EMDRIVE Architecture: Embedded Distributed Computing and Diagnostics from Sensor to Edge

Patrick Schmidt[1], Iuliia Topko[1], Matthias Stammler[1], Tanja Harbaum[1], Juergen Becker[1], Rico Berner[2], Omar Ahmed[2], Jakub Jagielski[2], Thomas Seidler[2], Markus Abel[2], Marius Kreutzer[3], Maximilian Kirschner[3], Victor Pazmino Betancourt[3], Robin Sehm[4], Lukas Groth[4], Andrija Neskovic[4], Rolf Meyer[4], Saleh Mulhem[4], Mladen Berekovic[4], Matthias Probst[5], Manuel Brosch[5], Georg Sigl[5], Thomas Wild[5], Matthias Ernst[5], Andreas Herkersdorf[5], Florian Aigner[6], Stefan Hommes [7], Sebastian Lauer [7], Maximilian Seidler[8], Thomas Raste[9], Gasper Skvarc Bozic[10], Ibai Irigoyen Ceberio[10], Muhammad Hassan[10], Albrecht Mayer[10]
[1]Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{patrick.schmidt2, iuliia.topko, stammler, harbaum, becker}@kit.edu
[2]Ambrosys GmbH, Potsdam, Germany
[3]FZI Research Center for Information Technology, Karlsruhe, Germany
[4]Universität zu Lübeck, Lübeck, Germany
[5] Technical University of Munich, Munich, Germany
[6]Elektrobit Automotive GmbH, Erlangen, Germany
[7]ZF Friedrichshafen AG, Friedrichshafen, Germany
[8]Siemens AG Technology, Erlangen, Germany
[9]Continental Automotive Technologies GmbH, Frankfurt am Main, Germany
[10]Infineon Technologies AG, Munich, Germany

***Abstract*—Future automotive architectures are expected to transition from a network-centric to a domain-centered architecture featuring central compute units. Powerful domain controllers or smart sensors alleviate the load on these central units and communication systems. These controllers execute tasks with varying criticalities on heterogeneous multicore processors, and are ideally capable of dynamically balancing the computing load between the central unit and sensors. Here, Artificial Intelligence (AI) capabilities play a crucial role, as it is in high demand for such an automotive architecture. However, AI still requires specialized accelerators to improve their computation performance.**

Task-oriented distributed computing with criticalities up to ASIL-D necessitates the development and utilization of specialized methodologies, such as safety, through the isolation and abstraction of low-level hardware concepts. Meanwhile, online monitoring and diagnostics become vital features to detect errors during operation.

The EMDRIVE architecture includes methods, components, and strategies to enhance the performance, safety, and security of such distributed computing platforms. The nationally funded EMDRIVE project connects its twelve partners from academia and industry and is currently in its intermediate stage.

***Index Terms*—automotive, safety, monitoring, load balancing, intrusion detection, AI accelerator**

I. Introduction

Vehicle functions increasing the safety are essential to modern vehicles, with more sophisticated systems being integrated to vehicles. The goal is to reach a level where those functions can take over the driver's responsibility to control the vehicle autonomously and safely [1]. With such features becoming a crucial component in modern vehicles, novel computational architectures are required to meet the demands for computational power and safety. Further, vehicles will be required to communicate with the cloud as some functions can not be performed on-board and must be offloaded to powerful compute clusters. Delivering high-quality driver assistance requires the use of sensor-fusion algorithms, as the information gathered by a single sensor may be incomplete. Current architectures, which perform object detection directly on the sensor node, are unable to meet this demand. Instead, distributing the computation between the sensor and a highly performant central compute unit will be necessary [2]. In this case, the sensor node only performs pre-computations, while more complex algorithms are performed centrally to enable sensor fusion. A drawback of this approach is the increase of on-board network traffic, as sensor data is transmitted instead of the detected objects.

Another concern lies in the central compute unit. While it provides significant computational power, a major challenge is the isolation of critical and non-critical tasks. Further, the introduction of AI algorithms lead to the inclusion of dedicated accelerators, contributing to a heterogeneous architecture. Hence, it is crucial to assign workloads at runtime to certain hardware units in order to maximize resource utilization [3].

Guaranteeing safety of such complex systems and achieving ASIL-D certification is a key issue. Novel software features, combined with the complex compute architecture, can result in the occurrence of extremely rare errors that can cause critical failures. Due to the complexity of the system, reliably identifying such errors at design time may be impossible. Therefore, online monitoring systems are needed to detect

This work was funded by the German Federal Ministry of Education and Research (BMBF) under grant number 16ME0454 (EMDRIVE). The responsibility for the content of this publication lies with the authors.

979-8-3503-4859-0/DATE24/© 2024 EDAA

anomalies at runtime and provide ways to mitigate them [4]. In the scope of the EMDRIVE project, we develop tools and components to tackle the three major challenges presented: distributed computation, dynamic distribution of workloads and online monitoring of in-vehicle communication.

Our consortium consists of twelve partners, five of which are academic, with seven industry partners. Each partner focuses on different aspects of the computing architecture. The Karlsruhe Institute of Technology (KIT) focuses on the co-design of hardware and software for efficient embedded AI applications. The Technical University of Munich (TUM) develops a diagnosis companion box for online monitoring and analyzes the developed hardware accelerator to find vectors for side-channel attacks. Ambrosys (AMB) develops a simulation and development platform for machine learning, and establishes the prototype of a cloud backend for the monitoring and diagnosis toolbox. The Universität zu Lübeck (UzL) focuses on two aspects: (1) the development of a virtual prototype of a CNN accelerator, and (2) the training of several AI models, especially CNN for radar data processing. Elektrobit Automotive GmbH (EB) is focussing on safety aspects in dynamic systems. Infineon Technologies (IFX) develops an architecture for remote access of production devices to facilitate remote diagnosis. ZF Friedrichshafen AG (ZF) develops a concept for an intrusion detection system to detect attacks at an early stage. Siemens AG is investigating techniques for dynamic load balancing in the industrial context. Forschungszentrum Informatik (FZI) works on methods to design service-based software architectures. Continental Automotive Technologies GmbH develops a framework to simplify the development of vehicular software.

The following sections will discuss our contributions to the three challenges we identified. Sec. II introduces our Smart Sensor Platform to preprocess and compress radar data. In Sec. III we describe solutions to address online monitoring and anomaly detection for in-field usage. Finally, isolation and load balancing techniques are discussed in Sec. IV.

II. Smart Sensor Platform

With the move towards distributed compute architectures becoming more common, designing the preprocessing stage is a crucial task. In the EMDRIVE project, we develop a hardware platform and accompanying software tooling to perform efficient processing of radar data. As computations move more towards the central platform, our focus lies on designing an optimized architecture for a given task and less on providing maximum compute power. We also develop the necessary software tools to effectively utilize the computational platform.

A. Preprocessing of Radar Data

The pre-processing of raw radar data usually employs Fast Fourier Transformation (FFT) as shown in Fig.1: radar sensors collect the data and an FFT unit extracts features [5]. The output of this processing is the Range-Doppler Spectrum (RDS). RDS serves as the input for further processing steps, extracting relevant and task-specific information. To accelerate this pre-processing step, typical systems feature dedicated hardware

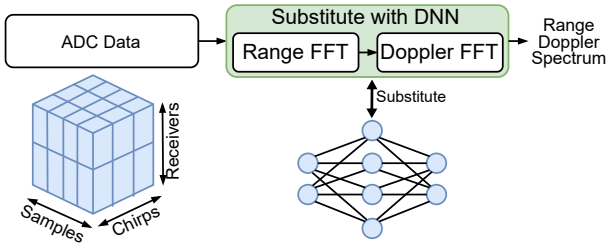

Fig. 1: FFT- vs. DNN-based Pre-processing of Raw Radar Data. We substitute the FFT with a DNN.

accelerators for FFT computation. In a modern system, it could be beneficial to replace such hardware components with multipurpose accelerators also suitable for deep neural networks (DNN) to reduce the hardware overhead. We aim to substitute the FFT in the pre-processing step with a DNN, thereby eliminating the need for dedicated FFT accelerators.

To achieve this, a DNN is trained on input/output pairs collected using the classical FFT approach. This allows explorations regarding trade-offs between DNN accuracy and execution time as a result of model size and complexity. As multiplication in the frequency domain is equivalent to convolution in the time domain, the proposed DNN model is mainly composed of convolution layers. Our initial results are very promising: the proposed DNN has ∼300K weights and can achieve a loss near zero when intentionally over-fitting on a small subset of the dataset.

B. HW/SW Co-design for AI based Radar Processing

In order to increase the performance and energy efficiency of AI computations, dedicated accelerators are needed. Here, the design of scalable architectures offer a way to reduce time-to-market through configuration for different performance domains. In the EMDRIVE project, we choose Gemmini [6] as the basis for our accelerator due to its scalability and open-source nature, and develop a compiler based on Apache TVM [7]. We extended the accelerator with support for relevant neural layers and activation functions to increase the performance.

In the second phase of the project, we will focus on further development of the compiler to achieve greater performance improvements. We build on a previous implementation [8] but extend it with support for auto-tuning of convolutional layers, as those are the most relevant for our use-case. Additionally, we will focus on enhancing the real-time capabilities of the architecture.

C. Analytical Performance Modelling of Systolic Arrays

Scalable architectures result in novel challenges for hardware designers. A major difficulty is finding the optimal configuration of a certain architecture to process a given workload in an optimal way. To deal with this challenge, we developed an automated design space exploration (DSE) tool to find an optimal configuration of a systolic array for a given workload. An overview of our tool is given in Fig. 2. We automatically construct the design space of valid configurations. For a given

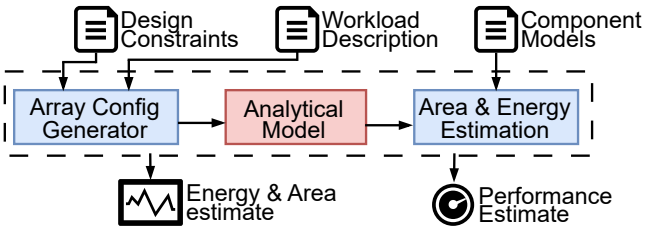

Fig. 2: Our DNN analysis tool with the analytical model at its center. We automatically construct the design space and evaluate configurations w.r.t. performance, area and energy [10]

workload, we reconstruct the executed instructions for any given design point and use them to calculate the computational intensity. With this information, we leverage the roofline model to calculate a performance estimate and use Accelergy [9] for energy and area estimates. Our results show a runtime reduction of up to 138x over state-of-the-art tools, while performance estimations are within 1% deviation [10].

D. Virtual Prototype of a Neural Network Accelerator

In hardware design, the process of DSE can be significantly accelerated with the use of Virtual Prototypes (VPs). Previous work demonstrated the feasibility of VPs in aerospace electronics [11]. Another general-purpose approach shows a SoC VP implementing a RISC-V Core with SystemC TLM [12].

However, none of the existing approaches cover neural network accelerators. Implementing a VP of a systolic-array-based neural network accelerator coupled to a CPU core can be used to speed up the DSE targeting automotive-specific AI algorithms. The VP enables the fast exploration of different DNN workloads with the target hardware architecture. Additionally, performing early design stage safety and security evaluations can give suggestions of potential threats [13].

E. Cloud Platform for Collaborative AI Development

The previous discussions prove that we require suitable tooling for an easy deployment of models for training or inference on high-performance chips in autonomous vehicles. The existing tooling is not satisfactory, e.g. reproducibility, collaboration and monitoring of ML models are not adequately addressed. With Mantik, we provide an open-source cloud platform, which simplifies the collaborative development of machine learning models on HPC facilities, and enhances reproducibility by supporting data and code versioning as well as experiment tracking [14].

Within the EMDRIVE project, Mantik is developed towards a platform that allows for generating artificial data for training and security as well as safety evaluations of autonomous driving algorithms.

F. Security Analysis of the Hardware Platform

With increasing popularity of moving AI to the edge, attacks also raise in popularity. Edge devices enable additional attack vectors such as side-channel attacks [15] that aim to steal the Intellectual Property (IP) stored in the AI-algorithm in the form of its parameters. Hence, countermeasures are necessary, with shuffling [16] and masking [17] being prominent examples.

To harden the accelerator used in the EMDRIVE project, we perform a vulnerability analysis. In a first step, we identified leakage with TVLA [18] when observing the overall power consumption of a Gemmini implementation, as well as the electromagnetic emanations (EM) of areas over the FPGA. This indicates that side-channels can pose vulnerabilities to potential IP-theft. Next, we employ the simulation tool TOFU [19] to identify attack vectors in detail. We aim to close those vectors in an iterative process by using mainly masking as a countermeasure, in order to make power consumption or EM independent of the processed secret data while maintaining performance.

III. Monitoring and Diagnosis

The classical approach for achieving the highest safety requirements, such as ASIL-D in the automotive industry, aims to identify all systematic errors early during the development phase. However, with increasingly complex systems, detecting all potential error causes at design time becomes infeasible.

We address the challenge of achieving ASIL-D compliant failure rates for distributed control units in autonomous driving. For this, it is essential to recognize, analyze, and mitigate errors and effects occurring in the field that negatively impact performance, safety, and overall system security. This necessitates a powerful, safe, and secure observation capability in the vehicle with greater flexibility than conventional error routines.

A. Monitoring Chip Traces

With the introduction of new hardware components and software features, monitoring features are becoming more relevant as the increased complexity results in many additional failure points. To ensure high dependability of the system, a continuous monitoring is required.

An integral part of any vehicle is an electronic control unit (ECU), which can comprise one or more interconnected microcontrollers (MCUs). By having an efficient architecture for accessing those MCUs, remote access is possible even in a production device, which is essential for remote diagnosis.

We developed a new tool access architecture called Tool Access Socket (TAS), as shown in Fig. 3. It can be used for debug & trace of microcontrollers from early development to in-field diagnosis. Different *Tools* are provided to access the hardware targets. They enable standard read/write access, channel interfaces for file I/O or direct access to the on-chip tracing infrastructure. The *TAS Server* processes incoming connections and assigns them to a selected target through the dispatcher. Target devices can be connected via a standard debug interface (*Tool HW*) or through Ethernet to enable remote access. When no specific tool hardware is used, a firmware called *TAS Agent SW* is provided to enable debug and trace access via Ethernet. To abstract the different debug interfaces, the TAS Server uses *Agents* to model common functionality. This design hides the physical interface from the Tool, allowing a unified tooling for different product families [20].

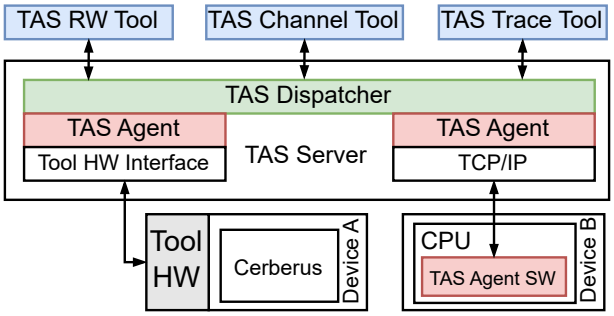

Fig. 3: The TAS architecture for continuous monitoring. Different tools are provided to access on-site or remote devices. The dispatcher connects Tools with devices through an Agent.

B. Layer Trace Based Anomaly Detection

In addition to a processor-based anomaly detection, we use layer tracing to monitoring AI accelerators and to detect anomalies during operation. While neural network-based models usually assign an arbitrarily large confidence to objectively false predictions, the execution and calculation of these models must be monitored during execution. In this project, this is done by layer tracing. To choose from various published methods, we introduce EFFECT, a framework to compare conventional, and machine learning based methods to automatically find an optimum for a given use case. EFFECT is able to automatically evaluate strategies in terms of accuracy, latency as well as memory consumption. Additionally, our tool is able to account for different dimensions of intermediate layer results [21].

C. Diagnosis Companion Box

To enable continuous diagnosis of sporadically occurring systematic errors which have not been detected at design time, we propose a diagnosis companion box (DCB) to detect these issues at runtime. The DCB continuously monitors the system for deviations from the expected behavior and performs an initial analysis of potential root causes. Fig. 4 shows an overview of the box and its components.

Due to the involved amounts of data, permanently tracing all ECU processing and data exchange, recording raw data for offline analysis, or analyzing all data at runtime to identify potential problems is impossible. Therefore, the DCB follows a sequential approach, with the underlying assumption that deviations of expected behavior manifest in anomalies of the data exchange among sub-functions. Such anomalies could be deviations from the expected arrival time pattern of Ethernet frames or their sizes. The DCB is permanently monitoring the traffic on the automotive Ethernet backbone. Once it detects an anomaly, it begins analyzing the processing within the ECU that is the source of the flow with the abnormal behavior. This ECU likely has problems in processing the associated sub-function sending out the abnormal traffic. For the analysis, the companion box is provided with a set of predetermined checks carried out sequentially, where each of these checks is associated with a potential root cause for that anomaly.

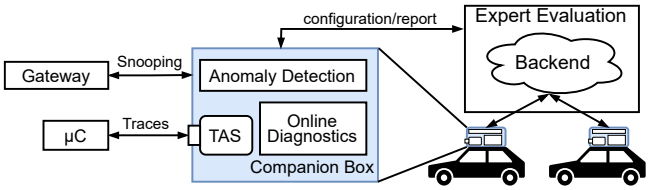

Fig. 4: Overview of the diagnosis box concept for online anomaly detection. Vehicles are equipped with a box to perform online diagnostics and anomaly detection.

The checks use the trace monitoring infrastructure as described in Sec. III-A and consist of verifying the timing of task executions, memory access behavior and other relevant metrics that can be extracted from processor traces. Each check is associated with a specific Multi-Core Debug Solution (MCDS) configuration that delivers the required trace data via the TAS Server to the DCB.Intermediate results or preprocessed traces may be forwarded into a cloud backend for further analysis and for controlling the overall diagnosis process.The DCB triggers the analysis tasks one after the other to correlate the communication anomaly with deviations observed in the processing within the ECU.

This online vehicle-based monitoring and analysis is supplemented by a cloud-based backend. Depending on the reports received from DCBs installed in a fleet of cars with identical hardware and software configuration, the backend can configure the individual DCBs so that the search for the potential root cause of an anomaly is done in a distributed manner, improving the statistical coverage of the diagnosis.

This describes the behavior of the DCB when it acts in a mode loosely coupled with the cloud backend. Alternatively, a tightly coupled operation is conceivable as well where the backend decides about assigning specific diagnosis sub-tasks to different diagnosis boxes within the fleet of cars so that different potential root causes can be analyzed in parallel.

As a first step we implemented a simulation of the anomaly detection and work is now ongoing on an FPGA prototype of the DCB.

D. Cloud backend for monitoring and diagnosis

In Sections III-A and III-C, we have discussed approaches for monitoring, error detection, and diagnosis operated on the vehicle. These solutions prove highly effective in addressing acute failures in the car. However, they face limitations concerning a comprehensive history of events and data, primarily due to constrained storage and computing capacities. To overcome this limitation, we augment the monitoring and diagnosis systems with a cloud backend, see Fig. 4. It enables the loading of data recorded on the edge device, including status updates, failure messages, detected failures, and the analysis of this data based on historical information. Using this data, the detection and diagnosis methods running in the car can be evaluated asynchronously and updated if necessary. Another notable feature of the cloud backend is its capability to harness the full potential of swarm intelligence: Data from various

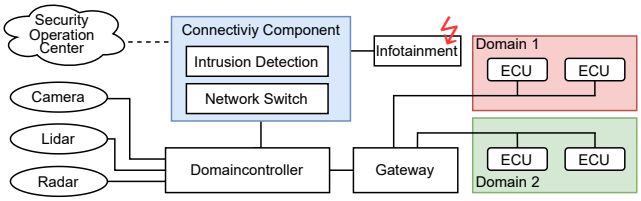

Fig. 5: Network Architecture with IDS and an attack on the infotainment system

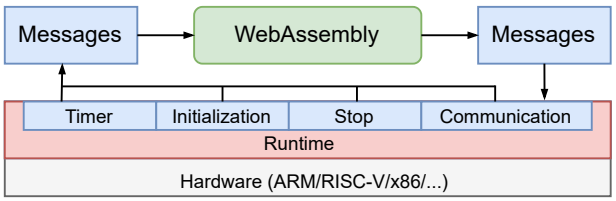

Fig. 6: High-level view of the concept of WebAssembly modules in service-oriented architectures

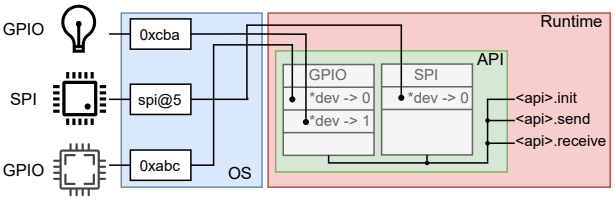

Fig. 7: Device API for WebAssembly

cars can be aggregated to collectively enhance the security and safety algorithms running in individual cars.

E. Intrusion Detection

Detecting attacks against a vehicle is necessary to guarantee security of the system [22]. For this reason, we have developed a concept for an intrusion detection system (IDS) in our Ethernet-based network architecture, see Fig. 5. The connectivity component is the core component of our concept. All traffic sent over the network is automatically routed via a network switch located on it. Our IDS is also located on the connectivity component. The switch mirrors all traffic and sends it to the IDS. In our threat model, we assume an adversary that has gained control of a component in the vehicle, such as the infotainment system, and can send messages into the vehicle network. As IDS software, we use Snort to analyze the traffic [23]. Snort is a rule-based IDS, which means it uses rules to check network traffic for suspicious packets. In case of an attack, Snort outputs a warning and additionally a recording of the traffic that triggered a certain rule. In a further step, these warnings can be sent together with a recording of the traffic to a backend-server, a so-called Security Operation Center (SOC), via an over-the-air solution. In the SOC, the alerts can be analyzed again and measures can be taken in the event of an attack. In order to keep the false-positive rate of the IDS as low as possible, our concept involves first simulating the vehicle network in a secure environment. Based on this simulation, the rules of the IDS can be adapted so that only actual attacks on the vehicle are detected and reported. This has the further advantage that the amount of data that has to be sent to an SOC is kept to a minimum.

IV. Dynamic Load Balancing

With the introduction of multicore architectures and the inclusion of dedicated accelerators, software tools to effectively utilize these heterogeneous architectures are needed. Within the EMDRIVE project, we develop systems for dynamic load balancing to maximize the utilization of system resources and provide tools to isolate safety critical tasks.

A. Load Balancing for Heterogeneous Compute Systems

For the deployment, management and migration of lightweight applications in heterogeneous systems, WebAssembly has been identified as a novel and flexible solution. WebAssembly applications are similar in structure to traditional containers, but have no direct dependency on the kernel or operating system and are in a completely isolated environment, so they have no independent access to interfaces or device hardware. To enable the integration of such modules into a vehicle architecture, a runtime environment to integrate WebAssembly into service oriented architectures for edge systems is required [24].

Fig. 6 shows the structure of the runtime environment. The communication protocol is completely shifted from the WebAssembly application to the runtime environment. This also enables the reuse of the same application in different systems with different communication protocols. The runtime environment also allows the WebAssembly application to use additional signal sources as input signals, such as timers, stop signals and initialization data.

To communicate with the outside world, we also require the use of peripherals without escaping the provided sandbox. Therefore, we expose peripheral devices to the WebAssembly module. We created a framework to semi-automatically generate APIs that mediate between the device and the module (see Fig. 7). The module interacts with those devices via `int32` typed handlers, preventing direct access to the devices' transportation API. This allows us to use memory-mapped and port-mapped devices safely without escaping the sandbox.

B. AI-Accelerators as a Service

For the dynamic usage of accelerator resources, an interface for the integration of accelerator hardware into service-oriented architectures is being developed. This allows efficient usage of heterogeneous accelerators from isolated applications without direct hardware access.

The general architecture is shown in Fig. 8. Applications can request acceleration from a central scheduler, which is able to load balance between multiple different hardware accelerators. Any input and output data is going directly from and to the required components, without needing to go through the application beforehand, to reduce the overall traffic.

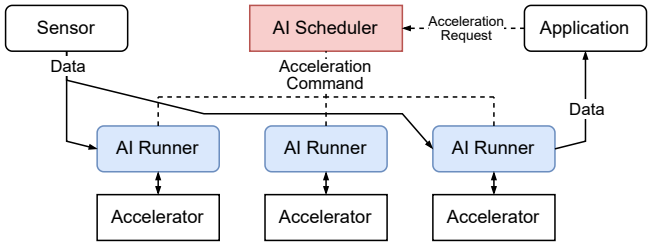

Fig. 8: Concept for the integration of accelerator hardware in service-oriented architectures

C. Safety Aspects of a Dynamic System

Running multiple software units on the same hardware poses a challenge for mixed criticality systems. Special treatment is required to allow units of different criticalities to reside on the same ECU without lowering the safety level of all components.

The L4 microkernel can be used to establish strict partitioning between a critical and non-critical partition of the same system. It provides a first abstraction layer as enabler for higher level operating systems or applications. This base system is also meant to handle scenarios that are considered Denial of Service in safety aspects. Each application is still behaving as defined, but on system level the load cannot be handled. These situations can occur in a general overload scenario, degrading system performance or concurrent hardware access. In EMDRIVE, we utilize L4Re [25] as a base operating system and further develop it towards a dynamic environment for safety needs.

D. Continental Automotive Edge Framework

To ease the co-development of future hardware and software components, we introduce the Continental Automotive Edge (CAEdge) framework. It provides a development environment for the design of software architectures necessary for future autonomous vehicles. To achieve this, CAEdge is equipped with the virtual twin of an ECU, providing feedback about potential errors much earlier in the development cycle than is currently feasible. Thereby, we are able to greatly reduce the development time necessary [1].

In EMDRIVE, CAEdge is used to develop the actuator control for a motion and stability system for autonomous and manual driving. For this, we must take requirements like energy efficiency and driving dynamics into account, while we have to pay special attention to safety aspects. CAEdge is able to synthesize a model predictive controller from these constraints.

V. Conclusion

The EMDRIVE project is developing the necessary tools and components to build compute architectures suitable for autonomous driving. We focus on novel architecture paradigms resulting from the distributed computing necessary to deliver sensor fusion. We identify the resulting challenges and present our first intermediate results. Special attention is given to safety-critical components of the architecture, such as online diagnosis features to detect sporadic errors that cannot be reliably caught at design time. Different aspects will continuously be elaborated in published and planned publications.

References

[1] T. Raste *et al.*, "Characteristics of modern motion and stability systems," in *31st Aachen Colloquium Sustainable Mobility 2022*, 2022.

[2] O. Burkacky, J. Deichmann, and J. P. Stein, "Automotive software and electronics 2030," 2019.

[3] A. A. Syed *et al.*, "Dynamic scheduling and routing for tsn based in-vehicle networks," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, 2021.

[4] I. P. Gomes and D. F. Wolf, "Health Monitoring System for Autonomous Vehicles using Dynamic Bayesian Networks for Diagnosis and Prognosis," *Journal of Intelligent & Robotic Systems*, vol. 101, p. 19, 2020.

[5] S. M. Patole, M. Torlak, D. Wang, and M. Ali, "Automotive radars: A review of signal processing techniques," *IEEE Signal Processing Magazine*, vol. 34, pp. 22–35, Mar. 2017. Conference Name: IEEE Signal Processing Magazine.

[6] H. Genc *et al.*, "Gemmini: Enabling Systematic Deep-Learning Architecture Evaluation via Full-Stack Integration," in *Proceedings of the 58th Annual Design Automation Conference (DAC)*, Dec. 2021.

[7] T. Chen *et al.*, "TVM: An Automated End-to-End Optimizing Compiler for Deep Learning," 2018.

[8] F. N. Peccia and O. Bringmann, "Integration of a systolic array based hardware accelerator into a dnn operator auto-tuning framework," 2022.

[9] Y. N. Wu *et al.*, "Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2019.

[10] T. Hotfilter *et al.*, "An analytical model of configurable systolic arrays to find the best-fitting accelerator for a given dnn workload," in *Proceedings of the DroneSE and RAPIDO: System Engineering for Constrained Embedded Systems*, RAPIDO '23, p. 73–78, Association for Computing Machinery, 2023.

[11] T. Schuster, R. Meyer, R. Buchty, L. Fossati, and M. Berekovic, "SoCRocket - a virtual platform for the european space agency's soc development," in *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, pp. 1–7, 2014.

[12] V. Herdt, D. Große, H. M. Le, and R. Drechsler, "Extensible and configurable risc-v based virtual prototype," in *2018 Forum on Specification & Design Languages (FDL)*, pp. 5–16, 2018.

[13] A. Neskovic *et al.*, "SystemC model of power side-channel attacks against ai accelerators: Superstition or not?," in *'23: Proceedings of the 42nd IEEE/ACM International Conference on Computer-Aided Design*, 2023.

[14] T. Seidler *et al.*, "Mantik: A workflow platform for the development of artificial intelligence on high-performance computing infrastructures," 2023. submitted.

[15] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *Proceedings of the 28th USENIX Conference on Security Symposium*, pp. 515–532, 2019.

[16] M. Brosch, M. Probst, and G. Sigl, "Counteract side-channel analysis of neural networks by shuffling," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1305–1310.

[17] A. Dubey *et al.*, "Modulonet: Neural networks meet modular arithmetic for efficient hardware masking," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 506–556, 2022.

[18] T. Schneider and A. Moradi, "Leakage assessment methodology," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 495–513, Springer, 2015.

[19] M. Gruber and G. Sigl, "Tofu - toggle count analysis made simple." Cryptology ePrint Archive, Report 2022/129, 2022. https://ia.cr/2022/129.

[20] G. S. Bozic *et al.*, "A new generation automotive tool access architecture for remote in-field diagnosis," in *WCX SAE World Congress Experience*, SAE International, apr 2023.

[21] M. Stammler *et al.*, "EFFECT: An end-to-end framework for evaluating strategies for parallel ai anomaly detection," *Procedia Computer Science*, vol. 222, pp. 499–508, 2023.

[22] UNITED NATIONS, "Addendum 154 – UN Regulation No. 155," 2021.

[23] Martin Roesch, Chris Green, Sourcefire Inc., Cisco, "Snort - network intrusion detection & prevention system."

[24] M. Kreutzer *et al.*, "Work-in-progress: Integrating webassembly into service-oriented architectures for edge systems," in *2023 International Conference on Embedded Software (EMSOFT)*, pp. 1–2, 2023.

[25] K. Lampka, J. Thurlby, A. Lackorzynski, and M. Hähnel, "Safety certification with the open source microkernel-based operating system l4re," in *International Conference on Computer Safety, Reliability, and Security*, pp. 31–45, Springer, 2022.