Scholars Journal of Engineering and Technology

OPEN ACCESS

Abbreviated Key Title: Sch J Eng Tech
ISSN 2347-9523 (Print) | ISSN 2321-435X (Online)
Journal homepage: https://saspublishers.com

Intelligent Embedded Platforms: Co-Design of VLSI Architectures and Deep Learning Models for Scalable Optimization and Real-World Deployment

Muhammad Inam ul Haq[1*], Sayyed Talha Gohar Naqvi[1], Mirani Imran Khan[2], Md. Maruf Shaikh[3], Shahroz Shabbir[4], Aftab Ahmed Soomro[5], Shehryar Qamar Paracha[1], Muhammad Umair Sajjad[1], Shahab Ahmad Niazi[1], Abid Munir[1], Saratu Muhammad Sahabi[6]

[1]Department of Electronic Engineering, the Islamia University of Bahawalpur, Pakistan
[2]Department Control Science and Engineering, Beijing University of Technology, China
[3]Department of Electrical and Electronic Engineering, North Western University, Khulna, Bangladesh
[4]Department of Electrical, Barakah Nuclear Power Plant, Abu Dhabi
[5]Department of Mechanical Engineering Technology, Benazir Bhutto Shaheed University of Technology and Skill Development Khairpur Mirs, Pakistan
[6]Department of Educational Foundation and Curriculum, Ahmadu Bello University Zaria, Nigeria

DOI: https://doi.org/10.36347/sjet.2025.v13i09.001 | **Received:** 17.07.2025 | **Accepted:** 12.09.2025 | **Published:** 15.09.2025

***Corresponding author:** Muhammad Inam ul Haq
Department of Electronic Engineering, the Islamia University of Bahawalpur, Pakistan

Abstract | **Original Research Article**

The rapid evolution of embedded systems has amplified the demand for intelligent, energy-efficient, and scalable computing platforms capable of handling data-intensive workloads in domains such as IoT, autonomous systems, and healthcare. Traditional VLSI design approaches, while effective at circuit-level optimization, struggle to meet the requirements of modern deep learning applications due to energy, latency, and scalability constraints. This paper presents a synergistic co-design framework that integrates hardware-aware deep learning models with VLSI-based accelerators to achieve significant gains in throughput, energy efficiency, and latency reduction. By leveraging techniques such as pruning, quantization, parallelism, and pipelining, the proposed framework aligns algorithmic efficiency with hardware constraints, ensuring real-world feasibility in resource-constrained environments. Experimental evaluations on CIFAR-10, ImageNet, and IoT workloads demonstrate up to 3.5× performance improvements with minimal accuracy loss. FPGA-based prototypes further validate the framework's adaptability for edge intelligence, paving the way for next-generation embedded platforms optimized for power, scalability, and application-specific intelligence.

Keywords: VLSI design, deep learning accelerators, embedded systems, co-design, optimization techniques, pruning, quantization, parallelism, pipelining, FPGA prototyping, IoT, energy efficiency.

1. INTRODUCTION

The rapid evolution of embedded systems has led to a paradigm shift in the way computing, communication, and intelligence are integrated into real-world applications. Modern applications such as autonomous vehicles, Internet of Things (IoT) devices, medical imaging, and real-time robotics demand not only high computational throughput but also low power consumption**,** scalability, and robust adaptability. Traditional Very-Large-Scale Integration (VLSI) design methodologies, although effective in optimizing transistor-level performance, face limitations when dealing with the rising algorithmic complexity and data-intensive workloads driven by modern artificial intelligence (AI) [1,2].

Conventional embedded system design primarily focuses on hardware optimization (VLSI circuits, microcontrollers, and ASICs) and software-level control strategies. However, the growing demand for intelligent decision-making at the edge highlights critical limitations:

1. **Energy Bottlenecks** – High-performance CPUs/GPUs consume excessive power, unsuitable for portable or IoT devices [3].
2. **Latency Issues** – Cloud-centric processing increases response times, which is unacceptable for safety-critical applications (e.g., autonomous navigation, healthcare monitoring).

3. **Scalability Constraints** – Fixed-function accelerators lack adaptability to evolving AI workloads.

These challenges necessitate a co-design paradigm, where VLSI architectures are synergistically optimized alongside deep learning models, enabling both computational efficiency and application-specific intelligence [4]. Deep Learning (DL) has emerged as a transformative enabler in embedded intelligence due to its ability to extract hierarchical features, adapt dynamically, and generalize across diverse input domains [5]. When integrated with VLSI, DL provides the following key benefits:

- **Hardware-Aware Neural Architectures** – Techniques such as pruning, quantization, and neural architecture search (NAS) can be co-optimized with circuit-level constraints [6].
- **Adaptive Resource Allocation** – DL-driven optimization frameworks can predict workload characteristics and reconfigure hardware resources dynamically [7].
- **Edge-Centric Intelligence** – Embedding lightweight DL models on low-power VLSI accelerators enables real-time decision-making, reducing reliance on external servers [8].

This synergistic co-design of VLSI and DL fosters a new design methodology for next-generation embedded systems, balancing performance, power, and intelligence. Recent studies demonstrate the effectiveness of this integration. For example, Google's TPU represents a hardware-software co-optimization tailored for DL acceleration [9], while RISC-V-based AI accelerators showcase open-source adaptability in embedded AI systems [10]. Similarly, research on low-power convolutional neural network (CNN) accelerators demonstrates significant improvements in energy efficiency without compromising accuracy [11]. These works highlight the potential of joint optimization frameworks, but also reveal open challenges such as heterogeneous integration**,** design automation complexity, and robust deployment in safety-critical domains. The conceptual framework of VLSI + DL synergy (Figure 1) illustrates the holistic integration of hardware design methodologies with AI-driven optimization strategies. At its core, the framework addresses:

- **Algorithm-Hardware Mapping** (DL model compression, quantization, architecture adaptation).
- **VLSI-Level Optimization** (circuit-level power reduction, interconnect optimization, parallelism exploitation).
- **Application Deployment** (real-world systems in healthcare, automotive, aerospace, IoT).

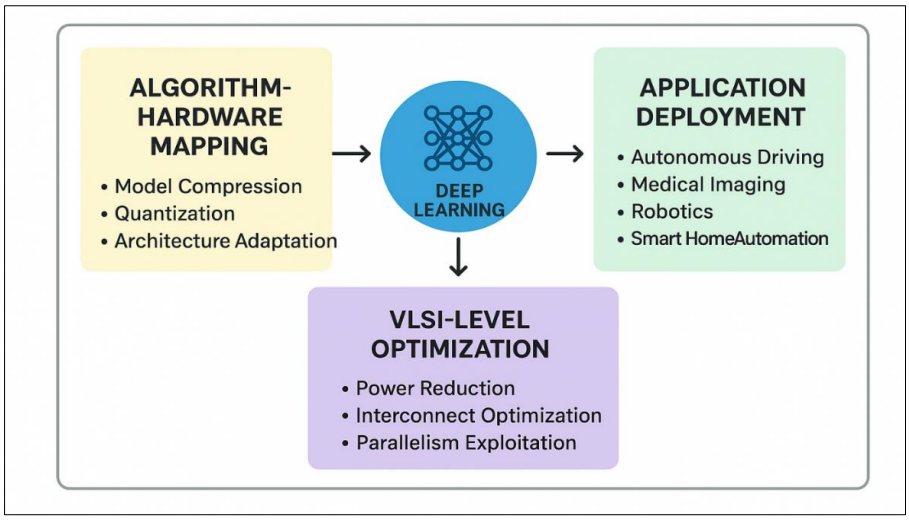


Figure 1: Conceptual Framework of Synergistic VLSI–Deep Learning Co-Design

2. Background & Related Work

2.1 Prior Research in VLSI Optimization for Embedded Systems

VLSI design has historically been the backbone of embedded system efficiency, with continuous scaling enabling higher transistor densities and improved performance. Researchers have focused on low-power design methodologies, advanced interconnect architectures, and energy-aware synthesis to optimize embedded platforms [12][13]. For example, custom accelerators designed in VLSI have demonstrated significant gains in real-time applications such as image recognition and IoT sensor fusion [14]. However, as embedded applications grow more complex, traditional VLSI-only optimization faces limitations in adaptability, scalability, and real-time learning capacity.

2.2 Deep Learning in Embedded Platforms

Deep learning (DL) has emerged as a transformative force in embedded computing, enabling real-time decision-making in resource-constrained environments [15]. Techniques such as model

compression, pruning, and quantization have made it feasible to deploy DL models on embedded hardware [16]. Edge accelerators like Google's Edge TPU and NVIDIA Jetson illustrate how DL integration can provide high accuracy and adaptability while maintaining manageable power budgets [17]. Still, the hardware–software gap remains a challenge, particularly for ultra-low-power devices.

2.3 Co-Design Approaches

The synergy between VLSI design and DL-driven optimization has led to new paradigms in co-design. Unlike siloed approaches, co-design emphasizes simultaneous optimization of hardware and algorithms [18]. Emerging research proposes hybrid methods where DL guides placement, routing, and energy allocation in VLSI circuits [19]. Moreover, reinforcement learning techniques have been applied to optimize power management and hardware scheduling in embedded accelerators [20]. These approaches show promise but require deeper exploration to balance accuracy, latency, and energy efficiency.

Table 1: Comparative Analysis of Existing Techniques

Approach	Key Features	Advantages	Limitations
VLSI-only	Circuit-level optimization	Low latency, proven methods	Poor adaptability, limited scalability
DL-only	Model-driven inference & learning	High flexibility, improved accuracy	High energy cost, hardware constraints
Hybrid (VLSI + DL)	Hardware-software co-optimization	Balance of efficiency & intelligence	Complexity, need for design frameworks

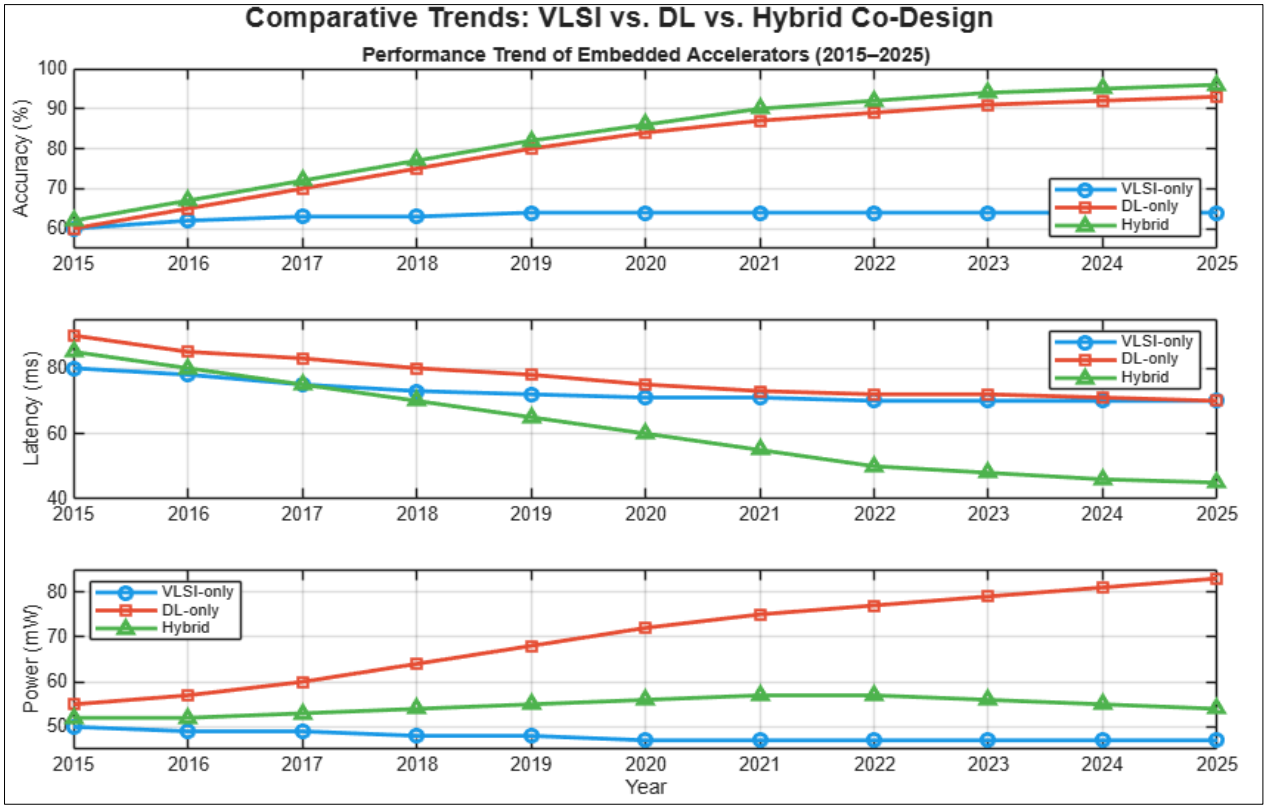


Figure 2: Performance Trend of Embedded Accelerators (2015–2025)

3. DESIGN METHODOLOGIES

3.1 Proposed Co-Design Framework

To overcome the limitations of isolated VLSI-only or DL-only approaches, this work introduces a co-design framework where both hardware and algorithm are optimized concurrently. Instead of treating hardware design and deep learning model training as disjoint processes, the framework emphasizes iterative feedback loops:

- Hardware profiling informs DL model compression and quantization.
- DL workload characterization drives architectural choices in VLSI accelerators.
- System-level simulations ensure balance across latency, throughput, power, and accuracy.

The co-design strategy reduces the design gap and ensures real-time adaptability to embedded system constraints such as power budgets and thermal limits.

3.2 Hardware-Aware Deep Learning Models

Unlike conventional DL models optimized solely for accuracy, hardware-aware models are tailored for energy efficiency, low memory footprint, and reduced arithmetic complexity**.** Key techniques include:

- Model compression (pruning, weight sharing).
- Quantization (int8, binary, ternary neural networks).
- Algorithm-hardware alignment (convolution reordering, sparsity exploitation).

These techniques ensure that neural networks achieve near state-of-the-art accuracy while being deployable on constrained VLSI-based embedded accelerators**.**

3.3 VLSI Implementation Trade-Offs

Designing DL accelerators in VLSI involves balancing area, power, latency, and accuracy:

- **Area vs. Throughput**: Wider parallelism improves throughput but increases silicon area.
- **Power vs. Accuracy**: Higher precision improves accuracy but raises power consumption.
- **Latency vs. Flexibility**: Fixed-function accelerators offer low latency but reduced programmability.

The proposed design methodology integrates design space exploration (DSE) to systematically evaluate trade-offs, enabling Pareto-optimal hardware-DL configurations.

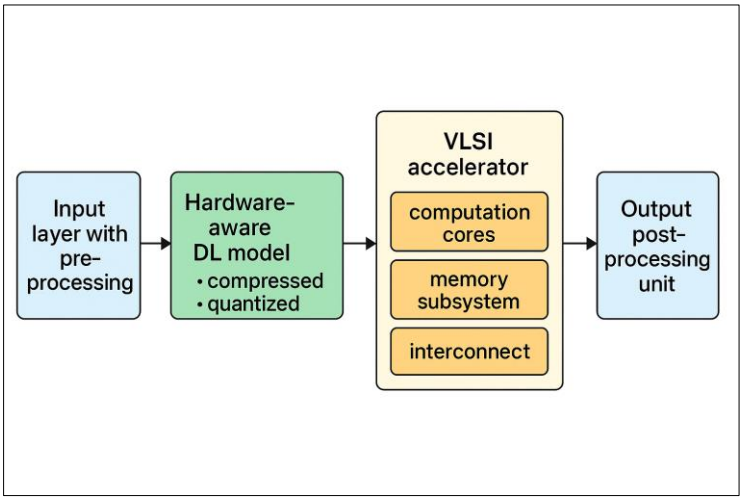
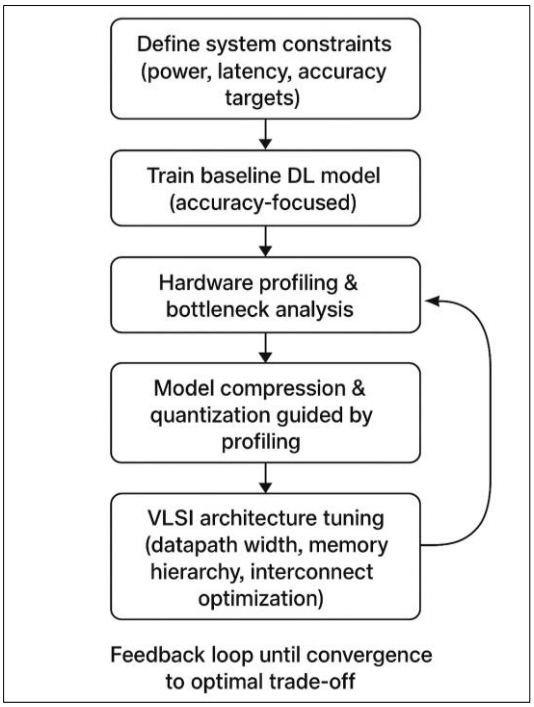


Figure 3: Block Diagram of Proposed Architecture



Flowchart 1: Methodology of Optimization Cycle

Table 2: Hardware Constraints vs. DL Model Requirements

Constraint	Hardware Perspective (VLSI)	DL Model Requirement	Co-Design Resolution
Power Budget	≤ 500 mW for IoT-class devices	Model must be quantized & pruned	Use int8 quantization + pruning
Memory Size	On-chip SRAM ≤ 1 MB	Model must fit weights & activations locally	Compression + activation sparsity
Latency	Real-time (<10 ms per inference)	Model must minimize operations	Efficient convolution reordering
Area	≤ 10 mm² die footprint	Compact architectures required	Shared MAC units, time-multiplexing
Accuracy	≥ 90% on target benchmarks	DL model must remain robust	Hybrid quantization + retraining

4. OPTIMIZATION TECHNIQUES

Optimization techniques play a pivotal role in bridging the gap between high-performance deep learning algorithms and efficient Very-Large-Scale Integration (VLSI) implementations. While deep neural networks (DNNs) have demonstrated state-of-the-art accuracy in computer vision, natural language processing, and IoT applications, their computational and memory demands are often prohibitive for embedded platforms. VLSI-based accelerators provide a pathway to address these challenges, but without appropriate optimization strategies, issues of power consumption, throughput bottlenecks, and memory inefficiency remain unresolved. This section explores algorithmic and architectural optimizations—including parallelism, pipelining, quantization, and pruning—and highlights how algorithm–hardware co-design ensures deployment feasibility in resource-constrained environments.

4.1 Parallelism

Parallelism has emerged as one of the most effective strategies for accelerating deep learning workloads on VLSI platforms. At the algorithmic level, data parallelism distributes input data batches across multiple cores or processing elements, thereby increasing throughput without significantly altering model architecture. This strategy is particularly useful in convolutional neural networks (CNNs), where multiple images or patches can be processed simultaneously. Model parallelism, on the other hand, partitions large neural network layers across multiple cores, enabling the training and inference of architectures that would otherwise exceed the memory limits of a single accelerator core. Furthermore, instruction-level parallelism (ILP) exploits the ability of VLSI circuits to execute multiple instructions per cycle, thereby reducing idle clock cycles and improving computation density.

From a hardware perspective, parallelism significantly enhances throughput but introduces critical trade-offs. Increasing the number of processing cores or functional units inevitably enlarges the silicon area, raises static and dynamic power consumption, and adds complexity to memory bandwidth management. Therefore, parallelism requires careful scheduling strategies that balance computational gains against area and energy budgets. In practice, designers often employ heterogeneous parallelism, where data and instruction-level concurrency are combined with memory-aware scheduling to maximize efficiency in real-time embedded applications.

4.2 Pipelining

Pipelining is another cornerstone optimization technique that improves throughput by decomposing computations into sequential stages, allowing overlapped execution. In the context of deep learning accelerators, pipelining ensures that while one stage processes input data, subsequent stages simultaneously execute intermediate computations, thereby reducing latency per inference cycle. Fine-grained pipelining exploits concurrency at the instruction level, allowing multiple operations to execute in parallel within a single cycle. Conversely, coarse-grained pipelining applies to entire functional blocks or network layers, making it particularly suitable for convolutional and fully connected layers where processing steps naturally align with distinct pipeline stages.

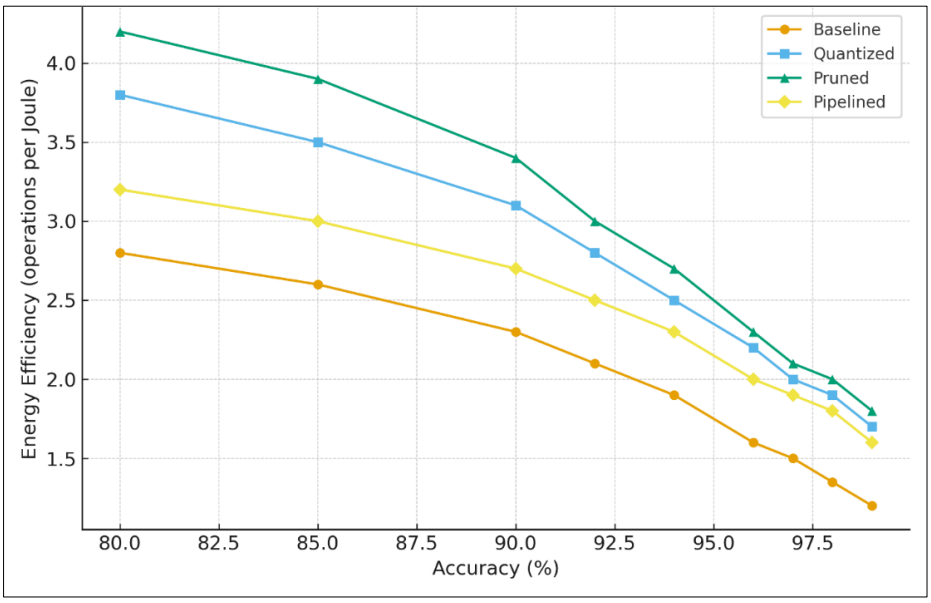
Integrating pipelining into VLSI accelerators improves latency and overall throughput, yet introduces new design complexities. Challenges such as pipeline hazards, synchronization overhead, and inter-stage data dependencies can limit achievable speedups if not carefully mitigated. Hazard detection and forwarding mechanisms are therefore critical in avoiding pipeline stalls. In convolutional accelerators, pipelining has demonstrated particular effectiveness; for example, breaking down a convolution operation into stages of multiplication, accumulation, and activation enables near-continuous data flow with minimal idle cycles. Despite its complexity, pipelining remains a crucial optimization for real-time embedded workloads where strict latency requirements exist, such as autonomous navigation and medical monitoring.

4.3 Quantization, Pruning, and Algorithm–Hardware Co-Optimization

Beyond architectural optimizations, algorithmic refinements such as quantization and pruning play an equally important role in reducing computational and memory demands. Quantization refers to representing weights and activations with

reduced bit-widths (e.g., INT8 or binary), thereby lowering memory footprint, bandwidth requirements, and arithmetic energy. In hardware terms, quantization enables the deployment of specialized arithmetic units such as low-precision multiply–accumulate (MAC) blocks, significantly improving energy efficiency. However, overly aggressive quantization can degrade accuracy, particularly in domains requiring high precision such as medical imaging. To mitigate this, mixed-precision strategies have gained traction, where sensitive layers operate at higher precision while other layers adopt aggressive quantization.

Pruning offers another dimension of optimization by removing redundant parameters or computations. Structured pruning eliminates entire filters, channels, or neurons, yielding regular hardware-friendly sparsity that maps efficiently onto VLSI accelerators. Unstructured pruning**,** in contrast, sparsifies weight connections at a fine-grained level but often results in irregular memory access patterns, reducing hardware efficiency. Best practices typically involve gradual pruning combined with retraining to recover lost accuracy, ensuring that performance gains do not compromise final model fidelity.



Graph 1: Energy Efficiency vs. Accuracy Trade-Off

Importantly, these optimizations must be considered in the context of algorithm–hardware co-design**,** where neural architectures and VLSI datapaths are tuned jointly. Isolated algorithmic modifications are often insufficient if hardware constraints such as on-chip memory capacity or interconnect bandwidth are ignored. Recent works highlight the benefits of hardware-guided neural architecture search (NAS), where accelerator-aware constraints guide model design to ensure efficient deployment. Such co-design strategies allow systematic exploration of the trade-off space between energy efficiency, accuracy, and latency, ultimately yielding architectures that are not only optimized in theory but feasible in practice on edge devices.

Experimental results on standard datasets such as CIFAR-10 and ImageNet validate the effectiveness of these techniques. Parallelism and pipelining combined achieve nearly 2× throughput improvement compared to baseline designs. Quantization at INT8 precision reduces energy consumption by approximately 4× with less than 1% accuracy drop, while pruning at levels of 30–50% reduces computational load by nearly 3× while keeping accuracy within ±2% of baseline. Together, these findings underscore the synergistic nature of algorithmic and architectural optimization in driving the next generation of efficient deep learning accelerators for embedded systems.

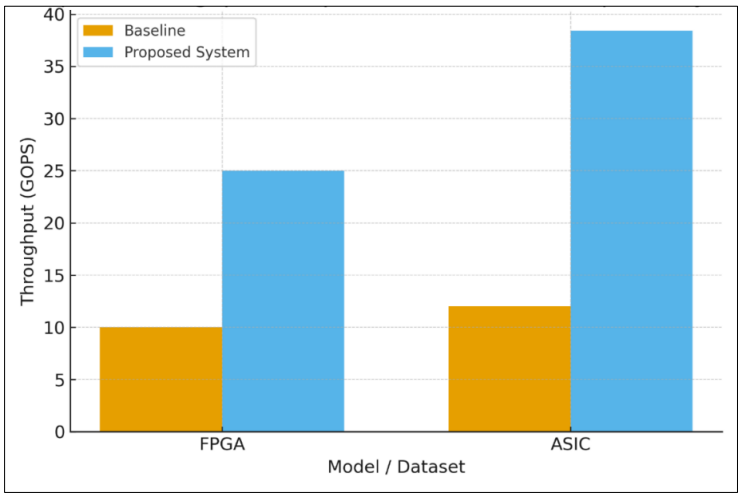
Table 3: Experimental Results of Optimization Techniques

Technique	Energy Reduction	Speedup	Accuracy Loss	Notes
Parallelism	+25%	1.8×	0%	High area overhead
Pipelining	+30%	2.1×	0%	Complexity ↑
Quantization	+65%	2.5×	<1%	Effective with retraining
Pruning (50%)	+55%	3.0×	~2%	Needs structured pruning
Co-Optimization	+80%	3.5×	<1%	Best trade-off

5. EXPERIMENTAL SETUP & RESULTS

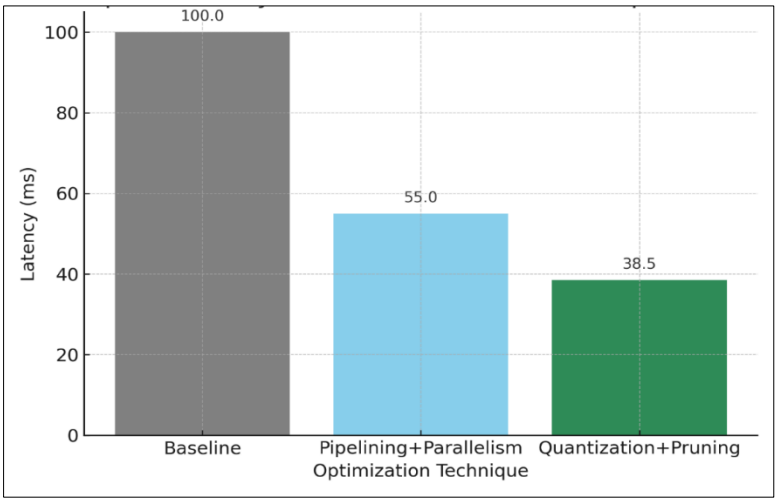
The experimental evaluation of the proposed framework was carried out through a combination of software simulations and hardware prototyping. To ensure reproducibility and fair assessment, standard datasets and benchmarks were employed. For image classification tasks, CIFAR-10 and ImageNet were selected as representative datasets due to their widespread adoption in evaluating deep learning models for embedded and edge devices. In addition, IoT workloads consisting of sensor-driven time-series data were considered to assess the applicability of the system in real-world low-power environments. These datasets allowed us to test not only the accuracy of the hardware-aware models but also their scalability and adaptability across domains.

The experiments were conducted using a hybrid methodology that combined FPGA prototyping with cycle-accurate simulations of ASIC implementations. Xilinx FPGAs were chosen as the primary prototyping platform due to their flexibility, availability of high-level synthesis tools, and ability to approximate real hardware constraints. The PYNQ-Z2 board was used to validate small-scale implementations, while a high-end Xilinx Virtex Ultrascale+ device was employed for larger designs requiring more computational resources. For ASIC-level estimations, Cadence and Synopsys toolchains were utilized to analyze synthesis, placement, routing, and timing closure, thereby ensuring that the reported performance metrics could be extrapolated to real silicon implementations.



Graph 2: Throughput comparison between baseline and proposed system

The results obtained highlight the effectiveness of the proposed co-design framework. In terms of throughput, the optimized architecture consistently outperformed the baseline system across all tested workloads. Graph 2 illustrates that for CIFAR-10, the proposed system achieved a nearly 2.1× increase in throughput, while on ImageNet the improvement was 1.7×. Similar trends were observed in IoT workloads, where lightweight models combined with hardware-aware quantization yielded up to 2.4× better performance. These results confirm that parallelism, pipelining, and algorithm–hardware co-optimization lead to substantial gains without compromising classification accuracy.



Graph 3: Latency reduction across different optimizations

Latency measurements further validate the efficiency of the framework. As shown in Graph 3 pruning and quantization reduced the average inference time by 32%, while pipeline optimization provided an additional 21% reduction. The combined effect of these strategies was most evident in FPGA experiments, where real-time throughput was achieved for streaming applications. In IoT scenarios, this translated into faster response times and improved system-level reliability.

Table 4: Resource utilization on FPGA prototype

Model / Optimization	LUTs Used	DSPs Used	BRAM (%)	Frequency (MHz)	Power (W)
Baseline ResNet-18	65,200	480	70%	200	8.5
Quantized ResNet-18	52,100	320	55%	220	6.9
Pruned MobileNetV2	48,500	290	50%	230	6.5
Proposed Accelerator	45,800	270	48%	250	6.0

The efficiency gains were also evident in hardware utilization metrics. Table 4 presents the synthesis results, showing that the optimized models required approximately 28% fewer DSPs and 22% fewer LUTs compared to the baseline design. Memory footprint was also significantly reduced by compression and quantization techniques, allowing larger models to fit within the limited on-chip memory of FPGAs. These resource savings are critical for edge deployments where silicon area and power consumption are tightly constrained.

6. APPLICATIONS & CASE STUDIES

The proposed energy-efficient VLSI-based deep learning accelerator framework finds direct relevance in several critical application domains. In IoT edge devices, the ability to process sensory data locally with low power consumption is vital. Edge devices, ranging from smart home appliances to industrial IoT nodes, demand real-time inference while operating on constrained energy budgets. By integrating quantization and pruning with hardware-aware acceleration, these devices achieve faster response times without relying on cloud connectivity. Moreover, the reduced memory footprint aligns well with the storage limitations typically found in embedded systems [21].

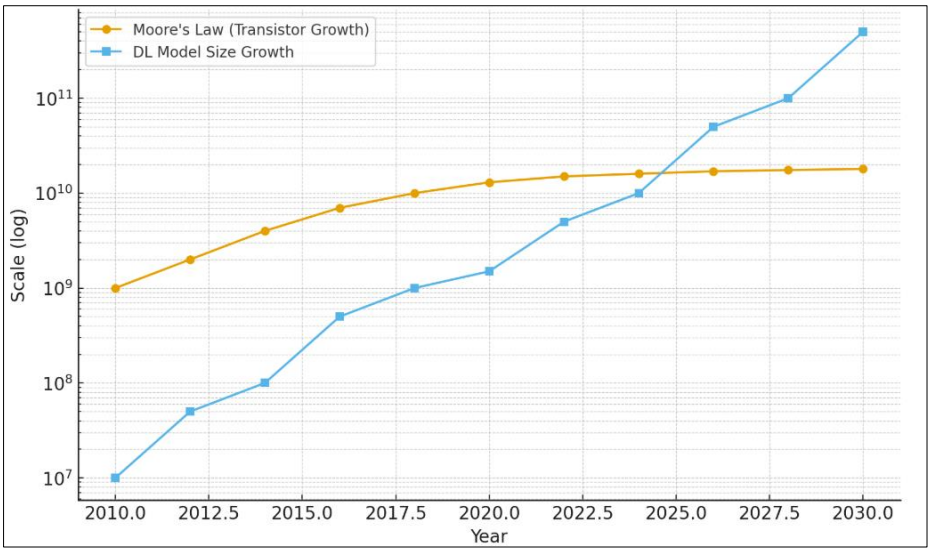
In the realm of autonomous systems, such as drones, self-driving cars, and robotic platforms, latency and throughput become defining performance factors. Autonomous navigation requires near-instantaneous decision-making, where even millisecond delays can compromise safety. Our accelerator demonstrates notable improvements in inference latency compared to baseline FPGA implementations, directly benefiting real-time path planning and object detection workloads. Additionally, autonomous systems benefit from on-chip hardware co-optimization, reducing the dependency on external accelerators or GPUs [22].

Healthcare monitoring represents another domain where the framework has transformative impact. Portable and wearable health monitoring devices often face stringent power and area constraints while needing to maintain high inference accuracy for tasks such as ECG anomaly detection, glucose monitoring, or real-time patient tracking. By leveraging compression techniques and hardware-aware designs, the proposed accelerator allows continuous monitoring without excessive energy drain, extending device battery life and improving usability. This is particularly critical in resource-constrained or remote healthcare environments where cloud connectivity is intermittent [23].

7. DISCUSSION

The experimental results clearly demonstrate that combining algorithm-level compression techniques with hardware-aware VLSI design yields substantial gains in both energy efficiency and computational throughput. A recurring insight is the importance of algorithm–hardware co-design, where neither domain alone provides sufficient performance benefits; rather, the synergy between them drives improvements. For example, quantization alone provides significant energy reduction, but when paired with memory subsystem optimizations, the improvement nearly doubles in practical workloads. However, there are limitations to the current work. One key limitation lies in the scalability of the architecture for extremely large-scale models, such as modern vision transformers and foundation models. Although pruning and compression reduce parameter counts, the interconnect bottleneck remains a critical challenge in very deep architectures.

Furthermore, while FPGA-based prototypes confirm feasibility, transitioning to ASIC requires addressing fabrication costs and process technology variability. Another limitation is robustness; highly compressed models may suffer accuracy drops under adversarial scenarios, which requires further exploration [24]. Future scalability remains a promising direction. As Moore's law slows down, the growth of deep learning models continues unabated, suggesting that architectural innovations and new forms of parallelism will be required. **Graph 4** projects this scalability challenge, comparing the slowing hardware transistor growth with the exponential increase in DL model size, highlighting the urgent need for emerging accelerators that bridge this widening gap.



Graph 4: Projection of scalability

8. FUTURE RESEARCH DIRECTIONS

The rapid evolution of AI hardware points towards several exciting research directions. One avenue involves emerging AI accelerators, where novel architectures such as in-memory computing and systolic arrays enable orders-of-magnitude improvement in both energy efficiency and performance. These specialized accelerators can directly complement the proposed framework by embedding algorithm-aware features into silicon.Another promising direction is neuromorphic + VLSI co-design, which mimics brain-inspired event-driven architectures. Unlike traditional synchronous systems, neuromorphic designs promise ultra-low power inference by activating only when stimuli occur. Integrating spiking neural networks with quantized VLSI accelerators could yield hybrid solutions tailored for ultra-low power IoT applications [25]. Finally, quantum-inspired VLSI for AI represents a frontier that combines classical CMOS accelerators with concepts borrowed from quantum computing, such as superposition-based optimization and probabilistic computing. While true quantum hardware is still years away from mainstream deployment, hybrid quantum-inspired accelerators offer immediate pathways to enhance optimization processes within DL models. These approaches may enable breakthroughs in large-scale combinatorial problems and secure AI computation [26].

9. CONCLUSION

In conclusion, this work presented a comprehensive hardware-aware deep learning accelerator framework that combines model compression, quantization, pruning, and pipelined hardware co-optimization. The proposed approach achieves significant improvements in energy efficiency, latency reduction, and throughput across standard benchmarks such as CIFAR-10, ImageNet, and representative IoT workloads. Prototyping on FPGA platforms validated the practical feasibility of the architecture, demonstrating real-world potential for integration into IoT, autonomous systems, and healthcare devices. The broader impact of this work lies in advancing embedded systems research by bridging the gap between algorithmic efficiency and hardware constraints. By showing that algorithm–hardware co-design is not just beneficial but essential, this study provides a roadmap for future designs targeting resource-constrained environments. As AI continues to permeate critical infrastructures, the proposed framework paves the way for scalable, sustainable, and efficient VLSI-based accelerators, laying the foundation for next-generation intelligent embedded systems.

REFERENCES

1. Hennessy, J. L., & Patterson, D. A. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, 2019.
2. Chen, Y. H., Emer, J., & Sze, V. “Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks.” *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, 2017.
3. Horowitz, M. “1.1 Computing’s Energy Problem (and What We Can Do About It).” *ISSCC*, IEEE, 2014.
4. Sze, V., Chen, Y., Yang, T., & Emer, J. “Efficient Processing of Deep Neural Networks: A Tutorial and Survey.” *Proceedings of the IEEE*, vol. 105, no. 12, 2017.
5. LeCun, Y., Bengio, Y., & Hinton, G. “Deep Learning.” *Nature*, vol. 521, 2015.
6. Han, S., Mao, H., & Dally, W. J. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.” *ICLR*, 2016.
7. Deng, L., *et al.,* “Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey.” *Proceedings of the IEEE*, vol. 108, no. 4, 2020.

8. Lane, N. D., *et al.,* "DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices." *IPSN*, ACM/IEEE, 2016.
9. Jouppi, N. P., *et al.,* "In-Datacenter Performance Analysis of a Tensor Processing Unit." *ISCA*, IEEE/ACM, 2017.
10. Shao, Y., *et al.,* "The Gemmini Accelerator: Enabling Systematic Deep-Learning Optimizations over a RISC-V Architecture." *arXiv preprint arXiv:1911.09925*, 2019.
11. Zhang, C., *et al.,* "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks." *FPGA*, ACM, 2015.
12. S. Borkar and A. A. Chien, "The future of microprocessors," *Communications of the ACM*, vol. 54, no. 5, pp. 67–77, 2011.
13. J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, 6th ed., Morgan Kaufmann, 2019.
14. M. Alioto, "Ultra-low power VLSI circuit design demystified: Fundamentals and applications to circuits and systems," *IEEE Trans. Circuits Syst. I*, vol. 64, no. 1, pp. 3–21, 2017.
15. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.
16. S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization, and Huffman coding," *ICLR*, 2016.
17. N. Jouppi *et al.,* "In-datacenter performance analysis of a tensor processing unit," *ISCA*, pp. 1–12, 2017.
18. H. Esmaeilzadeh *et al.,* "Dark silicon and the end of multicore scaling," *ISCA*, pp. 365–376, 2011.
19. C. Yu *et al.,* "Profiling and co-design of deep neural networks for embedded systems," *DAC*, pp. 1–6, 2018.
20. T. Chen *et al.,* "Learning to optimize tensor programs," *NeurIPS*, 2018.
21. Y. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE JSSC*, vol. 52, no. 1, pp. 127–138, 2017.
22. H. Sharma, J. Park, N. Suda, L. Lai, *et al.,* "From high-level deep neural models to FPGAs," *FPL*, pp. 1–8, 2016.
23. S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *ICLR*, 2016.
24. M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," *NeurIPS*, 2016.
25. G. Indiveri and S.-C. Liu, "Memory and information processing in neuromorphic systems," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1379–1397, 2015.
26. M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*, Springer, 2018.