-design of the scalable Central Car Server platform, and supporting work in vehicle technology, reliability, and verification infrastructure. The full consortium consists of 29 partners, including 10 academic institutions and 19 industrial companies, each contributing domain-specific expertise.

As the complete scope of CeCaS goes beyond this paper, we focus here on one central aspect: the design of application-specific hardware accelerators for automotive use cases, especially for AI workloads, which are becoming increasingly important in driver assistance and autonomous driving applications. Graphics Processing Units (GPUs), which dominate many AI applications, are not suitable here due to their high energy demand and limited operational domain for consumer grade use. Instead, heterogeneous architectures that combine automotive-grade processors with dedicated accelerators provide a more suitable alternative.

Our contributions to this field are threefold:

- We developed an open-sourced design of a systolic-array based accelerator *FlexXNNGine* to accelerate data-intensive workloads
- We contributed to *YANA*, an open-source neuromorphic accelerator framework for Spiking Neural Networks (SNNs), as a complement to conventional AI engines
- We enhanced system-level heterogeneous platform design through AI benchmarking, virtual prototyping, and workload partitioning across components

In this work, four partners were involved.

The FlexXNNGine was developed by the Karlsruhe Institute of Technology (KIT). Section II introduces the accelerator, including its memory interface design and safety mechanisms compliant with ISO 8800.

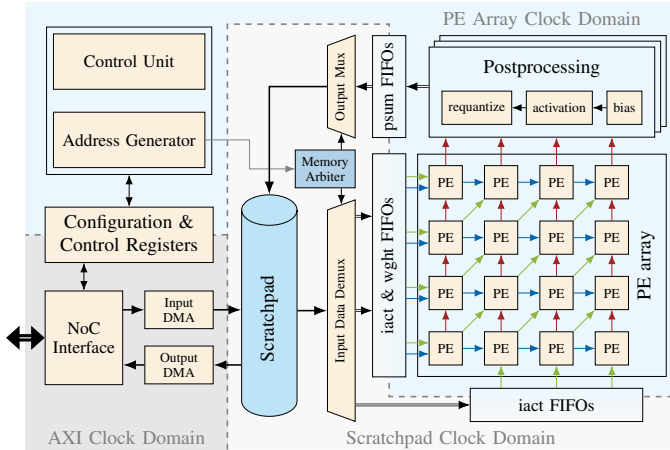


Fig. 1: Overview of the FleXNNengine architecture. In this example a 4×4 Processing Element (PE) array is shown with coloured arrows showing data movement directions. The architecture encompasses several clock domains to enhance memory bandwidth. [3]

FZI Research Center for Information Technology and Infineon Technologies both contribute to the YANA framework, which couples an Field-Programmable Gate Array (FPGA)-based accelerator with a hardware-aware framework for training, optimization and deployment of SNNs [2] and is described in section III.

Section IV addresses system-level integration, covering benchmarking of AI workloads (KIT), RISC-V based virtual prototyping including safety/security evaluations (UzL) and inference partitioning across heterogeneous architectures (KIT).

II. MULTI-PURPOSE ACCELERATOR

In many Neural Networks (NNs), convolutional layers dominate the computational demand. Due to the inherently memory-bound nature of convolution operations, maximizing data reuse is key to reducing memory bandwidth demands and energy consumption in Convolutional Neural Network (CNN) accelerators. Systolic arrays, particularly when combined with the energy-efficient Row-Stationary (RS) dataflow [4], enhance data reuse by storing subsets of input activations, weights, and partial sums locally within the PEs.

However, existing RS-based accelerator designs are not publicly available and often lack detailed implementation information. We therefore developed an in-house implementation and open-sourced our design as *FleXNNengine* [3], [5].

A. Design of a flexible systolic array-based accelerator

Figure 1 shows the architecture of our proposed accelerator. The design primarily targets the acceleration of CNNs, with a focus on vision processing workloads, but can be reconfigured to other workloads, such as accelerating transformer architectures.

Building upon a HDL model, we construct a simulation framework that enables detailed workload characterization and

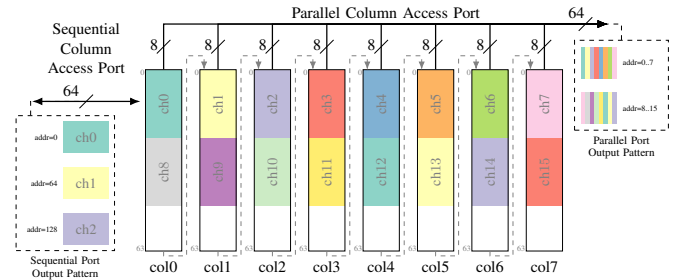


Fig. 2: Custom column-based scratchpad memory for on-the-fly data transposition. Data can be accessed linearly or from all columns in parallel depending using two different ports.

architectural exploration. The design is highly parameterizable, allowing adaptation to a wide range of deployment scenarios.

We analyze various architectural parameters and evaluate their impact on performance. Additionally, we introduce an extended RS dataflow optimized for parallel execution of convolutions with a large number of channels [3]. Our optimizations achieve up to $2.3\times$ performance improvement on convolutional layers of representative neural networks. A prototype implementation comprising 70 PEs on a Xilinx UltraScale+ ZCU104 FPGA demonstrates a peak performance of 4.012 Giga Operations per Second (GOPS) at 100 MHz.

Transformer acceleration relies on deploying General Matrix Multiply (GEMM) operations. Since the reuse advantages of the RS dataflows do not extend to matrix multiplications, we instead employ an output-stationary dataflow with similar data movement as our extended RS dataflow. This reduces the required local buffer size per PE, allowing the freed capacity to be used as additional scratchpad memory for data preloading, which is well suited for transformer networks due to their large parameter sets.

B. Custom Scratchpad Memory Design

Integrating specialized accelerators into System-on-Chips (SoCs) presents significant interfacing challenges, particularly concerning memory interconnect and data format mismatches, which are often underestimated. For instance, the RS dataflow, used for convolutions in our accelerator, requires a Height-Width-Channel (HWC) data layout for inputs, while outputs are generated in a Channel-Height-Width (CHW) format¹. This mismatch necessitates data reordering between consecutive convolutional layers, as the output of the previous layer can not be fed back into the accelerator without conversion. As deep learning frameworks commonly use the CHW layout, support for this format is the preferred solution, especially regarding the first and last layers. Creating an HWC stream from CHW data stored in the scratchpad memory requires single-byte memory access. This would limit bandwidth significantly, leading to either unrealistic clocking requirements on the scratchpad interface or low utilization of the PE array.

¹The HWC layout is similar the storage format of RGB bitmap images, i.e. for each pixel, the values of all channels are stored next to each other. For CHW, on the contrary, channels constitute the outermost loop variable, iterating over consecutive blocks of $H \times W$ elements.

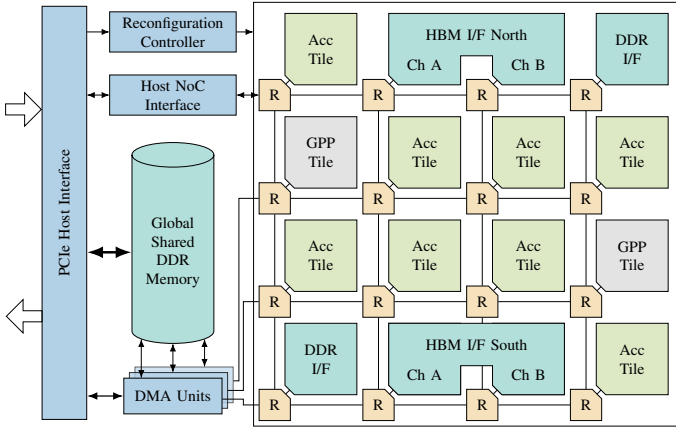


Fig. 3: A NoC-based accelerator with 16 tiles, including two HBM and DRAM interfaces each. Accelerator (Acc) and GPP tiles are reconfigurable, whereas performance-critical memory interfaces and interconnect are hard logic to achieve peak performance [6].

To address this performance bottleneck, we implemented a custom dual-port scratchpad memory architecture that supports efficient multi-word access and implicit data reordering without run-time overhead. The memory is organized as a set of multiple equally sized columns, which can be accessed both sequentially and in parallel, as shown in figure 2. The sequential access port allows both the host system and the accelerator to write data in the conventional CHW layout. The parallel access port accesses all memory columns in parallel, directly providing the HWC data stream required by the PE array. This is achieved by storing each channel with an offset equal to the memory’s column size, which aligns data for parallel retrieval. Data can be efficiently transferred between DRAM and our custom scratchpad using a DMA engine in scatter/gather mode, which applies the required column offsets automatically.

Our design eliminates explicit data reordering while significantly boosting memory throughput. As a result, the required scratchpad clock frequency is reduced to just 1.5x that of the PE array, while maintaining the required bandwidth to prevent stalls. The memory architecture is fully parametric, allowing for customization of the number of columns, address width, and element size.

C. Scalable NoC-based Accelerator Platform

To further increase performance and flexibility beyond the previously introduced accelerator, the architecture is scaled up into a Network-on-Chip (NoC)-based platform [6]. This approach increases computational throughput and enhances flexibility, as the NoC can be configured heterogeneously enabling efficient acceleration of various workloads.

The proposed architecture combines a fine-grained interleaving of fixed logic in the high-performance data path with an FPGA-based reconfigurable fabric for algorithm-specific components. As illustrated in figure 3, the scalable accelerator architecture is realized by integrating a NoC that enables efficient data transfer between multiple accelerator tiles and mem-

ory components. Each tile is implemented as reconfigurable logic, allowing dynamic adaptation at runtime to meet the requirements of the given applications. This reconfigurability also facilitates component isolation, enabling flexible resource management and improved fault containment. Corresponding software interfaces have been developed to support system integration and runtime control.

In addition to compute tiles, dedicated memory tiles provide access to both DDR4 and HBM memory. In the current prototype, a DDR tile offers 4 GiB of memory with a 64-bit bus interface, while up to 16 HBM tiles, each equipped with 256 MiB of memory and a 256-bit interface, can be instantiated to meet the demands of bandwidth-intensive workloads. The complete system has been implemented on a Xilinx VCU128 FPGA prototyping platform, where a representative application has been demonstrated.

Performance evaluation of a single compute tile shows a spatial efficiency of up to 90% at 6.28 GOPS. A 5x5 mesh prototype achieved a performance gain of $\times 19.75$ with 20 processing tiles, demonstrating excellent scalability. Overall, the architecture provides a flexible and extensible foundation for implementing a wide range of algorithms that can be dynamically reconfigured at runtime.

D. Safety Mechanisms

Considering functional safety is crucial when deploying AI models in safety-critical applications, as it has been shown that one single hardware fault during AI inference can ultimately lead to wrong and possibly hazardous predictions [7]. During the project period, the new international standard ISO/PAS 8800 - *Road vehicles — Safety and artificial intelligence* has been published, marking an important milestone for enabling AI safety in autonomous driving. Motivated by the requirements derived from this standard, the FleXNNengine accelerator has been extended to support architectural redundancy patterns. These redundancy patterns help to cope not only with transient hardware faults but also with functional insufficiencies such as out-of-distribution data and distributional shifts. One way to generate these redundancy patterns is the transformation of deterministic AI models into sampling-based approximate Bayesian Neural Networks (BayNNs).

The core idea is that by incorporating non-deterministic, i.e. stochastic layers (e.g. Dropout) in the AI model and generating multiple Monte-Carlo samples, the uncertainty of an AI model can be represented [8]. Crucially for AI safety, highly uncertain predictions can be detected and ignored during subsequent processing. While initially designed to address functional insufficiencies, the work in [9] demonstrates that approximate BayNNs also offer protection against random hardware faults. This is because the overall fault-tolerance increases and because a single-bit weight fault only produces a false prediction when the prediction’s inherent uncertainty is already high.

However, enabling sampling-based BayNN processing comes at the cost of high computational complexity. To ease this cost, we provide inherent hardware support in the FleXNNengine accelerator, thereby reducing the number of off-chip memory accesses. Flexible stochastic hardware modules are integrated

in either the activation or weight data paths of the accelerator to enable this support. These modules can manipulate incoming data by applying Bernoulli noise or multiplicative/additive Gaussian noise, allowing the implementation of several approximate BayNN concepts, such as Monte-Carlo Dropout [8], Monte-Carlo DropConnect [10], or Variational Inference [11]. These extensions to the FleXNNengine are openly available and published as the BayNNengine [12]. As a result, the CeCaS project provides a novel safety mechanism that meets the requirements of emerging safety standards in current and future accelerator designs.

III. NEUROMORPHIC ACCELERATOR DESIGN

Beyond conventional AI accelerators, neuromorphic computing with SNNs offers an alternative paradigm that computes only on events, achieving activity-proportional latency, low energy, and reduced data movement [13]–[15]. Such characteristics make SNNs well suited for near-sensor pre-processing and always-on functions to evaluate sparse vision streams from automotive radar [16], [17], LiDAR [18], event cameras (DVS) [19], [20] or other sensor data and temporal tasks such as motor control [21] or trajectory planning [22].

In CeCaS, we therefore study SNN accelerators as a complement to conventional AI engines, centered on the open-source YANA framework [2]. YANA comprises a flexible FPGA-based hardware accelerator and a software stack for model training, optimization and deployment. The software stack is specifically designed to integrate seamlessly with popular open-source tools, lowering the barrier to entry for developers [2].

A. Accelerator Hardware

The hardware accelerator is a digital, FPGA-based architecture designed for event-by-event SNN processing. Its key properties are:

- **Event-Driven Processing:** The hardware is inherently event-driven, processing data only when spike events occur. This allows it to fully exploit both dynamic temporal sparsity (infrequent activations) and static spatial sparsity (pruned or sparse connections) for significant reductions in computational latency.
- **Arbitrary Graph Topologies:** Leveraging a near-memory computing paradigm with point-to-point connection encoding, YANA supports arbitrary and highly recurrent SNN graphs, moving beyond the rigid, layered structures common in conventional Deep Learning (DL) accelerators.
- **Resource Efficiency:** The architecture uses a time-multiplexed processing pipeline, allowing multiple neurons and synapses to share the same physical logic, which reduces FPGA resource utilization.
- **Neuron Model Implementation:** It implements a quantized Leaky Integrate-and-Fire (LIF) neuron model where the leak dynamics are efficiently calculated using a Look-Up Table (LUT), a feature that can be mirrored in software for hardware-aware training [23].

The first fully integrated YANA prototype is implemented on the AMD Kria KR260, which features a Zynq UltraScale+ SoC-FPGA. It combines an ARM processor subsystem, which runs

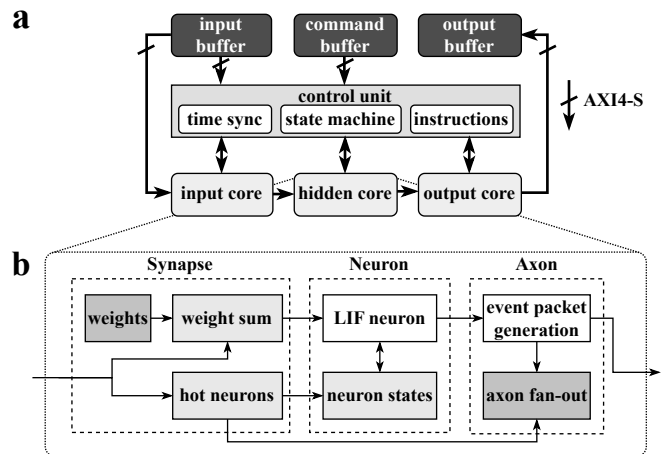


Fig. 4: **a** shows a block diagram of the YANA accelerator with a single hidden core. The pipelined core microarchitecture is depicted in **b**.

the software environment, alongside a programmable FPGA fabric, where the custom hardware accelerator is implemented. Communication between the software and hardware is managed via AXI4 bus interfaces, allowing the PS to configure the accelerator, stream input data, initiate inference, and retrieve results from the PL. The complete deployment also includes dedicated hardware modules for handling input event multicasting and integrating output spike counts (Figure 4).

B. Software Framework

The YANA software stack provides an end-to-end workflow for training, optimization, and deployment. So far it is composed of three main modules and utilizes the Neuromorphic Intermediate Representation (NIR) format to ensure interoperability with other neuromorphic frameworks.

- **YANA train:** This module facilitates hardware-aware SNN training. It is built upon the Norse framework and incorporates YANA’s specific quantized LIF neuron model, including the LUT-based leak calculation, ensuring that models are trained with hardware constraints in mind.
- **YANA deploy:** This module translates a trained SNN graph from its NIR representation into a deployable hardware configuration. It parses the network, optimizes the memory layout for physical hardware resources, and generates the final memory configuration files for the accelerator.
- **YANA runtime:** Operating directly on the ARM processor subsystem, this module interfaces with the hardware accelerator in the PL. It is responsible for initializing the accelerator with the generated configuration files, managing data flow during inference, and tracking performance metrics like execution latency.

C. Experimental Results

To demonstrate the efficacy of the event-driven computation utilized in our approach, we conduct experiments with varying

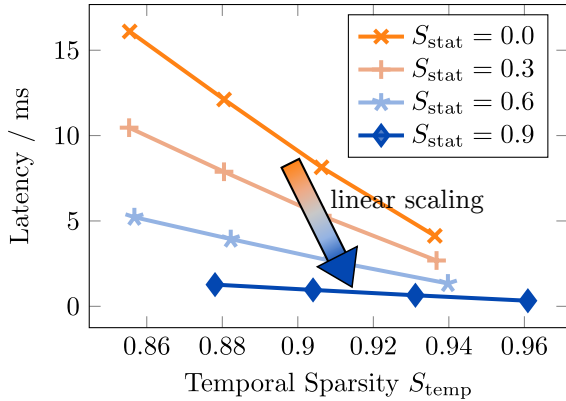


Fig. 5: Inference latency of SNNs deployed on YANA versus increasing temporal sparsity S_{temp} and for varying levels of S_{stat} . The data samples are taken from the SHD dataset [24]. Each point presents the average results of measurements of 20 data samples.

degrees of sparsity and measure the resulting execution time of the accelerator. In general, we distinguish between two types of sparsity. Dynamic temporal sparsity S_{temp} refers to sparsity along the time dimension of the input data, as well as the output of the hidden neurons within the network. Static spatial sparsity S_{stat} describes the sparsity in the connectivity between the neurons of a network. For our experiments, we sweep both sparsity types separately to determine their individual impact on the end-to-end execution time. The network consists of one hidden layer with 100 leaky integrate and fire (LIF) neurons and an output layer with 20 leaky integrate (LI) neurons with linear projections between the neuron layers. The training and deployment of the networks is done using our software framework. In order to adjust the temporal sparsity of the input data and hidden layers, we augment the input data by randomly dropping tensor elements with a probability according to the target sparsity. To improve the static sparsity, we use magnitude pruning on the linear projections of the network layers.

Figure 5 shows the average inference latency of 20 random samples taken from the Spiking Heidelberg Digits (SHD) dataset [24]. It is clear, that the inference latency decreases with increasing temporal sparsity. In fact, the correlation between temporal sparsity and inference latency is approximately linear. This shows the ability of the neuromorphic accelerator to take substantial advantage of tensor sparsity due to its event-driven nature. When comparing the latencies of networks exhibiting different static sparsities, we observe the same linear relation with high connectivity sparsities resulting in lower execution times. Finally, the latency resulting from hidden events could be further reduced by separately targeting the temporal sparsity of hidden layer units using regularization methods [25].

YANA’s event-driven design and its hardware-aware train/deploy/runtime have the potential to execute time-critical automotive workloads on a power budget. Integrated in a hybrid car-compute platform, it could solve tasks end-to-end or in combination with other accelerators, such as FlexNNGine, to process Hybrid Neural Networks [26], [27].

IV. DESIGN OF HETEROGENEOUS AUTOMOTIVE SYSTEMS INCORPORATING ACCELERATORS

A. Automotive AI benchmarks developed in CeCaS

Designing heterogeneous systems requires careful selection of hardware components to meet system requirements. In the automotive domain, AI workloads must satisfy strict latency, reliability, and energy constraints while operating under limited compute and thermal budgets. Estimating the real performance of a chosen hardware accelerator under these conditions is challenging, as vendor documentation typically lists peak GOPS that do not reflect achievable inference throughput. Therefore, AI benchmarks are commonly used to evaluate and compare different hardware options under realistic operating scenarios.

Most existing benchmarks target mobile edge AI, which is deployed onto consumer grade hardware platforms. While they consider tasks relevant to automotive applications, they do not give a complete or representative view of the final performance after integration into an Electronic Control Unit (ECU). Extreme thermal conditions and high image resolutions further reduce final performance due to thermal throttling. This highlights the need for benchmarking methodologies that explicitly account for automotive operating requirements.

Since automotive applications impose stringent demands on both functional correctness (model quality) and execution efficiency (hardware performance), a new benchmarking suite has been developed in collaboration between KIT, ZF, and Continental [28], as summarized in Table I. The suite groups AI models according to their target tasks. For image classification, we use ResNet50 [29], MobileNetV2 [30], and EfficientNet [31] as well-established classification backbones for embedded applications. For object detection and segmentation, we use YOLO V11n [32], a good trade-off between accuracy and hardware efficiency. For a higher accuracy segmentation model, we employ DeepLabV3 [33] with a ResNet50 backbone. Additionally, we include synthetic ResNet-like models to evaluate the performance of the CeCaS accelerators under high load conditions with extreme input image resolutions. This enables a more representative assessment of accelerator performance within the intended operational domain.

In addition, we examine the gap between theoretical and actual hardware performance. Many NNs are designed without considering the target hardware, leading to software optimizations that do not map efficiently onto AI accelerators, such as systolic arrays. For example, EfficientNet and MobileNet achieve only 2.83% and 3.14% hardware utilization on the FlexNNGine, respectively, due to their use of depthwise separable convolutions [28]. By designing NNs through Hardware-Aware Neural Architecture Search (HW-NAS), hardware utilization can be significantly improved, resulting in lower inference latency while maintaining network accuracy [34].

However, existing AI accelerator benchmarks define concrete pretrained models as reference workloads. This restricts the use of advanced neural network co-optimization methods such as Knowledge Distillation and HW-NAS, because these techniques modify the model architecture. We therefore argue that future benchmarks should instead define functional and non functional

TABLE I: Automotive Benchmarks developed in CeCaS. All models are quantized to int8, use ONNX opset 13 for exchangeability between frameworks, and are open source [28].

Task	Models	Dataset and Input Resolution	Metrics	
			Model Quality	HW performance
Classification and detector backbone	ResNet 50, MobileNetV2, EfficientNet	ImageNet, 3x224x224	Accuracy	
Detection	YOLO V11n	MS Coco, 3x640x640	Mean Average Precision (mAP)	Inference latency, throughput, power consumption, energy efficiency
Segmentation	YOLO V11n, DeepLabV3	MS Coco, 3x640x640	Intersection over Union (IoU)	
General HW benchmark, compiler test	ResNet-like synthetic models	ImageNet, upscaled up to 3x7680x4320	<i>not considered</i>	

constraints as well as target datasets for the System Under Test (SUT), where the SUT refers to a system involving both AI accelerators and an AI model as a whole. This approach would broaden the available design space for AI models and enable the discovery of more efficient, hardware-optimized solutions for integrated automotive AI systems [28].

B. RISC-V Virtual Prototype for Hardware Software Co-Design and Safety/Security Evaluation

Rapid hardware/software integration, as well model-based safety and security evaluation can be achieved using Virtual Prototypes (VPs) implemented in SystemC [35]. In CeCaS, a RISC-V-based SystemC virtual platform was extended to incorporate VPs of generic AI accelerators, as well as a model of the FleXNNgine. This simulation framework enables software workload testing before hardware prototypes are available. Built around Transaction-Level Modeling (TLM), the VPs can simulate the execution of target workloads much faster than RTL-level simulators.

To meet the goal of automotive safety and security compliance, specialized models and frameworks were deployed alongside VPs. To evaluate the power side-channel information leakage from AI accelerators, virtual prototypes are annotated with a dedicated power model [36]. Furthermore, the impact of faults can also be modeled at transaction level in VPs [35]. However, to match the fault propagation with actual hardware implementations at RTL, an open-source Fault-Injection (FI) framework called NAIL [37] was introduced. NAIL enables FI evaluation with RTL simulation or FPGA-based emulation for RISC-V-based AI accelerator SoCs.

C. Neural Network Inference Partitioning

An important aspect of designing a heterogeneous computing system is the partitioning of workloads across suitable hardware components. While CeCaS primarily relies on a central HPC unit for computation, processing all sensor data centrally is inefficient: raw camera streams place high demand on interconnect bandwidth, and multi-task CNN inference requires substantial compute resources. Offloading parts of the processing to sensor-edge devices can therefore reduce data traffic and alleviate the computational load on the central unit. However, since edge nodes offer only limited compute and memory resources, CNN partitioning must be carefully designed to meet

system-level latency and energy constraints. Performing early data preprocessing at the edge ultimately relieves in-vehicle data buses and supports a more balanced power distribution across the system.

For the evaluation of AI partitioning, an existing framework [38] was extended to enable automated inference partitioning and to analyze the distribution of computational workloads in a Gigabit Ethernet-based embedded system under various metrics [39]. These metrics include not only latency and energy consumption, but also the bandwidth required for data transmission between platforms and the memory footprint on each platform, which is typically highly constrained.

In addition, the framework was enhanced to support partitioning across more than two platforms. The results show that larger networks, such as EfficientNet-B0, benefit from multi-platform partitioning in terms of throughput and energy efficiency. In contrast, for smaller networks like SqueezeNet, the overhead of transmitting intermediate results contributes significantly to total latency and energy consumption, making partitioning across more than three platforms less effective.

V. CONCLUSION

In this work, we presented an overview of our contributions within the CeCaS project to the design and integration of AI accelerators in automotive systems. By utilizing both a conventional and a neuromorphic accelerator architecture that are both flexible, interoperable, and open-sourced, a wide range of current and future workloads can be executed efficiently. In addition, mechanisms for functional safety were introduced in accordance with ISO 26262 to ensure the suitability of the proposed designs for automotive applications.

We further introduced methods and tools for the design of heterogeneous automotive systems incorporating AI accelerators. These include a dedicated automotive AI benchmarking suite with considerations about improved future benchmarks, a RISC-V-based virtual prototyping framework for hardware/software co-design and safety evaluation, and an extended CNN inference partitioning framework for distributed computation across car servers and sensor-edge devices.

Together, these contributions establish a foundation for developing and evaluating next-generation high-performance automotive computing platforms that combine performance, flexibility, and safety.

REFERENCES

- [1] "Cecac homepage," <https://www.mannheim-cecas.de/>, last accessed: 2025/09/09.
- [2] B. Pachideh, S. Nitzsche, M. Neher, J. Krausse, C. Weigelt, K. Knobloch, V. P. Betancourt, and J. Becker, "YANA: Bridging the Neuromorphic Simulation-to-Hardware Gap," in *Brain Informatics*. Bari, Italy: Springer Nature, 2025, to be published.
- [3] F. Lesniak, A. Gutermann, T. Harbaum, and J. Becker, "Enhanced accelerator design for efficient cnn processing with improved row-stationary dataflow," in *Proceedings of the Great Lakes Symposium on VLSI 2024*, ser. GLSVLSI '24. New York, NY, USA: Association for Computing Machinery, 2024.
- [4] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks," *SIGARCH Comput. Archit. News*, vol. 44, no. 3, p. 367–379, Jun. 2016.
- [5] Institute of Information Processing Technology (ITIV), "FleXNNgine," <https://github.com/itiv-kit/flexnngine>, 2025, accessed: 2025-08-08.
- [6] F. Lesniak, A. Gutermann, A. Serdyuk, T. Harbaum, and J. Becker, "Asap: The adaptive scalable accelerator platform for versatile embedded systems," accepted for publication at Driving the Future Symposium, Springer, 2025.
- [7] J. Hofer, F. Kempf, T. Hotfilter, F. Kreß, T. Harbaum, and J. Becker, "SiFi-AI: A Fast and Flexible RTL Fault Simulation Framework Tailored for AI Models and Accelerators," in *Proceedings of the Great Lakes Symposium on VLSI 2023*. Knoxville TN USA: ACM, Jun. 2023, pp. 287–292.
- [8] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: representing model uncertainty in deep learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. New York, NY, USA: JMLR.org, Jun. 2016, pp. 1050–1059.
- [9] J. Hofer, M. Stammner, F. Kreß, T. Hotfilter, T. Harbaum, and J. Becker, "BayWatch: Leveraging Bayesian Neural Networks for Hardware Fault Tolerance and Monitoring," in *2024 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. Didcot, United Kingdom: IEEE, Oct. 2024, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/10753546/>
- [10] A. Mobiny, P. Yuan, S. K. Moulik, N. Garg, C. C. Wu, and H. Van Nguyen, "DropConnect is effective in modeling uncertainty of Bayesian deep networks," *Scientific Reports*, vol. 11, no. 1, p. 5458, Mar. 2021.
- [11] D. P. Kingma, T. Salimans, and M. Welling, "Variational Dropout and the Local Reparameterization Trick," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015.
- [12] J. Hofer, F. Lesniak, A. Gutermann, F. Wöran, T. Harbaum, and J. Becker, "Baynngine: Hardware-enabled bayesian neural network support for dependable ai inference," in *2025 IEEE 38th International System-on-Chip Conference (SOCC)*, 2025, pp. 1–6.
- [13] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, and et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [14] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.
- [15] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, pp. 607–617, 2019.
- [16] N. Reeb, J. Lopez-Randulfe, R. Dietrich, and A. C. Knoll, "Range and angle estimation with spiking neural resonators for FMCW radar," *Neuromorphic Computing and Engineering*, vol. 5, no. 2, May 2025.
- [17] B. Vogginger, F. Kreutz, J. López-Randulfe, C. Liu, R. Dietrich, H. A. Gonzalez, D. Scholz, N. Reeb, D. Auge, J. Hille, M. Arsalan, F. Mirus, C. Grassmann, A. Knoll, and C. Mayr, "Automotive Radar Processing With Spiking Neural Networks: Concepts and Challenges," vol. 16.
- [18] A. Shalunov, R. Halaly, and E. E. Tsur, "LiDAR-driven spiking neural network for collision avoidance in autonomous driving," vol. 16, no. 6, p. 066016. [Online]. Available: <https://doi.org/10.1088/1748-3190/ac290c>
- [19] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [20] W. Shariff, M. S. Dilmaghani, P. Kielty, M. Moustafa, J. Lemley, and P. Corcoran, "Event Cameras in Automotive Sensing: A Review," *IEEE Access*, vol. 12, pp. 51 275–51 306, 2024.
- [21] F. Kreutz, D. Scholz, J. Hille, H. Jiabin, F. Hauer, K. Knobloch, and C. G. Mayr, "Continuous Inference of Time Recurrent Neural Networks for Field Oriented Control," in *2023 IEEE Conference on Artificial Intelligence (CAI)*, Jun. 2023, pp. 266–269.
- [22] R. Halaly and E. Ezra Tsur, "Autonomous driving controllers with neuromorphic spiking neural networks," *Frontiers in Neurobotics*, vol. 17, Aug. 2023.
- [23] J. Krausse, S. Nitzsche, B. Pachideh, C. Weigelt, K. Knobloch, and J. Becker, "Exploring neuronal leakage for spiking neural networks on event-driven hardware," in *2024 International Conference on Neuromorphic Systems (ICONS)*. IEEE, 2024, pp. 140–147.
- [24] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2744–2757, Jul. 2022.
- [25] J. Krausse, M. Neher, K. Knobloch, and J. Becker, "How activity regularization harms the performance of pruned spiking neural networks," in *2025 International Conference on Neuromorphic Systems (ICONS)*. IEEE, 2025.
- [26] P. Gerhards, M. Weih, J. Huang, K. Knobloch, and C. G. Mayr, "Hybrid Spiking and Artificial Neural Networks for Radar-Based Gesture Recognition," in *2023 8th International Conference on Frontiers of Signal Processing (ICFSP)*, Oct. 2023, pp. 83–87.
- [27] A. Kugele, T. Pfeil, M. Pfeiffer, and E. Chicca, "Hybrid SNN-ANN: Energy-Efficient Classification and Object Detection for Event-Based Vision," in *Pattern Recognition*, C. Bauckhage, J. Gall, and A. Schwing, Eds. Cham: Springer International Publishing, 2021, pp. 297–312.
- [28] A. Serdyuk, A. Gutermann, F. Lesniak, F. Oberhoff, M. Dreher, M. Majer, A. Pschorr, I. Fürst-Walter, T. Harbaum, and J. Becker, "On benchmarking systolic-array-based accelerators for automotive applications," accepted for publication at Driving the Future Symposium, Springer, 2025.
- [29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015, arXiv:1512.03385 [cs]. [Online]. Available: <http://arxiv.org/abs/1512.03385>
- [30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. [Online]. Available: <http://arxiv.org/abs/1801.04381>
- [31] M. Tan and Q. V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. [Online]. Available: <http://arxiv.org/abs/1905.11946>
- [32] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [33] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. [Online]. Available: <http://arxiv.org/abs/1706.05587>
- [34] A. Gutermann, A. Serdyuk, F. Paraskevas, H. T. Kiesa, F. Lesniak, J. Schwarz, M. Hartmann, T. Harbaum, and J. Becker, "Improving ai accelerator performance through co-designing neural networks and systolic hw," in *2025 IEEE Nordic Circuits and Systems Conference (NorCAS)*, 2025, pp. 1–7.
- [35] A. Mahmoudi, A. Nešković, C. Thermann, R. Sehm, C. Hübner, T. Platenteich, R. Meyer, R. Buchty, M. Berekovic, and S. Mulhem, "A systematic mapping study on systemc/tlm modeling capabilities in new research domains," *ACM Transactions on Design Automation of Electronic Systems*, vol. 30, no. 4, pp. 1–41, 2025.
- [36] A. Nešković, S. Mulhem, A. Treff, R. Buchty, T. Eisenbarth, and M. Berekovic, "Systemc model of power side-channel attacks against ai accelerators: Superstition or not?" in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–8.
- [37] R. Sehm, C. Ewert, R. Buchty, M. Berekovic, and S. Mulhem, "Nail: Not another fault-injection framework for chisel-generated rtl," *arXiv preprint arXiv:2508.06344*, 2025.
- [38] F. Kreß, V. Sidorenko, P. Schmidt, J. Hofer, T. Hotfilter, I. Walter, T. Harbaum, and J. Becker, "Cnnparted: An open source framework for efficient convolutional neural network inference partitioning in embedded systems," *Computer Networks*, vol. 229, p. 109759, 2023.
- [39] F. Kreß, J. Hofer, T. Hotfilter, I. Walter, E. M. El Annabi, T. Harbaum, and J. Becker, "Automated search for deep neural network inference partitioning on embedded fpga," in *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Cham: Springer Nature Switzerland, 2023, pp. 557–568.