Multi-partner Project: Open-source Design Tools for Co-development of AI Algorithms and AI Chips

(Initial Stage)

Mehdi Tahoori[1], Jürgen Becker[1], Jörg Henkel[1], Wolfgang Kunz[4],
Ulf Schlichtmann[2], Georg Sigl[2], Jürgen Teich[3], Norbert Wehn[4]
[1]Karlsruhe Institut of Technology, Karlsruhe, Germany, [2]Technical University of Munich, Munich, Germany,
[3]Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany,
[4]University Kaiserslautern-Landau, Kaiserslautern, Germany

Abstract— **Chip technologies are crucial for the digital transformation of industry and society. Artificial Intelligence (AI) is playing an increasingly important role in both our daily lives and in industry. The development of advanced AI chip designs, essential for the successful deployment of AI, is of critical importance for innovation and competitiveness. However, challenges arise from the complexity of hardware development, expensive access to state-of-the-art design tools, and a global shortage of hardware experts. In addition to cost optimization, computational power, and energy consumption, security and trustworthiness are becoming increasingly important. This project aims to address these challenges in AI chip design by enabling efficient hardware development. We are developing a seamless transition between software-based AI model development and optimization, and efficient hardware implementation, while considering security, trustworthiness, and energy efficiency. An open-source approach plays a key role, facilitating access for small and medium-sized enterprises (SMEs) and expanding the community involved in AI chip design to help mitigate the shortage of skilled professionals.**

Keywords—Algorithm-Hardware Co-design, Artificial Intelligence, AI accelerator, open-source hardware, Electronic Design Automation

I. Introduction

Chip technologies are essential for the digital transformation of both industry and society. As artificial intelligence (AI) plays an increasingly central role in everyday life and industrial applications, the development of cutting-edge AI chip designs has become crucial for driving innovation and maintaining global competitiveness. However, the advancement of hardware design faces significant challenges, particularly due to the slow evolution of hardware tools and languages in comparison to their software counterparts. This growing disparity between hardware and software development hinders the progress needed for the creation of next-generation AI chips and exacerbates the difficulty in attracting and training the future workforce required to meet these demands.

To overcome these obstacles, there is a pressing need to rethink existing electronic design automation (EDA) methodologies and algorithms, leveraging AI itself to design and optimize AI hardware. By developing powerful, secure, and trustworthy AI chips, exponential technological advancements can be realized. This project seeks to close the gaps in AI chip design and deliver sustainable, reliable AI chips and systems. We propose creating a seamless workflow that integrates software-based AI model development and optimization with the corresponding hardware mapping and optimization, specifically tailored to meet application constraints.

A key element of this effort is the use of open-source tools, AI-based toolchains, and system architectures to strengthen the German research landscape. By providing small and medium-sized enterprises (SMEs) with turnkey solutions for AI hardware, the project will help establish Germany as an internationally competitive force in AI hardware design. The primary objectives of this project are to:

- Develop a trustworthy, automated workflow that spans from AI algorithm development to AI hardware architectures, optimized for specific use cases, ensuring functional correctness, safety, and security across the entire AI software-to-hardware pipeline.
- Train and engage the next generation of researchers and professionals in AI chip design, addressing the skills shortage in this critical field.
- Create a sustainable ecosystem for AI system design, based on open-source and AI-driven solutions, with a focus on building a strong community, expanding the user base, and developing viable business models that primarily support SMEs.

Our approach emphasizes an automated flow that bridges the gap between AI model optimization in software and AI hardware mapping, incorporating algorithm-hardware co-design. This method combines AI algorithm optimization with hardware design space exploration, taking into account specific hardware requirements and optimization goals. We will develop various AI hardware accelerators and architectures, which will be automatically selected and optimized to meet the user requirements for specific applications. Ensuring functional correctness, safety, and security will be integral to the entire process, from AI model design to hardware mapping.

Led by a team of internationally recognized EDA experts from top German universities, this project aims to foster long-term sustainability and support for these open-source tools, building a vibrant community and facilitating technology transfer to industry, particularly SMEs. Ultimately, the project will provide a powerful and adaptable AI platform that addresses the needs of German industry and application domains such as automotive and industrial automation. By doing so, this initiative will promote innovation, competitiveness, and technological sovereignty in Germany and Europe, ensuring that AI serves as a driving force for a thriving industrial future.

II. Preliminaries and Related Work

A. AI Hardware Blocks and Architectures

Recent advancements in AI hardware architecture have been the focus of both industry and academia, with numerous approaches continuously being explored and developed.

These efforts have included techniques such as custom accelerators, systolic arrays, Neural Processing Units (NPUs), and specialized hardware designs aimed at improving the performance and speed of AI systems [BAV22, TAL21]. One of the most significant developments has been in memory-based computation (Computation-in-Memory - CiM), leveraging technologies like Resistive-RAM (RRAM), Magnet-RAM (MRAM), and Phase-Change Memory (PCM) [SEB20, YU21]. These innovations provide faster data access and efficient processing, leading to significant gains in the overall performance and efficiency of AI systems.

Energy-efficient techniques such as quantization and sparsity have also been critical in reducing computational complexity and optimizing energy consumption, enhancing the efficiency of AI systems while minimizing their environmental impact. Additionally, FPGA-based AI accelerators have gained popularity due to their versatility and adaptability. These accelerators allow for rapid prototyping and implementation of AI algorithms, providing optimal performance across a wide range of applications [UMU17]. Despite these advancements, the automated generation and optimization of AI hardware blocks remain underdeveloped, still requiring manual effort to meet area, performance, and energy constraints. Furthermore, no open-source automation methods or tools are currently available for these tasks.

B. *Co-Design of Hardware and Software for AI and ML Compilers*

Embedded systems, in particular, face stringent resource constraints (e.g., memory, power consumption, computational capacity), making it challenging to implement AI models while meeting strict application requirements. Often, AI models are too complex in terms of the number of required operations or parameters and must be compressed or reduced to fit the limited memory and processing capabilities of target platforms. To efficiently explore the vast search space of compression techniques (such as pruning and quantization), evolutionary algorithms [HEI22] and methods based on explainable AI (XAI) [SAB20] have been proposed. While these approaches primarily target GPUs, they also show promise for co-designing AI applications and hardware architectures. For example, XAI-based methods have been used to design robust and resource-efficient binary neural networks [SAB23].

Early Neural Architecture Search (NAS) approaches [ZOP16, BAK17] utilized reinforcement learning to guide the search process, but they were highly compute-intensive. To address this complexity, differentiable NAS [LIU19] and hybrid approaches combining both techniques [VAH20] have been proposed. Existing hardware-aware NAS and AutoML solutions focus on automating the design and optimization of neural network architectures while considering hardware-specific constraints [BEN21]. These methods aim to find neural network structures optimized for target hardware platforms, resulting in improved performance and energy efficiency [ZHA20, WU19]. AutoML techniques integrated into hardware-aware NAS strive to automate the entire process from architecture search to model deployment, accounting for hardware limitations and requirements [GUP20].

In the field of machine learning (ML) compilers, both open-source frameworks such as Apache TVM, TFLite, and Glow [ROT19], and commercial solutions like X-CUBE-AI (STMicroelectronics) and ARM's CMSIS-NN, exist. Academic frameworks [Al21, Ji20, Ji21] have also been developed, but they all require significant manual effort to support new AI hardware architectures. To address this challenge, we aim to develop an open-source approach based on TVM that can be quickly configured for various AI hardware architectures.

C. *CAD for Correctness, Security, and Functional Safety of AI Hardware and Systems*

AI is now seamlessly integrated into Systems-on-Chip (SoCs), with AI applications often relying on accelerators to achieve optimal performance. Security is a critical concern when integrating AI accelerators, as protecting the confidentiality of neural network models and associated intellectual property is paramount. Securing these accelerators against adversarial attacks and ensuring user privacy on shared accelerators have become major research topics [WE18]. Specifically, executing AI on edge devices exposes these systems to hardware attacks, where attackers can gain physical access and target the intellectual property embedded in the AI algorithms. Research has shown that ML models, especially neural networks, can be extracted through side-channel attacks, allowing attackers to obtain copies of proprietary AI models [DON19, BAT19].

Both physical and remote power attacks pose serious threats, as they enable the extraction of proprietary neural network parameters. For instance, [YA18] demonstrated how microarchitectural timing could be exploited to extract network architectures and parameters, while [XI20] leveraged power side-channels to expose the model parameters. These attacks take advantage of the specific execution patterns within neural network layers and the data flow between neurons, revealing key architectural details [BRO22]. In addition to side-channel attacks, fault injections have been shown to induce incorrect classifications or extract information about the AI algorithm [LIU17, BRE20].

To counter these threats, countermeasures such as masking [DUB20, ATH22] and shuffling within neural networks have been proposed, with shuffling emerging as a new research direction [BRO22]. Although methods for verifying secure implementations exist [HAD21], they suffer from poor scalability for large designs and are limited to evaluating masked implementations. Additionally, current approaches lack the capability to show where mask reuse is feasible, a critical factor for AI algorithms with a large number of parameters. At the microarchitectural level, no established verification methods currently exist to support the design of such countermeasures. Therefore, one of our goals is to develop formal verification techniques that systematically ensure the security of AI SoCs.

As AI algorithms are increasingly deployed in safety-critical systems, reliability and functional safety are becoming key concerns, affecting not only the algorithms themselves but also the underlying processors and their resilience to random hardware faults. For example, Tesla's Full Self-Driving (FSD) chip incorporates functional safety through redundancy, with two independent FSD chip instances running separate operating systems [TAL20]. However, this level of hardware redundancy is costly and limits performance and accuracy. Therefore, efforts are being made to quantify the inherent fault tolerance of neural networks at various abstraction levels, such as in ML frameworks, hardware modeling [GUA17], and register-transfer level designs [BEH18].

III. EDAI Approach

In response to the growing complexity and demand for trustworthy and efficient AI hardware, the EDAI project aims to develop advanced open-source AI hardware solutions that address critical gaps in current technology. As there are significant challenges in bridging the hardware-software divide and ensuring reliable AI hardware, and there are limitations in automation, security, and functional safety, EDAI takes a comprehensive approach to tackle these issues head-on.

The research method of the EDAI project is structured into four key pillars, as represented in Figure 1. The first pillar focuses on creating open-source AI hardware blocks and architectures with integrated functional safety features. The second pillar deals with the co-design of AI algorithms and hardware architectures, facilitating seamless compilation and synthesis. The third pillar addresses safety and functional safety, ensuring secure and reliable AI hardware systems. Finally, the fourth pillar encompasses the open-source release, along with the long-term support and maintenance of these hardware solutions. These pillars together aim to close the identified gaps and provide a robust framework for AI hardware development.

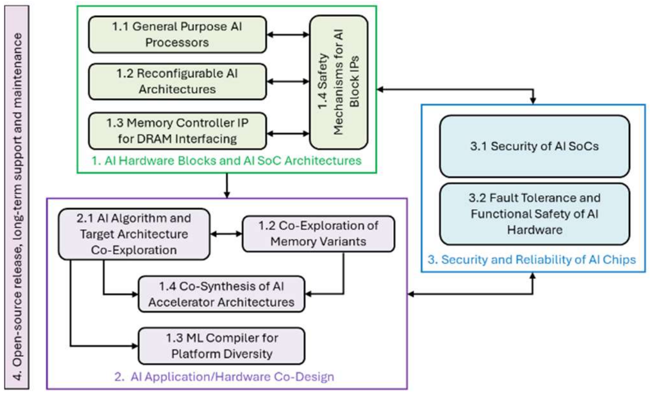
4. Open-source release, long-term support and maintenance
1.1 General Purpose AI Processors
1.2 Reconfigurable AI Architectures
1.3 Memory Controller IP for DRAM Interfacing
1.4 Safety Mechanisms for AI Block IPs
1. AI Hardware Blocks and AI SoC Architectures
3.1 Security of AI SoCs
3.2 Fault Tolerance and Functional Safety of AI Hardware
3. Security and Reliability of AI Chips
2.1 AI Algorithm and Target Architecture Co-Exploration
1.2 Co-Exploration of Memory Variants
1.4 Co-Synthesis of AI Accelerator Architectures
1.3 ML Compiler for Platform Diversity
2. AI Application/Hardware Co-Design

Figure 1: The pillars of the EDAI project and their interactions.

A. *Open-Source AI Hardware Blocks and AI SoC Architectures*

The first research pillar is dedicated to the development of a range of open-source AI hardware IP blocks and architectures that will be utilized both in the subsequent pillars and in future projects. These hardware components are designed to accelerate ML applications and address the growing demand for customizable and efficient AI systems.

AI Extensions for General-Purpose Processors based on RISC-V

This part of the project aims to identify and develop hardware modules (IP blocks) specifically for accelerating ML applications in processors with extensible instruction sets. The research focuses on three classes of IP blocks: (a) arithmetic functional units for highly quantized neural networks, supporting low-precision arithmetic operations like floating-point (16/8-bit) or integer arithmetic (4/2/1-bit), enabling multiple calculations in vector units simultaneously. Additionally, the project will develop related units for (de)packing, permutation, reduction, and data type conversion, particularly for neural networks utilizing mixed precision. (b) Customized memory components, such as codebooks for quantization and pruning, and buffers with address generators to enhance data locality spatially and temporally. (c) Specialized functional units for efficient execution of activation functions (ReLU, tanh, sigmoid) commonly used in Long Short-Term Memory (LSTM) networks. The project will explore various design trade-offs such as cost and accuracy, including approaches like Taylor series, piecewise linear functions, or lookup tables.

The overall goal is to integrate these domain-specific blocks into existing open-source microprocessor architectures, especially RISC-V, as generic extensions via HDL generators or templates. The project will ensure interoperability with open-source compilers for instruction set extensions and synthesis tools like YoSys for RTL synthesis of IP blocks. Projects like Google's CFU Playground and the PULP platform will serve as foundational resources for these developments.

Reconfigurable AI Architectures

In this part of the pillar, the focus will be on developing reconfigurable AI modules using FPGAs and integrating them with ARM- and RISC-V-based processor cores and SoCs. Commercial FPGA-SoCs typically include ARM CPUs as ASIC elements, which achieve higher clock frequencies compared to programmable FPGA logic. While the high clock frequency is advantageous for control tasks, FPGA logic compensates for its relatively slower speed through high parallelism and distributed local memory. This cost-effective approach allows for prototyping and potential product implementation without the high manufacturing costs of ASICs. However, the challenge lies in minimizing the latency and overhead from coarse-grained CPU-to-FPGA communication. To address this, the project will develop larger compute units (combinations of AI IPs) and an automatic framework to enable efficient execution without frequent synchronization with the CPU/software.

Additionally, the project will work on runtime reconfiguration of FPGAs to dynamically load AI IP blocks as needed, optimizing resource usage. A RISC-V prototype in FPGA logic will be developed with extensible commands for AI accelerators, supporting inference and retraining of neural networks. This RISC-V implementation will serve as a prototype for ASIC implementation with embedded FPGA (eFPGA). One specific application will be the implementation of Content Addressable Memories (CAM) for hyperdimensional computing (HDC), a brain-inspired AI method. The automated workflow will enable HDC models to be trained and mapped onto FPGAs, providing adaptable hypervectors. The project will also extend the open-source platform FINN, developed by Xilinx for Quantized Neural Networks (QNNs), to support popular long short-term memory (LSTM) networks, while integrating closer with RISC-V or ARM processors for runtime reconfiguration.

Memory Controller IP for DRAM Interfacing

ML algorithms, particularly for larger models, require significant memory during both training and inference. On-chip memory, typically based on SRAMs, is often insufficient, requiring data storage in external memories (DRAM), which presents significant challenges in terms of bandwidth, latency, and energy consumption. DRAM accesses quickly become the bottleneck in both performance and energy efficiency. The memory controller plays a crucial role in determining effective memory bandwidth and energy efficiency, with scheduling strategies, address mapping, and refresh strategies being key factors.

In contrast to general-purpose memory controller IPs, this project adopts an application-specific approach, optimizing scheduling, refresh, and address-mapping strategies for specific applications. This will result in simpler controllers with lower complexity. The project will develop a parametrizable memory controller architecture template that can be configured based on the target applications to optimize effective memory bandwidth and energy efficiency. Additionally, the project will integrate encryption capabilities in the memory controller, ensuring that data is securely stored in DRAM and protected against external attacks.

Safety Mechanisms for Functional Safety in AI Block IPs

The final part of this research pillar focuses on extending AI block IPs to include safety mechanisms to ensure functional safety against random hardware faults. This task requires a thorough analysis and enhancement of all three classes of IP blocks (AI extensions for processors based on RISC-V, reconfigurable AI units, and memory) developed within this research pillar. First, relevant fault classes will be defined, and their impacts on individual modules during AI workload execution will be documented. This approach will enable the identification of appropriate safety mechanisms for the modules and their interfaces, as addressing faults depends on their effect on the overall system.

It is essential to differentiate between faults that affect control flow—potentially leading to incorrect behavior—and faults that only impact arithmetic results. The latter may prove less critical due to inherent redundancies in neural networks and can, therefore, tolerate lower reliability requirements. Additionally, in preparation for Research Pillar 3, it is necessary to identify module classes with similar structures, such as configuration registers and state machines. For each of these module classes, configurable and scalable safety mechanisms will be designed to meet varying reliability requirements. Based on the required reliability levels for each module, the IP library will be completed, providing a secure and reliable foundation for the implementation of AI workloads. This ensures that the developed AI hardware IPs can meet the functional safety demands critical in many application domains.

B. AI Application/Hardware Co-Design

Building on the AI hardware IP blocks and SoC architectures developed in the first research pillar, this second research pillar addresses the challenge of systematically optimizing AI network architectures and their efficient hardware implementations under given non-functional constraints. The goal is to automate this process while considering factors like hardware costs, latency, energy consumption, and robustness, as well as safety and security aspects, which are further detailed in Research Pillar 3.

AI Algorithm and Target Architecture Co-Exploration

The co-design of AI applications and hardware focuses on selecting an optimal target architecture (e.g., RISC-V with hardware extensions or SoC architectures) for one or more AI applications while optimizing non-functional properties such as hardware cost, latency, energy efficiency, and AI accuracy. Additionally, safety and security considerations are integrated into the exploration (as detailed in Research Pillar 3).

The exploration space is vast, encompassing both algorithmic optimizations (network topology, compression, network architecture search) and highly parametric hardware extensions and memory configurations (as developed in Research Pillar 1). To efficiently explore this design space, multi-objective optimization methods will be investigated, including techniques for compressing models using Explainable AI (XAI) and Neural Architecture Search (NAS). We aim to develop an iterative process where one component (e.g., hardware architecture) is fixed while the other (e.g., the neural network) is optimized, alternating between them to identify bottlenecks related to latency or energy consumption. This approach, referred to as "Co-Search," will yield a methodology and software tool to efficiently explore, evaluate, and automate the determination of Pareto-optimal solutions for AI applications and hardware architectures.

Co-Exploration of Memory Variants

As mentioned in Research Pillar 1, DRAM memory access often becomes a bottleneck in large AI models. This task integrates DRAM subsystem exploration into the AI accelerator design space. We will examine the memory controller strategies developed in Pillar 1 and evaluate their impacts when using different DRAM technologies (e.g., DDRx, LPDDRx, HBMx, GDDRx) for various applications. An initial methodology, which selects optimal address mappings based on application profiles, will be further developed and incorporated into the overall exploration framework to ensure optimal memory controller strategies for AI workloads.

ML Compiler for Platform Diversity

Mapping optimized AI workloads onto specific hardware platforms is a critical challenge in the design of optimized HW/SW-based systems. Typically, AI workloads are described in the form of trained neural networks. Executing these networks on specialized hardware (e.g., RISC-V cores with instruction set extensions or dedicated accelerators) is significantly more efficient than running them on general-purpose hardware (e.g., CPUs, GPUs). However, designing dedicated hardware accelerators for a specific neural network involves considerable design effort. This requires not only the hardware but also a suitable deployment toolchain (ML compiler) that can generate inference code for the hardware platform based on the explored neural networks.

We plan to develop an open-source approach that optimally maps AI workloads to both existing and project-specific hardware accelerators. RISC-V CPUs allow for close coupling of accelerators through instruction set extensions, as explored in Research Pillar 1. We will also address "TinyML" challenges—running AI algorithms under extreme resource constraints, such as in battery-powered sensors or devices that operate using energy harvested from the environment.

Co-Synthesis of AI Accelerator Architectures

Once a design point consisting of a compressed AI model and optimized AI hardware architecture is selected through co-exploration, the next step is to synthesize the AI hardware. Many AI accelerators rely on CiM techniques using DRAM or memristors, as well as a mix of digital and analog components. However, existing EDA and synthesis tools, which primarily target digital logic, are not equipped to handle and optimize these mixed-mode architectures. This will ensure that AI applications can be efficiently synthesized and deployed on cutting-edge hardware architectures, maximizing performance while meeting non-functional constraints.

C. Security and Reliability of AI Chips

Building on the microarchitectures explored in Research Pillar 1 and the design methodologies developed in Research Pillar 2, we focus on the critical analysis and assurance of security and reliability in AI accelerators. These non-functional properties are currently inadequately addressed in global AI hardware flows. AI chips not only face challenges regarding data privacy but also with respect to the protection against model theft. AI models deployed on HW accelerators often involve substantial training costs. Therefore, protecting the model from unauthorized replication is crucial. Through novel approaches to identify and mitigate security vulnerabilities, analyze safety properties, and enhance fault tolerance, this research pillar aims to address a significant gap in the global state of the art.

Security of AI SoCs

AI Systems-on-Chips (SoCs) integrate domain-specific hardware modules, including accelerators for arithmetic operations and specialized memory components. This subsection analyzes the security properties of these architectures, identifies specific vulnerabilities, and proposes mitigation strategies. Several "threat models" will be explored to address these risks.

One major concern is the violation of data obliviousness by the hardware. It is critical to ensure that runtime behavior, resource utilization, and memory access patterns of the hardware modules are independent of the deployed model's parameters and structure so that attackers cannot reverse-engineer the model. For instance, the timing of computations in an AI accelerator must not leak information about the weights and biases of the neural network. Another threat involves the potential manipulation of hardware modules in AI SoCs through backdoors or trojans. Furthermore, improper integration of hardware modules into SoC communication structures can create vulnerabilities that expose proprietary information about the AI accelerator's architecture and parameters. To counter these threats, the threat models will be formalized, and new formal verification techniques will be developed to reliably detect security violations.

In addition to addressing data obliviousness at the microarchitectural level, it is essential to secure AI accelerators also against other side-channel attacks, which could compromise the confidentiality of the AI algorithm or the data being processed. Data obliviousness mitigates timing-based side-channel attacks, but other forms of side-channel attacks—such as those based on power consumption or electromagnetic emissions—remain a concern. The required analyses can be performed using simulations based on RTL models. These models will be stimulated with input data that may trigger side-channel leaks, with switching behavior recorded to generate virtual power measurements. This data will be used to identify any data-dependent leaks. The identification of critical signals will be automated, allowing designers to iteratively adapt the design to eliminate side-channel vulnerabilities. The effectiveness of these methods will be validated through experiments on an appropriate FPGA platform.

These simulation-based methods complement formal verification techniques for AI-specific threat models. The combination of simulation and formal verification will provide security guarantees for both functional behavior and side-channel resistance in AI chips. The research outcomes, including Verification IPs, will be made open-source and published in standard languages such as SystemVerilog Assertions (SVA). The analysis methods will be designed to be tool-agnostic, and open-source RISC-V platforms, such as Pulpissimo and OpenTitan, will be used as the experimental foundation to ensure the reproducibility of results.

Fault Tolerance and Functional Safety of AI Hardware

While addressing security concerns, it is equally important to ensure the functional safety of AI hardware, particularly in safety-critical applications. This subsection focuses on protecting AI hardware against random errors that could lead to critical misclassifications and faulty predictions. AI accelerators differ fundamentally from traditional von Neumann architectures and therefore require specialized reliability techniques. Neural networks, if trained and structured explicitly for fault tolerance, can inherently manage expected errors, making them resilient to hardware faults.

This research aims to accelerate the design of reliable AI hardware. The IP and AI block-based design described in Research Pillar 1 will be extended to incorporate necessary safety mechanisms. Modern Hardware Description Language (HDL) generators, such as Chisel, which use an object-oriented approach based on Scala and are used in the design of Network Processing Units (NPUs) like Google's Edge-TPU, will be enhanced with integrated safety mechanisms to automatically generate synthesizable RTL code. Open-source RISC-V platforms, along with NPU extensions such as Chipyard, will serve as the underlying hardware architecture. The resulting IP blocks will be tested in simulation environments that support fault injection and inference partitioning. These tests will evaluate the resilience of AI algorithms to various error classes, and evaluations will be conducted on different hardware abstraction levels depending on the complexity of the models.

The plan is to develop both passive and active redundancy techniques to improve the reliability of AI hardware in the presence of field errors. In addition, the project will focus on enhancing the availability of neural networks, addressing retention and aging errors, and ensuring rapid system repair to maintain acceptable levels of accuracy. The system will be designed and trained to minimize the impact of runtime errors on inference accuracy. A lightweight AI-based runtime monitoring system will be integrated with the primary neural network to detect, locate, and predict catastrophic failures. This will be complemented by a compressed backup model to ensure functional safety and system availability.

IV. Conclusions

The EDAI project represents a comprehensive effort to advance the state of artificial intelligence hardware through a multi-faceted research approach that spans security, reliability, and hardware-software co-design. By integrating cutting-edge methodologies in AI architecture, this project addresses critical challenges such as fault tolerance, functional safety, and security vulnerabilities inherent in AI accelerators. Through the exploration of innovative design frameworks and verification techniques, the project aims to create robust AI chips that not only meet the demanding performance requirements of modern applications but also ensure the integrity and reliability of the underlying hardware.

By focusing on the intricate relationship between AI algorithms and their hardware implementations, the EDAI project seeks to enhance the efficiency of AI workloads while safeguarding against potential threats. The research conducted within this project will culminate in the development of open-source tools, methodologies, and frameworks that will not only benefit academic research but also have a significant impact on industry practices. Ultimately, the EDAI project aims to pave the way for more secure, reliable, and efficient

AI systems, addressing the evolving needs of the digital landscape while fostering innovation in hardware design.

ACKNOWLEDGEMENT

This project is funded by the German Federal Ministry of Education and Research (BMBF) in the framework of design tools for sovereign chip development with open source (DE:Sign).

REFERENCES

[AL21] A. Burrello et al. 2021. Dory: Automatic end-to-end deployment of realworld dnns on low-cost iot mcus. IEEE Trans. Comput. 70, 8 (2021), 1253–1268.

[ATH22] K. Athanasiou, T. Wahl, A. Ding, and Y. Fei. 2022. Masking Feedforward Neural Networks Against Power Analysis Attacks. Proceedings on Privacy Enhancing Technologies 2022, 1 (2022), 501–521.

[BAK17] B. Baker, O. Gupta, N. Naik, and R. Raskar. "Designing Neural Network Architectures using Reinforcement Learning", In Proceedings of the International Conference on Learning Representations (ICLR). 2017. URL: https://openreview.net/forum?id=S1c2cvqee.

[BAT19] L. Batina, S. Bhasin, D. Jap, and S. Picek. 2019. CSI NN: reverse engineering of neural network architectures through electromagnetic side channel. In Proceedings of the 28th USENIX Conference on Security Symposium (SEC'19). USENIX Association, USA, 515–532.

[BAV22] S. Bavikadi, A. Dhavlle, A. Ganguly, A. Haridass, H. Hendy, C. Merkel, V. J. Reddi, P. R. Sutradhar, A. Joseph, and S. M. P. Dinakarrao. 2022. A survey on machine learning accelerators and evolutionary hardware platforms. IEEE Design & Test, 39(3), pp.91-116.

[BEH18] B. Salami, O. S. Unsal, and A. C. Kestelman. 2018. On the Resilience of RTL NN Accelerators: Fault Characterization and Mitigation. In 2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD). 322–329.

[BEN21] H. Benmeziane, K. El Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang. 2021, August. Hardware-Aware Neural Architecture Search: Survey and Taxonomy. In IJCAI (pp. 4322-4329).

[BRE20] J. Breier, D. Jap, X. Hou, S. Bhasin, and Y. Liu. 2020. SNIFF: Reverse Engineering of Neural Networks with Fault Attacks.

[BRO22] M. Brosch, M. Probst, and G. Sigl. 2022. Counteract Side-Channel Analysis of Neural Networks by Shuffling. In 2022 Design, Automation and Test in Europe Conference and Exhibition (DATE). IEEE, Antwerp, Belgium.

[DON19] G. Dong, P. Wang, P. Chen, R. Gu, and H. Hu. 2019. Floating-Point Multiplication Timing Attack on Deep Neural Network. 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), 2019, pp. 155-161.

[DUB20] A. Dubey, R. Cammarota, and A. Aysu. 2020. BoMaNet: boolean masking of an entire neural network. In Proceedings of the 39th International Conference on Computer-Aided Design (ICCAD '20). Association for Computing Machinery, New York, NY, USA, Article 51, 1–9.

[GUA17] G. Li, et al. 2017. Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications. In SC17: International Conference for High Performance Computing, Networking, Storage and Analysis. 1–12.

[GUP20] S. Gupta and B. Akin. 2020. Accelerator-aware neural network design using AutoML. arXiv preprint arXiv:2003.02838.

[HAD21] V. Hadžić and R. Bloem. 2021. COCOALMA: A Versatile Masking Verifier. In 2021 Conference on Formal Methods in Computer-aided Design (FMCAD).

[HEI22] C. Heidorn, N. Meyerhöfer, C. Schinabeck, F. Hannig, and J. Teich. 2022. Hardware-Aware Evolutionary Filter Pruning. In Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS).

[HIP] "Hipacc: A Domain-Specific Language and Compiler for Image Processing", https://hipacc-lang.org

[Ji20] J. Lin, W. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han. 2020. Mcunet: Tiny deep learning on iot devices. arXiv preprint arXiv:2007.10319 (2020).

[Ji21] J. Lin, et al. 2021. Mcunetv2: Memory-efficient patch-based inference for tiny deep learning. arXiv preprint arXiv:2110.15352 (2021).

[LIU17] Y. Liu, L. Wei, B. Luo, and Q. Xu. 2017. Fault injection attack on deep neural network. 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2017, pp. 131-138.

[LIU19] H. Liu, K. Simonyan, and Y. Yang. 2019. DARTS: Differentiable Architecture Search. In Proceedings of the International Conference on Learning Representations (ICLR). URL: https://openreview.net/forum?id=S1eYHoC5FX.

[ROT19] N. Rotem, J. Fix, S. Abdulrasool, G. Catron, S. Deng, R. Dzhabarov, N. Gibson, J. Hegeman, M. Lele, R. Levenstein, J. Montgomery, B. Maher, S. Nadathur, J. Olesen, J. Park, and A. Rakhov. 2019. Glow: Graph Lowering Compiler Techniques for Neural Networks. arXiv preprint arXiv:1805.00907v3.

[SAB20] M. Sabih, F. Hannig, and J. Teich. 2020. Utilizing Explainable AI for Quantization and Pruning of Deep Neural Networks. arXiv preprint arXiv:2008.09072.

[SAB23] M. Sabih, M. Yayla, F. Hannig, J. Teich, and J.-J. Chen. 2023. Robust and Tiny Binary Neural Networks using Gradient-based Explainability Methods. In Proceedings of EuroMLSys@EuroSys 2023.

[SEB20] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou. 2020. Memory devices and applications for in-memory computing. Nature nanotechnology, 15(7), pp. 529-544.

[TAL20] E. Talpes et al., Compute Solution for Tesla's Full Self-Driving Computer, in IEEE Micro, vol. 40, no. 2, pp. 25-35, 1 March-April 2020, doi: 10.1109/MM.2020.2975764.

[TAL21] M. A. Talib, S. Majzoub, Q. Nasir, and D. Jamal. 2021. A systematic literature review on hardware implementation of artificial intelligence algorithms. The Journal of Supercomputing, 77, pp.1897-1938.

[UMU17] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers. 2017. Finn: A framework for fast, scalable binarized neural network inference. In Proceedings of the 2017 ACM/SIGDA international symposium on field-programmable gate arrays (pp. 65-74).

[VAH20] A. Vahdat, A. Mallya, M. Liu, and J. Kautz. 2020. UNAS: Differentiable Architecture Search Meets Reinforcement Learning. In Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR).

[WE18] L. Wei, et al. "I know what you see: Power side-channel attack on convolutional neural network accelerators." Proceedings of the 34th Annual Computer Security Applications Conference. 2018.

[WU19] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10734-10742).

[XI20] Y. Xiang, Z. Chen, Z. Chen, Z. Fang, H. Hao, J. Chen, Y. Liu, Z. Wu, Q. Xuan, and X. Yang, "Open dnn box by power side-channel attack," IEEE Transactions on Circuits and Systems II: Express Briefs, 2020

[YA18] M. Yan, C. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn dnn architectures," arXiv preprint arXiv:1808.04761, 2018.

[YU21] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu. 2021. Compute-in-memory chips for deep learning: Recent trends and prospects. IEEE circuits and systems magazine, 21(3), pp.31-56.

[ZHA20] L. Zhang, Y. Yang, Y. Jiang, W. Zhu, and Y. Liu. 2020. Fast hardware-aware neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 692-693).

[ZOP16] B. Zoph and Q. V. Le. 2016. Neural Architecture Search with Reinforcement Learning. arXiv preprint arXiv:abs/1611.01578.