

Multi-Partner Project: A Holistic and Open-Source Approach to Efficient, Secure and Reliable AI Hardware Deployment in DI-EDAI

Georgios Sotiropoulos*, Felix Frombach*, Julian Hoefler*, Tanja Harbaum*, Juergen Becker*, Henrik Iver Thorøe*, Vincent Meyers*, Mehdi Tahoori*, Zeynep Demirdag*, Mohammed Bakr Sikal*, Hassan Nassar*, Heba Khdr*, Jörg Henkel*, Christopher Wolters†, Philipp van Kempen†, Johannes Geier†, Ulf Schlichtmann†, Batuhan Sesli‡, Muhammad Sabih‡, Jakob Wittmann‡, Frank Hannig‡, Jürgen Teich‡, Lukas Steiner§, Norbert Wehn§, Mohamed Shelkamy Ali§, Philipp Schmitz§, Wolfgang Kunz§, Stefan Koegler†, Georg Sigl†¶

*Karlsruhe Institute of Technology (KIT), †Technical University of Munich (TUM),

‡Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), §RPTU University Kaiserslautern-Landau,

¶Fraunhofer Institute for Applied and Integrated Security (AISEC), Germany

Abstract—Artificial Intelligence (AI) has demonstrated strong capabilities across various domains over the past decade. Edge and specifically mission-critical applications, such as automotive and aerospace, require both high performance and efficiency without compromises in security and reliability. This stems from tightly constrained power consumption, failures that can have catastrophic consequences and devices that may be physically accessible to malicious actors. AI algorithm deployment to hardware also presents significant barriers, requiring specialized knowledge and expensive development tools. The DI-EDAI project aims to offer a holistic approach for connecting high-level AI algorithms with hardware implementations while tackling the aforementioned issues. Unlike other approaches that address individual aspects of the AI deployment flow, we investigate solutions across multiple layers of the design stack. Through our work we develop efficient hardware, map AI algorithms to hardware while simultaneously ensuring security and reliability. Furthermore, we leverage AI-techniques to assist with Electronic Design Automation (EDA) workflows for design optimization, verification and implementation. Our open source approach aims to reduce entry barriers, promote transparency and education, and spark innovation. This paper presents the current state of the DI-EDAI project at mid-term, highlighting our latest contributions, identifying limitations in existing state-of-the-art approaches, and outlining ongoing work to address these gaps.

Index Terms—AI Hardware Deployment, Hardware Software Co-Design, AI-assisted EDA, Security and reliability

I. INTRODUCTION

The past decade has seen a rapid development and deployment of Artificial Intelligence (AI) algorithms across diverse fields. Breakthrough advancements and widespread adoption demonstrate its transformative potential in both consumer and industrial applications. As the field advances through larger datasets and more complex models, it is apparent that algorithmic breakthroughs alone cannot address the challenges of

This work is funded by the German Federal Ministry of Research, Technology and Space (BMFTR) in the framework of design tools for sovereign chip development with open source (DE:Sign DI-EDAI, grant numbers 16ME0990K, 16ME0991, 16ME0992, and 16ME0993).

real-world AI deployment. The bottlenecks of general-purpose processors highlight the need for custom hardware accelerators to meet the efficiency, security and reliability demands [1]. This is especially true for edge devices where we face strict resource constraints, potential physical access by malicious actors and usage in mission-critical applications [2].

Furthermore, the increasing design and verification complexity of modern hardware systems, along the current proprietary and expensive AI deployment pipelines, create barriers that limit widespread accessibility and innovation. This motivates the exploration of alternative approaches in Electronic Design Automation (EDA) that are driven by AI to automate and optimize the hardware design process [3]. Moreover, it highlights the need for building and actively maintaining an open-source ecosystem.

The DI-EDAI project aims to address these issues and provide a competitive advantage for research institutes, as well as Small and Medium-sized Enterprises (SMEs), located in Germany and Europe in general. To achieve this, an open-source and holistic approach is adopted for deploying AI algorithms on hardware that meets the demands of modern applications. The primary objectives are to eliminate entry barriers, educate the next generation of engineers and researchers and build a community that uses and supports open-source AI hardware design.

The project's technical approach focuses on bridging the gap between software model development and hardware deployment while minimizing the need for high expertise and expensive EDA tools. Consequently, three focus areas of the project can be identified.

- Design of multiple hardware accelerators covering various use case requirements and AI workloads.
- Efficient mapping of AI algorithms to the selected hardware.
- Security and reliability assurance through hardware-based

methods and early-stage design verification.

In this paper, we present limitations in the state-of-the-art, our work in progress and the contributions we have made so far to the DI-EDAI project. First, we present hardware architecture designs that handle the demanding AI computations in Section II. In Section III, we cover the ways of performing optimizations and mapping AI algorithms to custom hardware. Section IV presents alternative methods to performing EDA tasks by utilizing AI. Finally, we address security concerns in AI hardware accelerators through developing early-stage design verification methods in Section V. Section VI concludes the paper.

II. HARDWARE ARCHITECTURE DEVELOPMENT FOR EFFICIENT AI COMPUTATIONS

A. End-to-End Framework for Spiking Neural Network Deployment on FPGAs

Spiking Neural Networks (SNNs) have emerged as an attractive alternative to standard artificial Neural Networks (NNs) in the context of edge devices, which must operate under strict resource and power constraints [4]. Their event-based processing models neuronal behavior more closely to biology, while achieving low power consumption due to sparse activations. SNNs are particularly well-suited for continuous sensor data and other streaming inputs, where their sparse, event-driven nature enables efficient real-time processing. Recent research has focused on mapping SNNs onto Field Programmable Gate Arrays (FPGAs), which are especially well-matched to such streaming workloads. Existing approaches like *S2N2* [5] and *Spiker+* [6] remain limited in supported quantization schemes, neuron models, hardware-aware training and lack of support for Convolutional Neural Networks (CNNs). Especially in image recognition tasks, CNNs exceed the capabilities of strictly fully-connected NNs in terms of prediction accuracy and number of parameters.

In this work, we address this limitation by presenting a flexible framework for the deployment of quantized SNNs on FPGAs. Our approach provides support for quantized SNNs, enabling a streamlined path from training to deployment. By designing a wrapper that extracts the required parameters for the hardware (e.g. membrane potential bit-width), the training library can be freely chosen by the user. Further, we integrate quantization-awareness into the training, to avoid an accuracy mismatch between software model and the resulting hardware. The model is then exported as an intermediate graph-based representation in the widely adapted Quantized Open Neural Network Exchange (QONNX) [7] format. By doing so, we create a bridge to FINN, which accepts QONNX models for generating NN accelerators. We extend FINN by support for spiking neurons in a similar fashion to [5], by adding new High-Level Synthesis (HLS) components. Furthermore, our framework allows easy integration of custom neuron models by modularly replacing HLS components. Additionally, our framework supports fully-connected as well as convolutional layers, which allows a wider range of architectures. A full overview of the framework is presented in Figure 1.

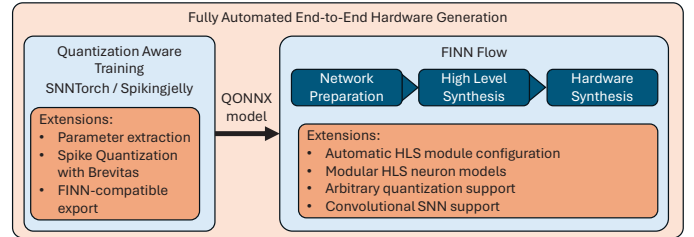


Fig. 1. Overview of the proposed framework for SNN FPGA accelerator generation. Extension are highlighted in orange boxes.

TABLE I
RESOURCE UTILIZATION FOR SPIKING NEURAL NETWORKS GENERATED WITH OUR PROPOSED FRAMEWORK

Network	#Neurons	LUTs	Registers	BRAM
FC	196/10	27,666 (53%)	17,343 (16%)	101.5 (73%)
CNN	6912/1024/10	11,354 (21%)	11,306 (11%)	19 (14%)

Utilizing our framework, we train two models on the MNIST dataset, a two layer fully-connected NN (FC) and a three-layer CNN and deploy them on a XC7Z020 FPGA board. The architecture can be seen in Table I. The table provides an overview of the number of neurons per layer, which are separated by "/" and the FPGA resource utilization in terms of Lookup Tables (LUTs), registers and Block Random Access Memorys (BRAMs). Both models achieve 95% accuracy on the test dataset. For *Spiker+* [6], the number of neurons is capped at 1220 for the same device, we are using. As our framework makes use of *FINN*'s hardware reuse through time-multiplexing, we can deploy substantially larger models as shown for the CNN in Table I.

In summary, the proposed framework provides a practical foundation for accelerating research on FPGA-based SNNs. By unifying training, quantization, and deployment, it simplifies the exploration of design trade-offs and supports reproducible hardware–software co-design studies. Beyond performance evaluation, it also opens opportunities for investigating the robustness and security of SNN hardware implementations under realistic deployment constraints.

B. Application-Specific DRAM Controller

Computational acceleration alone is not sufficient since many of today's AI workloads are memory bound due to their low operational intensity and the small increase in memory bandwidth compared to computing power over the years [8]. This means that execution speed is primarily determined by the performance of the memory subsystem. For Dynamic Random Access Memory (DRAM), which is the memory technology used to realize large capacities, the bandwidth and latency are not constant, but depend on the memory access pattern of the executed workload. This is due to the hierarchical internal architecture consisting of banks, rows and columns that is necessary to achieve high storage density. Columns can only be read or written if the corresponding row is active, and in each bank, only one row can be active at a time because of shared circuitry. Switching the active row requires additional commands that increase the latency of accesses and decrease the bandwidth utilization. Furthermore, since DRAM uses a

bidirectional data bus, switching between read and write mode also comes with penalties.

The DRAM controller is a complex piece of hardware that translates incoming memory read and write requests from the rest of the system into DRAM commands. It consists of numerous state machines and counters, which are necessary for observing the correct order and timing of commands. In addition, it can be equipped with a scheduler that reorders requests on-the-fly to minimize the number of row switches and read/write switches. However, this scheduler increases the required hardware resources once again, adds an additional fixed latency and can only operate on a limited window of requests. As the memory access patterns of AI accelerators are often fixed and known in advance, optimizations can also be carried out a priori. One example is to determine an application-specific mapping from system physical addresses to DRAM addresses that minimizes the number of row switches [9]. The implementation complexity of such a mapping is much lower than that of a scheduler, but the state machines and counters of the controller are still required.

In our new optimization approach, we therefore go one step further and determine the entire DRAM command sequence in advance, similar to the approach shown in [10]. This sequence is replayed in hardware by an application-specific DRAM controller that consists of a state machine and a Static Random Access Memory (SRAM). Since the capacity of the SRAM is very limited, it can only store short command sequences. To overcome this limitation, a state machine is added, which allows to implement control flow reflecting the regularity and repetitiveness of the AI accelerator’s memory access pattern. Each SRAM entry consists of a DRAM command, the lower part of the DRAM address and the number of delay cycles to the next DRAM command. The address to the SRAM entry and the upper part of the DRAM address are provided as output of the state machine. Consequently, different memory regions can be addressed using the same command sequence.

III. HARDWARE/SOFTWARE CO-DESIGN FOR AI APPLICATIONS

A. Co-Design of RISC-V Custom Neural Network Instruction Extensions

Traditionally, Instruction Set Architectures (ISAs) were closed-source and highly proprietary. The advent of RISC-V, an open-source ISA, has democratized processor customization and development, fostering innovation and generating significant interest in both industry and academia [11]. This shift has particularly spurred advancements in processor extensions, such as Custom Functional Units (CFUs).

In DI-EDAI, we explore software optimization techniques for deploying NNs, paired with the hardware design of specialized CFUs that leverage these optimizations to boost computational efficiency. These CFUs are integrated seamlessly into the processor pipeline, improving efficiency through specialized kernels. Our research in co-designing hardware and software for RISC-V CFUs can be organized into three main areas, namely custom computation engine investigation, data locality

improvement and bandwidth reduction, and software approximation for activation functions.

We have investigated several lightweight RISC-V extensions to accelerate NNs. One notable technique is *weight clustering*, which is widely used for compressing CNNs. Our CFU-based approach [12] not only uses weight clustering for compression but also facilitates CNN acceleration by 4.9–6.7 \times with minimal area overhead. By incorporating target-specific optimizations and data layout transformations into a semi-automated compilation flow, additional speed-up potential could be showcased [13]. In [14], we even proposed compressing Binarized Neural Networks (BNNs) using a fixed-length compression scheme that can be efficiently decoded using a CFU, comprising an integrated codebook storage and parallel operation for XOR and population-count. These *sub-bit neural networks* [14] achieve a 2 \times speedup over conventional BNNs with minimal accuracy loss. While NN sparsification alone does not inherently yield acceleration—since hardware may still fetch and process zero-weight values—we directly address this inefficiency by introducing two sparsity-aware CFUs in [15]. These units eliminate unnecessary weight-load operations and multiplications, thereby significantly reducing execution time. For *semi-structured sparsity*, our approach sacrifices a few bits in a block of CNN weights to encode the information about sparsity in the succeeding blocks, i.e. how many weights can be skipped by the loop counter. For *unstructured sparsity*, our proposed CFU dynamically identifies zero weights and avoids multiplications. The proposed CFU sparsity accelerators achieve a speed-up of up to 5 \times . In recent work [16], we have targeted accelerating activation functions, particularly within LSTM networks, by storing approximated function tables in FPGA LUTs, employing interval splitting and exploring different precision levels for each function table, resulting in a 1.7 \times end-to-end network speedup with minimal LUT overhead.

Decoupling hardware and software development processes often leads to inefficient design decisions and missed optimization opportunities. The CFUs proposed above, along with optimized software kernels, enhance open-source processor pipelines, resulting in increased efficiency through offloading. To promote an ecosystem of diverse CFUs as essential building blocks for AI processor extensions, we plan to release our CFUs as open-source. This will further advance research on co-design optimizations and design-space exploration.

B. ML Deployment on the Edge

To enable effective co-design and maximize the potential of custom hardware architectures, such as CFUs, compiler frameworks like TVM [17] and IREE [18] are essential. These frameworks address both the hard and soft requirements of retargeting AI. Hard criteria, such as inference latency, throughput, and memory footprint, govern their performance on heterogeneous and embedded targets. Soft criteria, such as open-source continuity and community, shape their long-term viability. Our preliminary comparison of the two shows that TVM excels in hard metrics through aggressive autotuning, while IREE also emphasizes modularity, ecosystem integration,

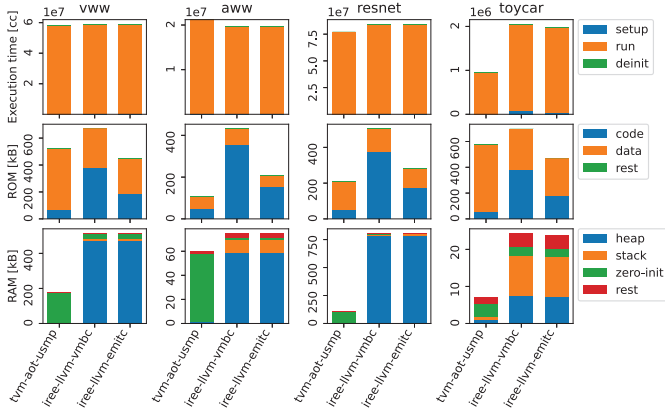


Fig. 2. Comparison of TVM and IREE deployment of extreme edge TinyML inference benchmarks from MLPerfTMTiny with the MLonMCU framework.

and sustainable open-source development, building on proven communities grounded in MLIR [19] and LLVM [20]. Both illustrate complementary strategies in balancing performance efficiency with community-driven portability.

Figure 2 compares three open-source AI deployment strategies for the extreme edge case of CPU-only inference, *tvm-aot-usmp*, *iree-llvm-vmcb*, and *iree-llvm-emitc*, across different Deep Neural Network (DNN) and CNN workloads (from MLPerfTMTiny [21]). Measurements were performed with the MLonMCU [22] TinyML benchmark framework. In *tvm-aot-usmp*, the model is lowered with MicroTVM with Ahead-of-Time (AoT) execution and Unified Static Memory Planning (USMP) features activated. With option *iree-llvm-**, IREE utilizes the LLVM CPU Hardware Abstraction Layer (HAL) backend. With *iree-llvm-vmcb*, the model is lowered to VM bytecode interpreted by the runtime, while configuration *iree-llvm-emitc* utilizes IREE’s EmitC backend for static C-code, similar to [23]. With the exception of the *toy-car* model, total inference time is similar. Regarding the Read-only Memory (ROM) footprint, EmitC consistently outperforms the VM bytecode IREE output format due to its reduced runtime and performs similarly to the TVM configuration. The most critical difference between TVM and IREE becomes apparent when comparing the Random Access Memory (RAM) demands: MicroTVM and USMP significantly reduce the system’s RAM footprint, compared to the two IREE configurations, which both allocate dynamically on the heap. This highlights the importance of the AI retargeting scheme when exploring the design and tool space of HW/SW co-design for AI applications.

IV. AI ASSISTED ELECTRONIC DESIGN AUTOMATION

A. AI-Driven Hardware Reliability Analysis and Enhancement Methods

The various application conditions of electronics, such as extreme temperatures, voltage fluctuations or even radiation can affect electronics in the form of Single Event Effects (SEEs), which appear as temporal bit-flips. These faults are undesirable in critical applications such as automotive and aerospace. When assessing the robustness of AI models like DNNs, it can be

shown that even a single bit-flip can lead to incorrect predictions [24]. Consequently, research focuses on analysis of fault effects on circuits and development of hardware techniques to identify errors and ensure correct functionality of components even in the presence of such faults.

Triple Modular Redundancy (TMR) provides a robust solution to the aforementioned issue by incorporating three instances of the same circuit in parallel and determining the output through majority voting. However, significant area and power overhead make their usage impractical, especially for edge applications where efficiency is of great importance [25]. For DNNs, we minimized these overheads by developing approximate Bayesian Neural Networks (BayNNs) inference to provide confidence metrics for the model’s output and providing a safety mechanism working against both random hardware faults and functional insufficiencies [26]. Moreover, we implemented a custom hardware module for approximating these computations on FlexXNNGine, a systolic array, open-source accelerator [27], [28].

Comprehensive reliability analysis can provide designers with meaningful insights about the fault tolerance level of their circuit, help identify critical modules and enable implementations with different diagnostic coverage levels. However, these techniques have a trade-off between accuracy and analysis time [29]. Graph Neural Network (GNN) approaches have demonstrated promising results when handling EDA tasks, since circuits can be naturally represented with graphs at different levels of abstraction. Specifically for reliability analysis, research shows particularly high accuracy, decreased simulation time, generalization capability to larger circuits and ability to identify critical components. Despite that, many aspects remain unexplored regarding SEEs, such as ways to optimize the efficiency-accuracy trade-off, address Single Event Transient (SET) faults, adapt for varying error rates and account for individual gate characteristics [3].

While [30] demonstrates strong performance, its test dataset comprises circuits that share identical functionality but vary in implementation. This methodology limits the ability to generalize beyond the specific circuit types examined and is primarily suited for Design Space Exploration (DSE) applications. Similarly, reference [31] uses its test dataset during the fine-tuning phase. Although this approach substantially reduces the number of fault injections required to obtain ground truth labels, it still necessitates 50 epochs of fine-tuning.

Our work focuses on developing a GNN model tailored specifically for reliability analysis. Our approach involves extracting small sub-circuits to generate a diverse ground truth dataset through fault injection with varying error rates. The error rates are then incorporated in the graph structure as node features, thus eliminating the need for retraining when analyzing circuit behavior for different values.

B. Open Source Dataset for LLM-Aided Design

Aside from GNNs, another interesting candidate for AI-based design automation are Large Language Models (LLMs), offering new possibilities for reasoning about hardware description languages and optimizing EDA workflows. However, the

true potential of *LLM-aided design* remains largely unexplored due to the absence of well-structured and tokenized hardware datasets suitable for model training. Most available Verilog datasets were created for conventional machine learning tasks or contain plain non-tokenized code that cannot be directly used in modern transformer-based architectures. Consequently, researchers and tool developers face significant challenges in adapting LLMs for EDA applications, such as design synthesis, anomaly detection, and hardware security.

To address this gap, we introduce an open-source dataset specifically tailored and tokenized for LLM training on Verilog hardware designs [32]. The dataset serves as a base for exploring LLM-assisted EDA tasks, and we demonstrated its effectiveness in one concrete use case: the detection of *power-wasters* [33] in routed FPGA netlists. Power wasters represent a critical design and security challenge, particularly in multi-tenant FPGA environments, as they create high-frequency switching activity that stresses the shared Power Distribution Network (PDN) [34] and may induce timing faults [35] or denial-of-service conditions [36].

Our dataset is generated using routed netlists from the Xilinx ZCU102 MPSoC FPGA, covering realistic multi-tenant floorplans with 3, 6, and 12 tenants. It includes 480 fully routed FPGA designs and 17,519 extracted and tokenized Verilog submodules, balanced between benign and malicious examples. Tokenization is performed at the syntax level using a lightweight regular expression parser that isolates Verilog module boundaries and preserves semantic structure, ensuring direct compatibility with text-based LLM training. This eliminates the need for vendor-specific preprocessing or feature engineering.

We demonstrate the usability of this dataset by fine-tuning three open-source LLMs, Llama 3.2 (3B), DeepSeek-R1 (7B), and Mistral (7B), using efficient adaptation of the Low Rank Adaptation (LoRA) parameters within the Unsloth framework. The models were trained on NVIDIA T4 GPUs that required less than 15 minutes and less than 14 GB of memory. Among all variants, DeepSeek-R1-80L achieved the highest classification performance, reaching 98.5% accuracy and an F1 score of 0.985 when trained to detect power-wasting circuits. This shows that our dataset provides sufficient structure and diversity to enable effective fine-tuning of LLMs for EDA-related design analysis.

We released this dataset and the fine-tuned models open source on Hugging Face, aiming to establish a reproducible benchmark for *LLM-aided design exploration*. This will serve as a useful base for the next generation of AI-based EDA tools.

This work shows how leveraging AI can accelerate hardware verification, despite the high complexity, by enabling faster reliability analysis with GNNs and automating the detection of power wasting circuits with LLMs.

V. SECURITY FOR AI HARDWARE

Security is a major concern for AI accelerators as they are of high economic interest. In the CIA triad, security comprises Confidentiality, Integrity and Availability [37].

Confidentiality in AI accelerators is multi-faceted. User data and the model itself need to be protected from malicious actors. The latter represents valuable Intellectual Property (IP) due

to the substantial resources put in its design and training. Therefore, model theft can cause significant financial and intellectual losses.

One prominent class of attacks is using microarchitectural timing, power or Electro-Magnetic (EM) Side-Channel Analysis (SCA) to reverse engineer the model [38], [39]. Successful attacks, based on these methods, have been demonstrated against software [40] and hardware [41], [42] implementations, highlighting the need for countermeasures. Using formal verification, we can detect vulnerabilities to model theft at the microarchitectural level [43], through a fully automated and scalable to real designs methodology. The case study covers the FINN data-flow architecture as well as a systolic array design, in which the proof identified a microarchitectural optimization for power efficiency within the systolic array design to cause a timing side channel.

The second aspect of the CIA triad we are covering is availability. For training and inference hardware developers often employ their AI models on cloud services as the process is resource intensive. This introduces a new threat of remote fault injection and denial-of-service attacks. As mentioned in Sec. IV-B, these attacks originate in high switching activity, causing stress on the shared power distribution network. The most influential metadata used to detect such attacks by AI is the power estimation [34].

The state of the art tool for power estimation is Xilinx Vivado power estimator. However, it is a heuristic approach targeting average instead of peak power consumption. Therefore, we apply formal techniques to obtain mathematical guarantees for the worst-case switching activity, which is related to the dynamic power consumption. The results can either serve as a standalone method for detecting DoS attacks or as a metric that can be fed into other offline tools.

We use property checking to detect the maximum possible switching activity of the Design Under Verification (DUV) based on the Hamming distance between valuations to the DUV's signals over time. This metric helps to determine whether the power consumption of the DUV can exceed a vendor-specific threshold.

One countermeasure that has been extensively researched for the application to cryptographic circuits is masking. Masking is an information-theoretically secure countermeasure that prevents SCA by ensuring that a device's power consumption or EM emanation is statistically independent of secret data being processed. Since masking provides an information-theoretically provable security guarantee, a masked circuit's side-channel resistance can be verified early in the design process based on its hardware description language definition. We evaluate the accuracy and performance of existing open-source formal verification tools for the application to NN hardware accelerators:

SILVER [44] analyzes the probability distribution of probed wires to verify the statistical independence of the joint distribution of probes from the joint distribution of sensitive inputs. The physical circuit is modeled as a Reduced Ordered Binary Decision Diagram (ROBDD) derived from the gate-level

TABLE II
BENCHMARK OF SILVER AND PROVER FOR MASKED MULTIPLY GADGETS WITH DIFFERENT INPUT AND OUTPUT BIT-WIDTHS.

	Multiplier(2,4)	Multiplier(3,6)	Multiplier(4,8)	Multiplier(5,10)
SILVER				
Probing standard	< 1 second	< 1 second	2.5 seconds	BDD Unique table full
Probing robust	< 1 second	220 seconds	TIMEOUT	—
NI standard	< 1 second	1 second	—	—
NI robust	2.5 seconds	65 minutes	—	—
SNI standard	< 1 second	1 second	—	—
SNI robust	2.5 seconds	64 minutes	—	—
PINI standard	< 1 second	2 seconds	—	—
PINI robust	3 seconds	65 minutes	—	—
Uniformity	< 1 second	< 1 second	—	—
PROVER				
Probing standard	< 1 second	< 1 second	< 1 second	BDD Unique table full
Probing robust	< 1 second	< 1 second	2 seconds	—
Uniformity	< 1 second	< 1 second	251 seconds	—

netlist. SILVER can verify hardware gadgets according to the formal security notions t -Non-Interference (NI), t -Strong Non-Interference (SNI), and t -Probe-Isolating Non-Inference (PINI), as well as uniformity.

PROVER [45] builds upon SILVER and improves the performance by reducing the size of observation and secret sets, as well as reordering ROBDDs. However, the formal verification is limited to standard probing security, glitch-extended probing security, and uniformity.

COCOALMA [46] employs a multi-step process based on grouping linear combinations that a gate correlates to into correlation sets. The correlation sets are formulated as a boolean satisfiability problem, which is solved by an SAT solver.

Because COCOALMA is prone to false positives, e.g. it incorrectly classifies an arithmetically masked domain-oriented multiply gadget as insecure, we find it unsuitable for verification of NN accelerators.

To evaluate the limitations of SILVER and PROVER we run verifications of an arithmetically masked domain-oriented multiply gadget and a masked Sigmoid activation function, with varying bit-widths. The failure points are either a runtime exceeding 24 hours or the size of the Binary Decision Diagram (BDD) representing the gate-level netlist exceeding the size at which it can be processed by the verification tools. All test are run on an Intel i7-10700 with 128 GB of memory. The results are shown in Table II for the Multiply Gadget and Table III for the activation function. Our benchmarks show that SILVER is limited by both runtime and size of the netlist, while PROVER is solely limited by the size of the netlist graph. This work benchmarks open-source formal verification tools for NN hardware accelerators. Physical side-channel measurements were not possible within the current timeline and remain as future work.

VI. CONCLUSION

This paper presents the current progress of the mid-term phase of the DI-EDAI project. We introduce hardware architectures and hardware/software co-design approaches to enable faster and more efficient AI computations. Furthermore, we present methods for exploiting AI for EDA tasks, as well as ensuring hardware security against IP and private data

TABLE III
BENCHMARK OF SILVER AND PROVER FOR MASKED SIGMOID ACTIVATION FUNCTIONS WITH DIFFERENT INPUT AND OUTPUT BIT-WIDTHS.

	Sigmoid(4,8)	Sigmoid(8,16)
SILVER		
Probing standard	< 1 second	BDD Unique table full
Probing robust	71.5 seconds	—
NI standard	< 1 second	—
NI robust	268.5 seconds	—
SNI standard	< 1 second	—
SNI robust	279.5 seconds	—
PINI standard	< 1 second	—
PINI robust	256.5 seconds	—
Uniformity	3.5 seconds	—
PROVER		
Probing standard	< 1 second	BDD Unique table full
Probing robust	1 second	—
Uniformity	48.5 seconds	—

theft. Special attention is given to providing our current and future contributions as open-source resources to accelerate research and innovation. Through these efforts, we have made considerable progress in bridging the gap between AI model development and hardware deployment in an efficient, secure and reliable way.

REFERENCES

- [1] Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang, "A survey of accelerator architectures for deep neural networks," *Engineering*, vol. 6, no. 3, p. 264–274, Mar. 2020.
- [2] X. Wang, Z. Tang, J. Guo, T. Meng, C. Wang, T. Wang, and W. Jia, "Empowering edge intelligence: A comprehensive survey on on-device AI models," *ACM Comput. Surv.*, vol. 57, no. 9, pp. 228:1–228:39, Apr. 2025.
- [3] Z. El Sayed, Z. Wang, H. Selmani, J. Knechtel, O. Sinanoglu, and L. Alrahis, "Graph neural networks for integrated circuit design, reliability, and security: Survey and tool," *ACM Comput. Surv.*, Sep. 2025, just Accepted. [Online]. Available: <https://doi.org/10.1145/3769081>
- [4] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, "Spiking neural networks and their applications: A review," *Brain sciences*, vol. 12, no. 7, p. 863, 2022.
- [5] A. Khodamoradi, K. Denolf, and R. Kastner, "S2N2: A FPGA accelerator for streaming spiking neural networks," in *Proc. of the ACM/SIGDA Int'l Symposium on Field-Programmable Gate Arrays*, 2021, pp. 194–205.
- [6] A. Carpegna, A. Savino, and S. Di Carlo, "Spiker+: a framework for the generation of efficient spiking neural networks fpga accelerators for inference at the edge," *IEEE Trans. on Emerging Topics in Computing*, 2024.
- [7] Y. Umuroglu, H. Borrás, V. Loncar, S. Summers, and J. Duarte, "fastmachinelearning/qonnx," 06 2022. [Online]. Available: <https://github.com/fastmachinelearning/qonnx>
- [8] A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney, and K. Keutzer, "AI and memory wall," *IEEE Micro*, vol. 44, no. 3, pp. 33–39, 2024.
- [9] M. V. Natale, M. Jung, K. Kraft, F. Lauer, J. Feldmann, C. Sudarshan, C. Weis, S. Krumke, and N. Wehn, "Efficient generation of application specific memory controllers," in *Proc. of the Int'l Symposium on Memory Systems (MEMSYS)*. ACM, 2021, p. 233–247. [Online]. Available: <https://doi.org/10.1145/3422575.3422796>
- [10] S. Bayliss and G. A. Constantinides, "Methodology for designing statically scheduled application-specific SDRAM controllers using constrained local search," in *Proc. of the Int'l Conference on Field-Programmable Technology*, 2009, pp. 304–307.
- [11] F. Hannig and J. Teich, "Open source hardware," *IEE Computer*, vol. 54, no. 10, pp. 111–115, 2021.

- [12] M. Sabih, B. Sesli, F. Hannig, and J. Teich, "Accelerating dnns using weight clustering on risc-v custom functional units," in *2024 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2024, pp. 1–2.
- [13] M. Tahoori *et al.*, "Special Session – Hardware-software co-design for machine learning systems made open-source," in *Proc. of the Int'l Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. ACM, 2025 (in press).
- [14] M. Sabih, M. Abdo, F. Hannig, and J. Teich, "Beyond bnns: Design and acceleration of sub-bit neural networks using risc-v custom functional units," *IEEE Embedded Systems Letters*, vol. 17, no. 5, pp. 329–332, 2025.
- [15] M. Sabih, A. Karim, J. Wittmann, F. Hannig, and J. Teich, "Hardware/software co-design of risc-v extensions for accelerating sparse dnns on fpgas," in *2024 International Conference on Field Programmable Technology (ICFPT)*, 2024, pp. 01–09.
- [16] B. Sesli, M. Sabih, F. Hannig, and J. Teich, "Design of machine learning accelerators as RISC-V extensions using an open source tool flow," in *Proc. of the Int'l Conference on Computer Aided Design (ICCAD)*. IEEE, 2025.
- [17] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, M. Cowan, H. Shen, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, "Tvm: An automated end-to-end optimizing compiler for deep learning," in *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'18. USA: USENIX Association, 2018, p. 579–594.
- [18] The IREE Authors, "IREE," Sep. 2019. [Online]. Available: <https://github.com/iree-org/iree>
- [19] C. Lattner, M. Amini, U. Bondhugula, A. Cohen, A. Davis, J. Pienaar, R. Riddle, T. Shpeisman, N. Vasilache, and O. Zinenko, "MLIR: Scaling compiler infrastructure for domain specific computation," in *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, 2021, pp. 2–14.
- [20] C. Lattner and V. Adve, "Llvm: a compilation framework for lifelong program analysis & transformation," in *International Symposium on Code Generation and Optimization, 2004. CGO 2004.*, 2004, pp. 75–86.
- [21] C. Banbury *et al.*, "Mlperf tiny benchmark," 2021. [Online]. Available: <https://arxiv.org/abs/2106.07597>
- [22] P. van Kempen, R. Stahl, D. Mueller-Gritschneider, and U. Schlichtmann, "Mlonmcpu: Tinyml benchmarking with fast retargeting," in *Proceedings of the 2023 Workshop on Compilers, Deployment, and Tooling for Edge AI*, ser. CODAI '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 32–36. [Online]. Available: <https://doi.org/10.1145/3615338.3618128>
- [23] H.-I. C. Liu, M. Brehler, M. Ravishankar, N. Vasilache, B. Vanik, and S. Laurenzo, "Tinyiree: An ml execution environment for embedded systems from compilation to deployment," *IEEE Micro*, vol. 42, no. 5, pp. 9–16, 2022.
- [24] J. Hoefler, F. Kempf, T. Hotfilter, F. Kreß, T. Harbaum, and J. Becker, "Sifi-ai: A fast and flexible rtl simulation framework tailored for ai models and accelerators," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, ser. GLSVLSI '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 287–292. [Online]. Available: <https://doi.org/10.1145/3583781.3590226>
- [25] F. Mireshghallah, M. Bakhshalipour, M. Sadrosadati, and H. Sarbazi-Azad, "Energy-efficient permanent fault tolerance in hard real-time systems," *IEEE Trans. on Computers*, vol. 68, no. 10, p. 1539–1545, Oct. 2019.
- [26] J. Hoefler, M. Stammner, F. Kreß, T. Hotfilter, T. Harbaum, and J. Becker, "BayWatch: Leveraging bayesian neural networks for hardware fault tolerance and monitoring," in *Proc. of the IEEE Int'l Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct. 2024, p. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/10753546>
- [27] J. Hoefler, F. Lesniak, A. Gutermann, F. Wöran, T. Harbaum, and J. Becker, "Baynngine: Hardware-enabled bayesian neural network support for dependable ai inference," in *2025 IEEE 38th International System-on-Chip Conference (SOCC)*, 2025, pp. 1–6.
- [28] F. Lesniak, A. Gutermann, T. Harbaum, and J. Becker, "Enhanced accelerator design for efficient CNN processing with improved row-stationary dataflow," in *Proc. of the Great Lakes Symposium on VLSI (GLSVLSI)*. ACM, Jun. 2024, p. 151–157. [Online]. Available: <https://dl.acm.org/doi/10.1145/3649476.3658737>
- [29] M. H. Ahmadiilivani, M. Taheri, J. Raik, M. Daneshtalab, and M. Jenihhin, "A systematic literature review on hardware reliability assessment methods for deep neural networks," *ACM Comput. Surv.*, vol. 56, no. 6, pp. 141:1–141:39, Jan. 2024.
- [30] A.-A. Koufopoulou, A. Papadimitriou, A. Pikrakis, M. Psarakis, and D. Hely, "On the prediction of hardware security properties of HLS designs using graph neural networks," in *Proc. of the IEEE Int'l Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct. 2023, p. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/10313544>
- [31] S. Khan, Z. Shi, Z. Zheng, M. Li, and Q. Xu, "DeepSeq2: Enhanced sequential circuit learning with disentangled representations," in *Proc. of the 30th Asia and South Pacific Design Automation Conference (ASPDAC)*. ACM, Mar. 2025, p. 498–504. [Online]. Available: <https://dl.acm.org/doi/10.1145/3658617.3697594>
- [32] M. B. Sikal, H. Nassar, H. Khdr, and J. Henkel, "A dataset for LLM-based detection of power-wasters in routed FPGA netlists," in *Proc. of the IEEE Int'l Conference on LLM-Aided Design (ICLAD)*, 2025, pp. 235–241.
- [33] H. Nassar, P. Machauer, D. R. E. Gnad, L. Bauer, M. B. Tahoori, and J. Henkel, "Covert-Hammer: Coordinating power-hammering on multi-tenant FPGAs via covert channels," in *Proc. of the ACM/SIGDA Int'l Symposium on Field Programmable Gate Arrays (FPGA)*. ACM, 2024, p. 43. [Online]. Available: <https://doi.org/10.1145/3626202.3637613>
- [34] H. Nassar, J. Krautter, L. Bauer, D. Gnad, M. Tahoori, and J. Henkel, "Meta-Scanner: Detecting fault attacks via scanning FPGA designs meta-data," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 11, pp. 3443–3454, 2024.
- [35] M. Fathy, H. Nassar, M. Abd El Ghany, and J. Henkel, "Timekeepers: ML-driven SDF analysis for power-wasters detection in FPGAs," *ACM Trans. Embed. Comput. Syst.*, vol. 24, no. 5s, Sep. 2025. [Online]. Available: <https://doi.org/10.1145/3761809>
- [36] H. Nassar, P. Machauer, L. Bauer, D. Gnad, M. Tahoori, and J. Henkel, *DoS-FPGA: Denial of Service on Cloud FPGAs via Coordinated Power Hammering*. ACM, 2025.
- [37] S. Samonas and D. Coss, "The CIA strikes back: Redefining confidentiality, integrity and availability in security," *Journal of Information System Security*, vol. 10, no. 3, 2014.
- [38] G. Dong, P. Wang, P. Chen, R. Gu, and H. Hu, "Floating-point multiplication timing attack on deep neural network," in *Proc. of the IEEE Int'l Conference on Smart Internet of Things (SmartIoT)*, 2019, pp. 155–161.
- [39] C. Gongye, Y. Fei, and T. Wahl, "Reverse-engineering deep neural networks using floating-point timing side-channels," in *Proc. of the 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.
- [40] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *Proc. of the 28th USENIX Security Symposium (USENIX Security)*. USENIX Association, 2019, pp. 515–532. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity19/presentation/batina>
- [41] A. Dubey, R. Cammarota, and A. Aysu, "MaskedNet: The first hardware inference engine aiming power side-channel protection," in *Proc. of the IEEE Int'l Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2020, pp. 197–208.
- [42] K. Yoshida, T. Kubota, S. Okura, M. Shiozaki, and T. Fujino, "Model reverse-engineering attack using correlation power analysis against systolic array based neural network accelerator," in *Proc. of the IEEE Int'l Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5.
- [43] M. S. Ali, L. Deutschmann, J. Müller, A. L. D. Antón, M. R. Fadiheh, D. Stoffel, and W. Kunz, "Security risks in AI accelerators: Detecting RTL vulnerabilities to model theft with formal verification," in *Proc. of the IEEE European Test Symposium (ETS)*. IEEE, 2025, pp. 1–6.
- [44] D. Knichel, P. Sasdrich, and A. Moradi, "SILVER – Statistical independence and leakage verification," in *Proc. of the 26th Int'l Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*. Springer, 2020, pp. 787–816.
- [45] F. Zhou, H. Chen, and L. Fan, "Prover – Toward more efficient formal verification of masking in probing model," *Cryptology ePrint Archive*, Paper 2024/1202, 2024. [Online]. Available: <https://eprint.iacr.org/2024/1202>
- [46] V. Hadžić and R. Bloem, "COCOALMA: A versatile masking verifier," in *Proc. of the Conference on Formal Methods in Computer Aided Design (FMCAD)*, 2021, pp. 14–23.