This document is a preprint of: F. Kempf, J. Hoefer, T. Harbaum et al., “The ZuSE-KI-Mobil AI Accelerator SoC: Overview and a Functional Safety Perspective,” in Proceedings of Design Automation and Test in Europe (DATE 2023), Antwerp, Belgium, Apr 2023.

# The ZuSE-KI-Mobil AI Accelerator SoC: Overview and a Functional Safety Perspective

Fabian Kempf[1], Julian Hoefer[1], Tanja Harbaum[1], Juergen Becker[1], Nael Fasfous[2], Alexander Frickenstein[2], Hans-Joerg Voegel[2] Simon Friedrich[3], Robert Wittig[3], Emil Matúš[3], Gerhard Fettweis[3], Matthias Lueders[4], Holger Blume[4], Jens Benndorf[5], Darius Grantz[5], Martin Zeller[5], Dietmar Engelke[5], Karl-Heinz Eickel[5]

[1]Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{fabian.kempf, julian.hoefer, becker}@kit.edu
[2] Bayerische Motoren Werke Aktiengesellschaft (BMW AG), Munich, Germany
[3] Technical University of Dresden (TUD), Dresden, Germany
[4]Leibniz University Hannover (LUH), Hannover, Germany
[5]Dream Chip Technologies GmbH, Hannover, Germany

***Abstract*—ZuSE-KI-Mobil (ZuKIMo) is a nationally funded research project, currently in its intermediate stage. The goal of the ZuKIMo project is to develop a new System-on-Chip (SoC) platform and corresponding ecosystem to enable efficient Artificial Intelligence (AI) applications with specific requirements. With ZuKIMo, we specifically target applications from the mobility domain, i.e. autonomous vehicles and drones. The initial ecosystem is built by a consortium consisting of seven partners from German academia and industry. We develop the SoC platform and its ecosystem around a novel AI accelerator design. The customizable accelerator is conceived from scratch to fulfill the functional and non-functional requirements derived from the ambitious use cases. A tape-out in 22 nm FDX-technology is planned in 2023. Apart from the System-on-Chip hardware design itself, the ZuKIMo ecosystem has the objective of providing software tooling for easy deployment of new use cases and hardware-CNN co-design. Furthermore, AI accelerators in safety-critical applications like our mobility use cases, necessitate the fulfillment of safety requirements. Therefore, we investigate new design methodologies for fault analysis of Deep Neural Networks (DNNs) and introduce our new redundancy mechanism for AI accelerators.**

***Index Terms*—System-on-Chip, AI Accelerator, Development Methodology, Fault Simulation, Functional Safety**

## I. Introduction

The race to provide more autonomous features for vehicles, robots or drones, has led to a clear trend in technological advancements in several domains, such as computer vision, decision-making, and artificial intelligence (AI). With the increasing level of autonomous driving features to be offered, the challenges in this field have changed dramatically. As of today, the enormous potential can only be estimated, but it is certainly going to change the market extensively and have long-term impacts.

Advances in autonomous driving and robotics have established the use of AI algorithms as they have become state-of-the-art in computer vision. Particularly, AI algorithms are set to enable high levels of autonomous driving to make independent decisions, even in safety-critical situations. Therefore, data received via cameras, LiDAR, radars, and other sensors must be interpreted and processed safely *and* in real-time. The problem here is the large amounts of data. Processing this amount of data pushes today’s embedded systems to the limits of their capabilities and even exceeds them.

At the same time, low- and zero-emission driving concepts are another core focus for the mobility domain, especially electric vehicles. With regard to battery-electric drives, the energy consumption of the autonomous driving system is becoming increasingly important. The energy stored in the battery systems should primarily be used to power the vehicle’s drive systems and not for on-board electronics and AI-based computing systems. To resolve this conflict and secure or expand their international market position, the availability of powerful and energy-efficient processors for AI-intensive applications is an essential prerequisite for automotive manufacturers. Furthermore, introducing powerful SoC designs and ecosystems for AI accelerators in the relevant value creation areas plays at least as important a role in the continued economic success of the automotive industry as the performance of the AI accelerators themselves.

Our consortium aims to establish such an SoC and corresponding ecosystem. It consists of seven partners; three universities and four industry partners. Each partner focuses on different aspects of the system. The Karlsruhe Institute of Technology (KIT) investigates safety features and the development methodology, including virtual platforms for SoCs and AI accelerators. Software tooling for compilation and deployment of AI models onto the SoC is done by the Leibniz University Hannover (LUH). The novel and energy efficient AI accelerator is developed by the Technische Universität Dresden (TUD). These three partners from academia combine their research activities with interests from industry. Dream Chip Technologies GmbH integrates the novel AI accelerator, safety features and other components into the SoC, and is responsible for layout and physical design. Bayerische Motoren Werke (BMW AG) provides a 3D object detection use case and is involved in the development methodology. A drone use case is provided by Infineon Technologies AG. Both uses cases will showcase the

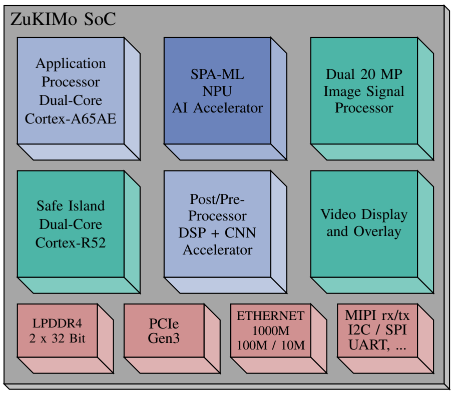
ZuKIMo SoC
Application Processor Dual-Core Cortex-A65AE
SPA-ML NPU AI Accelerator
Dual 20 MP Image Signal Processor
Safe Island Dual-Core Cortex-R52
Post/Pre-Processor DSP + CNN Accelerator
Video Display and Overlay
LPDDR4 2 x 32 Bit
PCIe Gen3
ETHERNET 1000M 100M / 10M
MIPI rx/tx I2C / SPI UART, ...

Fig. 1: Block diagram of the ZuKIMo SoC. The green components are developed by Dream Chip. The SPA-ML architecture is developed within the consortium by TUD.

project’s results. Finally, the goal to build up the ecosystem around the SoC platform and acquiring new use cases and end users is managed by Infineon and T3-Technologies.

With this article we give an overview of the ZuKIMo project to provide a reference for future publications emerging from the research activities in this project. In the following section the project’s core goals are presented. Then in section III, design methodologies are brought to focus highlighting current results and scientific papers emerging from the project. After this we explain our DNN resilience studies and functional safety features. Finally, a conclusion is given.

## II. The ZuKIMo Project

In this section we will give an overview of the ZuKIMo project. First, we present the overall SoC design. Then we introduce the newly developed AI accelerator and the software tooling before giving an overview of the target use cases.

### A. The ZuKIMo SoC

The SoC consists of multiple state-of-the-art components to fulfill the platform requirements. Figure 1 shows the essential components of the ZuKIMo SoC. The majority of computing performance comes from the application processor and two accelerators. The ARM dual-core Cortex-A65AE processor provides enough performance capacity for executing high-performance applications. Additionally, the SoC provides a commercial DSP for data pre- and post-processing. Furthermore, the DSP provides enough performance for limited CNN acceleration. The central accelerator of the ZuKIMo SoC is the novel mixed-precision AI accelerator. The Scalable-Precision Accelerator for Machine Learning (SPA-ML) is developed within this research project and is targeted for efficient CNN acceleration. The key functionalities and the basic design are described in a subsequent section.

The *Safe Island* is intended for dedicated safety tasks. The dual-core Cortex-R52 runs in lock-step operation with fault detection. Ultimately, the Safe Island orchestrates the safety features of the chip.

The Dual 20 MP Image Signal Processor (ISP) is used for image pre-processing of the four lane MIPI interface. The ISP is developed by Dream Chip according to the ISO26262:2018 standard and supports up to ASIL-B/D. A 24-bit hard-wire image pipeline with low energy consumption is used. The ISP requires no additional frame buffer which leads to a low latency.

The SoC provides multiple IO interfaces. High-speed communication is guaranteed by PCIe Gen3 and 1 Gbit Ethernet controllers. The LPDDR4 interface provides enough bandwidth to the data processing units on the SoC.

At the end of the project, we plan to tape-out the chip with GlobalFoundries’ 22nm FDX technology at 750 MHz. The platform should achieve a total of 10 TOPS with INT8 precision.

### B. Scalable-Precision Accelerator for Machine Learning

The ZuKIMo SoC incorporates an energy-efficient Scalable-Precision Accelerator for Machine Learning (SPA-ML) that is developed by TUD. The focus of the design is to efficiently support mixed-precision operands within the entire compute pipeline of SPA-ML. Therefore, a bit-serial architecture is developed. This bit-flexible methodology applies not only to the computational engines within the Neural Processing Unit (NPU) core for accelerating convolutions, pooling, and shortcut addition, but also to the memory alignment. SPA-ML consists, as shown in Figure 2, of a lightweight processor core with dedicated program memory, a closely coupled separate memory, DMAs, and the NPU Core. The integrated processor is an open-source implementation of the Rocket Chip RISC-V core. It is used to control the computation of a DNN and to fully support layer-fusion without hardware limitations [1].

The NPU core consists of a PE array and is optimized for 2D convolutions. Additionally, the NPU accelerates fully-connected layers. Piecewise linear approximation is supported, allowing for flexibility and a wide range of activation functions.

The memory scheme, in combination with a regular instruction set architecture, enables the exploitation of the regularity of convolutional layers and results in a small instruction footprint for entire CNNs. Moreover, these advantages also apply to the acceleration of the more general dilated and transposed convolution operations with efficient zero-skipping. Compared to other accelerators, SPA-ML efficiently computes these layers without additional hardware or restrictions to the supported layer parameters as detailed in [2].

### C. Software Tooling

To extract maximum performance and energy efficiency from the accelerator and SoC, modern and advanced tooling is implemented in the ZuKIMo project. The tooling includes AI optimization tool flows for the accelerator and a software pipeline for image processing integrated into a customized Linux operating system. To compile and optimize CNNs for the accelerator, an Apache TVM integration implements the translation of standard CNN models from the ONNX format to an executable format in C for the control flow and a YAML file as the layer sequence representation. In addition to the translation, various optimization steps, like layer fusion and data-reuse strategies, are integrated into the Apache TVM toolchain. A customized mainline Linux with an integrated

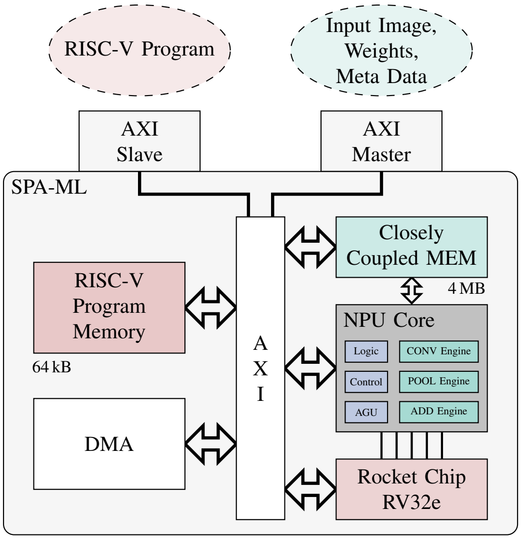

Fig. 2: Block diagram of the SPA-ML architecture of TUD.

gstreamer pipeline is deployed for image and video processing on the ARM Cortex-A65AE processor to enable efficient image processing.

### *D. ZuKIMo Use Cases*

As highlighted earlier, the ZuKIMo project is focused on edge scenarios, particularly autonomous driving and drone use cases. The core idea of developing a complete SoC with a custom accelerator and developer-friendly tooling, from design to fabrication, is meant to inspire such bottom-up, hands-on development projects at the national level, and potentially at the continent level. A corresponding ecosystem invites other key players in the German automotive industry to take part in using and further developing the tools and future iterations of the SoC down the line.

For the automotive use case, we started with the complex yet critical task of 3D object detection. To process LiDAR sensor data for this task, CNNs in this domain use 3D convolutions. These networks can be very demanding in terms of compute complexity, particularly on an edge device. To facilitate the efficient deployment of the AI-based task, a more hardware-aware approach was chosen, namely the PointPillars method. PointPillars creates 2D maps of the 3D point clouds, and relies only on standard 2D convolutions to process the data. We further apply uniform and flexible quantization techniques [3], [4] to extract execution advantages on our bit-flexible hardware accelerator. This is one example of the hardware-software co-design techniques applied in the ZuKIMo project.

## III. Development Methodology

The design of highly integrated electronic circuits with application-specific components, such as an AI accelerator, is characterized by a large number of degrees of freedom. In particular, the decisions to be made in the context of hardware/software co-design, such as the allocation of tasks to different hardware and software units, the memory hierarchy, or hardware-aware quantization techniques, all encompass a design space of considerable size.

For the specific case of AI accelerators, it is necessary to weigh different and conflicting optimization objectives against each other. The aim is to maximize the computing efficiency that can be achieved by the SoC while minimizing the power and energy consumption. Additionally, the safety requirements to be met make it necessary to integrate safety mechanisms, but these in turn can have a detrimental effect on the achievable computing performance and/or energy consumption. Therefore, to develop the best-fit AI accelerator architecture with respect to multi-objective optimization, it is essential to compare different implementation alternatives early in the design process and discard inferior alternatives. In particular, the components and mechanisms necessary to meet the safety requirements must be taken into account, as their integration can have a significant impact on the various optimization objectives.

We investigate methods and tools suitable for the early evaluation of implementation alternatives and automated exploration of the design space. A special focus is set on the aspects of analytical hardware-CNN co-design and early integration and optimization on a virtual platform. Finally, we give an overview on partitioning options for CNN inference on multiple computing nodes.

### *A. AnaCoNGA: Analytical Co-Design using Nested Genetic Algorithms*

Within the context of developing design methodologies to achieve hardware-software co-design, we investigated opportunities of creating design loops which parallely design the hardware *and* the software. The work in *AnaCoNGA*, proposed analytical hardware-CNN co-design using nested genetic algorithms (GAs) [5]. In detail, we investigated the design space of mixed-precision quantization for weights and activations over the layers of a CNN, as well as the design space of a scalable hardware accelerator. Generally, searching two design spaces takes longer than searching one, however, by nesting the design spaces in each other, we forced one design space (the hardware) to become a constraint on the other (the software, i.e. the CNN), thereby reducing the overall search time of the design methodology. This allowed a *fast* GA, which quickly designs optimal hardware based on fast hardware-model rewards, to act as a constraint on the *slow* outer GA which fine-tunes different CNNs with variable bit-widths across their layers and requires costly GPU-hours. With the novel nested design formulation, AnaCoNGA improved the accuracy of a ResNet20 by 2.88 p.p. compared to a uniform 2-bit quantized variant on CIFAR-10, and achieved a 35% and 37% improvement in latency and DRAM accesses, while reducing hardware LUT and BRAM resources by 9% and 59% respectively, when compared to a standard edge variant of the accelerator. The nested genetic algorithm formulation also reduced the search time by 51% compared to an equivalent, sequential co-design formulation.

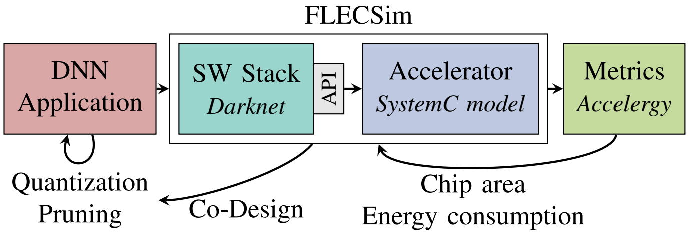

Fig. 3: Design flow of the DNN application and hardware accelerator co-design realized in FLECSim [6].

### B. FLECSim-SoC: A Flexible Co-Design Simulation Framework for System-on-Chips

With FLECSim we designed a framework for end-to-end simulation of an SoC with dedicated accelerators [6]. The simulation framework covers the main building blocks of SoCs such as the CPU and memory that can be co-simulated with a custom SystemC or RTL accelerator model. This allows an early system performance analysis and, by flexible configuration, enables exploration of the accelerator and memory design space. The framework is built around the virtual platform Imperas OVPSim [7], which provides the CPU models.

The complete design flow and co-design process of FLECSim is detailed in Figure 3. The input of FLECSim is a DNN application or model that may have already undergone optimizations in terms of quantization and pruning, e.g., with the design loops proposed by AnaCoNGA. The DNN model is then implemented on the respective software stack, i.e. the ML framework *Darknet* [8] in the FLECSim example. The convolutional layers are mapped on the accelerator model described either in SystemC or Verilog, and during execution of the simulation, traces and action counts are generated. This information can be used to estimate crucial design metrics such as dynamic energy consumption or chip area. For this purpose, FLECSim offers an interface to the open-source tool *Accelergy* [9]. By making the accelerator model configurable, e.g. in terms of number of compute units, buffer sizes and memory interface bandwidth, the design space of the SoC and the accelerator can be explored. All layers and operations not supported by the accelerator under test are executed in an instruction-accurate manner by OVPsim. This enables accurate end-to-end performance analysis of the entire SoC and closes the co-design loop for DNN optimization if latency or energy requirements are not met.

### C. Hardware-aware Partitioning of CNNs

The scalable and transferable architecture of the SPA-ML architecture allows integration in platforms of different performance categories. Thus, a small size accelerator can be used close to a sensor node, e.g. vision camera, or a large one in embedded HPC systems, e.g. centralized Advanced Driver Assistance Systems (ADAS). As depicted in Figure 4, the dimensional reduction characteristic of CNNs, especially in object recognition and semantic segmentation, offers an interesting potential for optimization when using distributed computing systems. When partitioning the CNN and computing on different computing nodes, sweet spots can be identified that help in reducing costly on-board communication over the respective link. In the ZuKIMo project, we systematically identified such sweet spots in common CNN architectures in our work [10], which complements our co-design solutions.

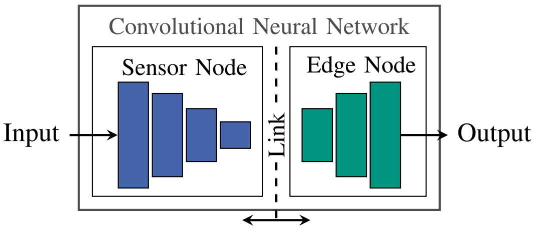

Fig. 4: The simulation toolchain presented in [10] evaluates different partitioning points of a CNN on distributed embedded computing nodes. The link layer transmits intermediate feature maps of the CNN.

## IV. Functional Safety for AI Platforms and DNN Resilience

ZuKIMo targets safety-critical applications. In this section, we present our work on improving the resilience of DNNs against faults and adversarial attacks. We study the effects of random hardware faults on the predictions of CNNs. Finally, we present a novel safety mechanism to support functional safety of AI accelerators.

### A. Adversarial Attacks and Fault Resilience

An important dimension to investigate the safety aspect of an AI deployment is to consider threats to the system due to hardware faults *and* potential software attacks. The field of adversarial attacks on DNNs garnered a lot of attention after it was proven that minimal, optimized perturbations to images can cause severe, high-confidence wrong classifications [11]. It is important to note that these perturbations are imperceptible to the human-eye, yet they strongly trigger paths in the network, with the intent to break its inference function.

As a consequence, another sub-field of research focused on training against such attacks using adversarial training techniques. A SoTA approach is Fast Adversarial Training (FastAT), proposed in [12]. Within the scope of this project, we investigated the fault resilience of FastAT-based networks to understand whether training for adversarial attacks had any effects on the performance of the DNN on hardware, in the presence of hardware faults, i.e. bit-flips. The results of this work were published in [13].

In summary, a network trained to be robust against adversarial attacks had significantly larger scaling factors during the quantization process, before deployment on edge hardware. The consequence of large scaling factors is that a single bit-flip in the value of weights and/or activations has a much larger impact on the DNN. Through theoretical analysis and extensive bit-flip injection testing, we showed that the failure

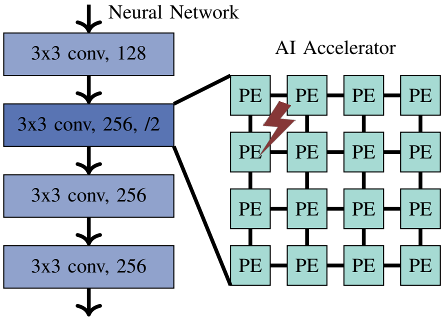

Fig. 5: A hybrid fault injection method to evaluate the influence of a fault on the result. Faults are applied precisely on a specific hardware module. Only calculations related to the fault are simulated with a cycle-accurate RTL simulation. All other CNN operations are performed without the hardware model.

rate was almost *doubled* in the presence of randomized bit-flips in hardware for an adversarially trained network compared to a vanilla trained one. To solve this problem and *maintain* robustness against adversarial attacks, we proposed training a FastAT DNN with high weight decay, in order to force the DNN's values to cover a smaller numerical distribution, thereby requiring smaller quantization scaling factors. With this approach we reduced the failure rate of an adversarially trained ResNet56 by 25% for large-scale bit-flip benchmarks on activation data, while gaining slightly improved accuracy and adversarial robustness.

### B. Fault Sensitivity Analysis for CNNs

Image classification or object detection are common applications of CNNs. The input image is processed by the CNN to extract features and give predictions on objects and bounding boxes. Under certain circumstances, the CNN can be disturbed by random hardware faults to the point that the predictions become false.

By nature, image data has a high degree of information redundancy. Adjacent pixels differ only slightly and contain correlated information. Typical low-level features in images are simple edges and line shapes. The low-level features can be aggregated to extract high-level, complex features which have semantic relevance.

CNNs learn to generalize during training by distributing the necessary information for extracting such features across several neurons. This instills a considerable amount of inherent redundancy in neural networks, and hence, a randomly occurring hardware fault might not always lead to a false prediction. To reduce the amount of redundant information, compression techniques such as pruning and quantization are often applied to reduce the network's memory footprint. From a functional safety perspective, the probability of false predictions due to hardware faults is critical knowledge.

Within ZuKIMo we investigate a CNN's behavior under random hardware faults. To this end, we develop a novel method of hardware fault simulation and analysis which overcomes the limitations of the state-of-the art. The method is not limited to analysis on a behavioral level, where faults are injected only to the algorithm. We consider the effects of a fault occurring in a hardware unit. Figure 5 demonstrates the approach, where we enable accurately targeting the fault injection on specific registers under test. With the targeted injection, we are not limited to data registers, but can also analyze faults in control logic. A fault is injected during the calculation of a CNN layer under test. The usage of a hardware model enables modeling the fault propagation in both hardware and the algorithm.

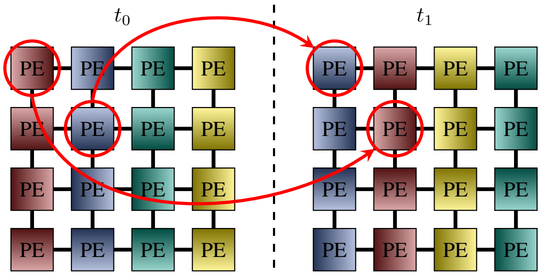

Fig. 6: A PE array with a time and spatial redundancy. The color represents the input of a column and the gradient the input of a row, respectively. In the first iteration weights and features are calculated in the default location. For the second iteration the inputs are permuted.

A complete cycle-accurate hardware simulation is extremely time-consuming, so simulating a complete CNN inference on a hardware accelerator can take hours to days. Hence this is not a feasible solution when it needs at least thousands of sample images to estimate a CNN's resilience against a fault. The approach developed in ZuKIMo uses fast DNN inference with calculations which are not affected by a fault. The hardware simulation on the register-transfer level (RTL) is limited to locations affected by the injected fault. The approach breaks down the problem by only simulating the affected loops of the CNN with respect to the hardware mapping, in a cycle-accurate manner. The rest of the inference is run in a standard ML framework. This hybrid methodology allows both a fast and accurate simulation of the fault behavior.

Our results reveal the different effects of faults on CNNs. There are CNN layers which are more resilient to faults than others, depending on the layer's properties and hardware mapping. Furthermore, we identify critical hardware registers in the accelerator PEs' data path and control logic.

### C. Efficient Safety Mechanism for AI Accelerators

Our CNN fault resilience analyses reveal varying degrees of fault sensitivity. We show that different layers can be more and less fault sensitive. Faults occurring during fault sensitive calculations are more likely to result in an incorrect prediction. On the other hand, there are layers and calculations with negligible influence, where faults do not affect the prediction result. Therefore, we investigate an adaptive safety mechanism.

Kempf et al. suggested runtime adaptive redundancy as a safety mechanism for a multi-core processor [14], [15]. The proposed safety mechanism can be seamlessly configured at runtime. In ZuKIMo the concept is transferred to a data-flow architecture such as the SPA-ML. The flexibility of enabling and disabling redundancy at runtime is able to protect fault sensitive layers and provide the maximum performance for the other layers. One major advantage of adaptive redundancy is the reuse of existing hardware. The high amount of parallelism in the hardware architecture is divided to either maximise the throughput or enable the safety mechanism.

Therefore, a novel approach is proposed and investigated. The approach combines time and spatial redundancy concepts where the advantages of both redundancy techniques are enhanced and the disadvantages are minimized. The time redundancy prevents common mode failures by calculating each operation twice with a delay of several clock cycles. Additionally, we perform the redundant calculations on different processing elements (spatial shift) which ensures a detection of permanent faults and faults in the data path to the PE. We call this approach a dynamic time and spatial redundancy concept which we depict in Figure 6.

Time redundancy is directly realized within the control logic of the NPU core. By re-execution of the internal microcode, the affected microinstructions of safety critical layers are repeated. This includes data loading (weights and features) from the memory, calculations, and the write back to the memory. Our safety mechanism ensures that the redundant calculations are done on different PEs as detailed in Figure 6. For the re-computation, the input data within the rows and columns are permuted. For the permutation we make use of the regular hardware structure. The PE array consists of an even number of PE rows and columns. Two neighboring rows and columns build a pair. For the re-computation the inputs of each pair is swapped. Swapping both rows and columns is used to prevent undetected permanent faults. Otherwise a permanent fault inside of the input data path could remain undetected.

Before the results are written back to the memory they are checked to be identical. Therefore, the result of the first calculation is not directly written to the memory, but temporarily stored for a later voting. The result of the second calculation is compared with the temporarily stored data and only if identical, the data is written to the memory. The presence of a fault triggers an interrupt and fault handling must be done by the superior instance.

The presented approach of time and spatial redundancy uses the advantages of both redundancy techniques. Permanent faults are detected and common mode failures are prevented. Disadvantages of both techniques are reduced to a minimum. Especially the hardware overhead of traditional spatial redundancy is minimized. The required safety mechanism is configured individually for each layer and changes during the execution. Depending on the fault sensitivity of a layer, redundancy is enabled or disabled. When faults are tolerable the complete PE array is utilized for maximum performance.

## V. Conclusion

The project ZuSE-KI-Mobil will deliver an SoC platform with the novel SPA-ML architecture and a software ecosystem. Partners from industry and academia work tightly together to fulfill the ambitious project goals and realize the tape-out planned in 2023. Besides a general overview, first intermediate project results are presented in this article with a particular focus on functional safety considerations and hardware-CNN co-design . In already published and planned publications, the consortium will continually report on specific aspects in greater detail.

## Acknowledgment

This work was funded by the German Federal Ministry of Education and Research (BMBF) under grant number 16ME0096 (ZuSE-KI-mobil). The responsibility for the content of this publication lies with the authors. Furthermore, we thank GlobalFoundries, Synopsys, Cadence, Arteris, and ARM for supporting this research.

## References

[1] S. Friedrich, R. Wittig, E. Matúš, and G. Fettweis, “Energy-based optimization for resource limited neural network accelerators with fused-layer support,” in *3 rd International Conference on Advances in Signal Processing and Artificial Intelligence (ASPAI’)*, Virtual Conference (Porto, Portugal), Nov 2021, pp. 41–45.

[2] S. Friedrich, S. Balamuthu Sampath, R. Wittig, M. Rohit Vemparala, N. Fasfous, E. Matúš, W. Stechele, and G. Fettweis, “Lightweight instruction set for flexible dilated convolutions and mixed-precision operands,” in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, April 2023, to be published.

[3] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, “PACT: Parameterized clipping activation for quantized neural networks,” *ArXiv*, vol. abs/1805.06085, 2018.

[4] N. Fasfous, M. R. Vemparala, A. Frickenstein, E. Valpreda, D. Salihu, N. A. V. Doan, C. Unger, N. S. Nagaraja, M. Martina, and W. Stechele, “Hw-flowq: A multi-abstraction level hw-cnn co-design quantization methodology,” *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5s, sep 2021.

[5] N. Fasfous, M. R. Vemparala, A. Frickenstein, E. Valpreda, D. Salihu, J. Höfer, A. Singh, N.-S. Nagaraja, H.-J. Voegel, N. A. Vu Doan, M. Martina, J. Becker, and W. Stechele, “Anaconga: Analytical hw-cnn co-design using nested genetic algorithms,” in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 238–243.

[6] T. Hotfilter, J. Hoefer, F. Kreß, F. Kempf, and J. Becker, “Flecsim-soc: A flexible end-to-end co-design simulation framework for system on chips,” in *2021 IEEE 34th International System-on-Chip Conference (SOCC)*, 2021.

[7] Imperas, “OVP: Fast simulation, free open source models, public APIs: Open virtual platforms.” [Online]. Available: https://www.ovpworld.org/

[8] J. Redmon, “Darknet: Open source neural networks in c,” http://pjreddie.com/darknet/, 2013–2016.

[9] Y. N. Wu, J. S. Emer, and V. Sze, “Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs,” in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2019.

[10] F. Kreß, J. Hoefer, T. Hotfilter, I. Walter, V. Sidorenko, T. Harbaum, and J. Becker, “Hardware-aware partitioning of convolutional neural network inference for embedded ai applications,” in *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*.

[11] I. j. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and Harnessing Adversarial Examples,” in *ICLR*, 2015.

[12] E. Wong, L. Rice, and J. Z. Kolter, “Fast is better than free: Revisiting adversarial training,” in *ICLR*, 2020.

[13] N. Fasfous, L. Frickenstein, M. Neumeier, M. R. Vemparala, A. Frickenstein, E. Valpreda, M. Martina, and W. Stechele, “Mind the scaling factors: Resilience analysis of quantized adversarially robust cnns,” in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022.

[14] F. Kempf, T. Hartmann, S. Baehr, and J. Becker, “An adaptive lockstep architecture for mixed-criticality systems,” in *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2021.

[15] F. Kempf, J. Hoefer, F. Kreß, T. Hotfilter, T. Harbaum, and J. Becker, “Runtime adaptive cache checkpointing for risc multi-core processors,” in *2022 IEEE 35th International System-on-Chip Conference (SOCC)*, 2022.