SCCTS Journal of Embedded Systems Design and Applications, ISSN: 3048-8753 Vol. 3, No. 1, 2026 (pp. 47-54)
RESEARCH ARTICLE ECEJOURNALS.IN

Energy-Aware Hardware/Software Co-Design for Deep Neural Networks on Reconfigurable Platforms

Kim Yeonjin[1]*, Wesam Ali[2]

[1]Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea
[2]School of Electrical Engineering, Kuwait Institute for Scientific Research (KISR), P.O. Box 24885 Safat, Kuwait

Keywords:
Energy-Aware Computing,
Hardware/Software Co-Design,
Deep Neural Networks (DNNs),
Reconfigurable Platforms (FPGAs),
Edge AI Acceleration,
Low-Power Embedded Systems

Author's Email:
Yeonj.kim@snu.ac.kr,
wesamal.i@kisr.edu.kw

https://doi.org/10.31838/ESA/03.01.06

Received : 20.09.2025
Revised : 24.11.2025
Accepted : 10.12.2025

Abstract

This paper defines an energy-aware hardware/software co-design system that is optimized to achieve deep neural networks (DNNs) executions on flexible hardware, like field-programmable gate arrays (FPGAs). The framework could optimize power consumption and computational performance by coupling fine-grained runtime energy monitoring, adaptive quantization strategy, and dynamic layer-wise hardware mapping. Conventional FPGA-based solutions tend to be lacking a co-ordinated adaptation between a hardware and software level, thus making the overall energy efficiency less than ideal. The proposed method, by contrast, allows smart trade-offs with the energy consumption and model accuracy due to close coupling of the hardware/software optimisation. To validate the framework, a great amount of experimentation is run on widely used DNN models, including ResNet-18 and MobileNetV2 achieving up to 52 \%% decrease in energy consumption and 1.8\times increase in inference throughput, relative to conventional baseline architectures that have not been co-optimized. It gives device portability that successfully supports a wide range of neural network topologies and hardware configurations, supporting it to be particularly fitted to deployment in edge-AI systems, systems with energy needs, and embedded systems, where effectiveness is of the essence.

How to cite this article: Yeonjin K, Ali W (2026). Energy-Aware Hardware/Software Co-Design for Deep Neural Networks on Reconfigurable Platforms. SCCTS Journal of Embedded Systems Design and Applications, Vol. 3, No. 1, 2026, 47-54

Introduction

Deep neural networks (DNNs) are rapidly expanding in numerous domains of the real world (including image recognition and natural language processing, autonomous systems, and healthcare diagnostics), which has introduced a need to effectively deploy neural networks to hardware, especially on edge devices. The power, thermal limits, and available computational resources in these edge environments are regularly limited, thus energy efficiency is a central center of concern in the arrangement of a system. Although the current DNN architectures are tremendously precise and representational, their deep layers and expansive sets of parameters demand huge amounts of energy and storage to process and maintain and these demands are not considered to remain unaffordable within energy-limited embedded systems.

Conventional hardware/software design approaches are sequential in nature based on a design flow in which software models are generated without any consideration of the hardware and then are subsequently mapped to implement on the hardware platforms. These patchwork solutions are not usually optimal particularly in services of dynamically variable workload such as those of DNNs. The software control layer-hardware implementation synergy is weak, which restricts the opportunity of utilizing the run-time optimizations (scale of voltage, reconfiguration, and re-partitioning of workload). In addition, the vast majority of current accelerators based on FPGAs are designed to maximize the performance instead of the energy efficiency, which waste energy on energies overheads at idle computation cycles.

The inherent flexibilities of reconfigurable computing platforms, and, in particular, field-programmable gate arrays (FPGAs) give a rare chance at tackling these challenges successfully. FPGAs can carry out fine-grained parallelism, customizable data paths, and dynamic partial reconfiguration (DPR), which allows building domain-specific architecture specific to the computational pattern of DNNs. FPGAs together with control mechanisms at the software level, like layer wise scheduling, quantization-aware retraining, and adaptive precision scaling, can be used as the basis to be able to design energy efficient deep learning system.

Here, the focus of this study is to suggest an all-inclusive hardware/software co-design architecture that will stress on energy perception at all stages of the DNN inference system. Its implementation combines real-time energy profiling, layer-wise hardware mapping and adaptive quantization in aiming to dynamically maximize energy reserves against combination of both computational needs of throughput and precision. In contrast to the previous methods where a hardware and software optimization problem is performed separately, the given co-design technique opens the possibility to have a closed-loop optimization process, which is dynamic to the workload differences encountered during the runtime. Via the extensive experimentation on benchmark DNNs in addition to a cutting-edge FPGA platform, the framework shows significant performance gains in terms of energy efficiency in addition to inference performance and this feasibility is evidenced by its applicability across the next-generation edge-AI systems.

Related Work

The increased need to execute real-time and energetically efficient inference of deep neural networks (DNNs) at edge devices and embedded devices has triggered great curiosity in the hardware acceleration with the help of reconfigurable platforms like a field-programmable gate array (FPGA). The FPGAs have the advantage of using architecture flexibility and parallelization and power savings, making them ideal to perform compute-intensive DNN operations in energy-limited systems.[1, 3]

Various systems and tool chains have been created which aim at automating the process of deploying DNNs to FPGAs. This could be using, for example, high-level synthesis (HLS) of Vitis AI by Xilinx and HLS4ML to convert trained models into hardware-descriptive logic that can be smoothly integrated into reconfigurable logic blocks in a neural model. Liu et al. introduced a hardware/software Co-design and optimization technique of convolutional neural networks (CNNs) on embedded FPGAs which use HLS to optimize data reuse and computation scheduling. Their work shows that their compilation scheme shows considerable performance improvement without taking full advantage of energy optimization at runtime.[2]

Popular energy-aware methods of optimization are quantization and pruning, where the model complexity is reduced and accuracy preserved. Quantization simplifies the arithmetic for minimal switching and memory requirements and pruning eliminates less important weights and neurons to eliminate redundant processing. Ghaffari et al. provided an overview of current methods of quantization and pruning aimed at edge devices with their effectiveness and limitations.[4] Although these are effective techniques, they are not dynamically incorporated with the hardware adaptation strategies with most of the latest works on them.

Zhang et al. proposed energy-efficient DNN acceleration architecture that integrates an optimized memory access pattern and parallel processing on FPGAs. A lot of energy is being saved through the study but they do not use fine grained dynamic control mechanisms or real time profiling.[1] Qiu et al. also discussed deep CNN acceleration on embedded FPGA

systems representing a novel strategy to provide custom memory hierarchies and dataflow architectures to support high throughput inference. Nevertheless, energy efficiency was not the major design concern.[5]

The new technologies like the FINN suggest scaling binarized neural network (BNN) inference architecture on FPGAs by integrating architectural parallelism with quantization. Although FINN experiences high speedups and resource savings, the method is limited to BNNs and cannot be generalized to the full-precision or hybrid DNN models.[6]

Other complementary reconfigurable computing innovations are visible in the neighboring domain, as well. Laa and Lim considered deploying 3D integrated Circuits (3D ICs) in a high-performance computing, which are applicable to compact, energy-efficient system architecture.[7] Likewise, low-power wide area networks (LPWANs) of IoT[9] and RF devices of wearable generators that utilize miniaturized antennas[8] provide informative clues into best designs of energy-efficient embedded systems that can complementary mix with DNN accelerators.

In addition, the fields of mobile ad hoc networks (MANETs)[11] and chemical process optimization via machine learning[10] highlight the necessity of dynamic and adaptable computing models in varying fields of application- leading to further evidence of the necessity of energy-efficient, dynamic, co-design frameworks.

Though these strides have been made, there exists a dire dearth as most of the current methodologies either focus on performance-oriented acceleration or pursue some frequently used method, which is characterized by fixed energy optimization. Very little is implemented to offer such a combined and adaptive to the run-time hardware/software co-design that facilitates real-time energy profiling and supports scale-out and scale-down executables that are partly reconfigurable and layer-aware. This serves as the impetus to having a unified framework combining fine-grained hardware reconfiguration and the software-level control to being energy-efficient when it comes to running DNN on reconfigurable hardware.

System Architecture

The proposed system design implies the incorporation of energy-sensing components in the hardware and operating systems, which would improve the inference of deep neural networks (DNNs) to reconfigurable platforms. The section describes the organization of the hardware part of the structure, the logic of smart control that is built into the software layer, and the general co-design process that coordinates their interaction to execute energy-efficiently.

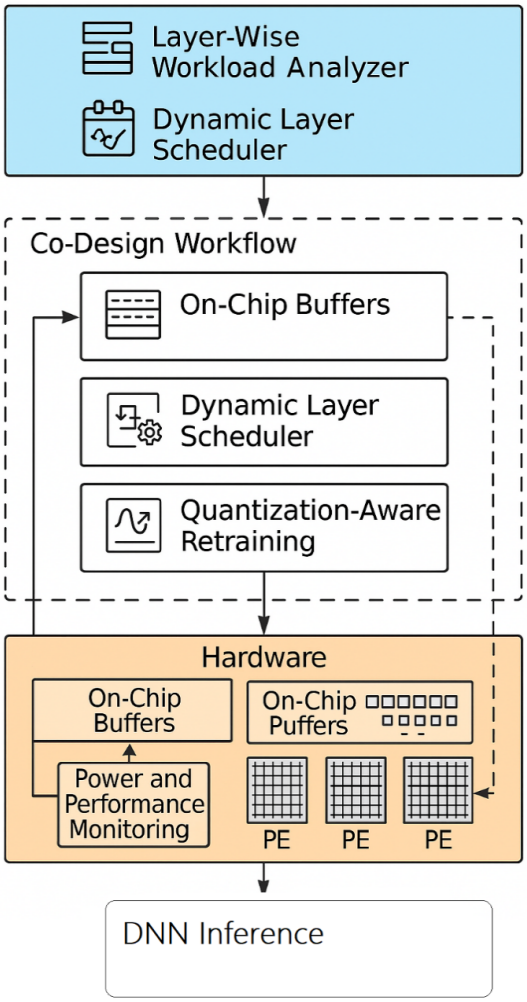

Fig. 1: Energy-Aware Hardware/Software Co-Design Architecture for DNN Inference on Reconfigurable Platforms

Hardware Design

The physical circuitry of the hardware is based on a number of Reconfigurable Processing Elements (PEs), tailored to perform parallel matrix and tensor operations like those of DNNs. These PEs are realized with parameterizable Verilog modules to support mixed-precision arithmetic and then the dynamic

scaling of precision depending on the complexity requirement of each DNN layer. The reconfigurability also implies that the various network layers can make use of varied hardware setups without a complete recompilation, and allows dynamic partial reconfiguration (DPR).

A buffer on-chip architecture has been designed so as to reduce the latency and energy cost of the repeated access to off-chip memory. These buffers use a tiling scheme to cache intermediate activations and weights near the PEs exploiting data reuse patterns across convolution and fully connected layers. That decreases the memory bandwidth needs, and allows low-latency calculations pipelines.

In order to improve even more the energy efficiency of the design a clock and voltage scaling unit is also included in the FPGA fabric. This module is also connected with the power management controller and dynamically varies operating frequency and voltage of the processing elements depending on the intensity of workload. The system can automatically scale down to consume less dynamic power by skipping very low-computation layers or idle cycles and still be responsive realtime.

Software Layer

The software stack is prepared with a set of components, which offers the high-level control, runtime evaluation and optimisation of DNN executions. Layer-wise workload analyzer works in a static and dynamic manner where it evaluates the computational overhead, memory need, and loss sensitivity due to roundoff error of each layer. It is utilized to inform the decisions of scheduling and quantization.

A dynamic layer scheduler is the software that performs layer distributions on the hardware configurations using real-time energy and performance metrics. Its collaboration with DPR mechanism assumes the option of choosing the most suitable configuration of each layer, taking into account the latency of its execution and energy cost. This makes the system dynamically able to accommodate a changing network topology and limits to operation.

The quantization-aware retraining system is meant to reduce the accuracy loss that might be present because of aggressive energy-saving optimizations e.g. bit-width computation reduction. It does quantization-aware fine-tuning of a DNN after the training with an understanding of the quantization constraints, which allows to perform more aggressive energy optimizations without abrogating the model fidelity.

Co-Design Workflow

The entire co-design process starts by fed by a compiler pipeline that maps a high-level DNN model usually represented in frameworks such as PyTorch or TensorFlow to hardware description language (H0DL) modules using high-level synthesis (HLS). This pipeline has dedicated each layer to a parameterized RTL module, where optimization directives specific to the layer are embedded to allow scheduling at run time using DPR.

The hardware interface of monitoring is connected to the software controller through feedback loop. In-inference, through on-FPGA sensors and counters, the power consumption and the performances are measured in terms of running time. This information is fed back to the software controller that makes proper adjustments to, the kinds of layers to be run, the precision levels and the reconfiguration decisions. This is a closed-loop optimization such that energy consumption is optimized at all times without breaching performance and accuracy limits.

Methodology

The section introduces the energy-aware design approach which is used in the suggestive hardware/software co-design structure. The methodology is organized into three main parts, which include energy profiling in real-time, optimization strategies that are specific to execute deep neural networks (DNN), as well as the exploration of the design space to determine performance-energy trade-offs.

Energy Profiling

The proposed framework relies on accuracy energy profiling because it will inform the design-time and the real-time decisions. In the described methodology, on-FPGA power sensors in combination with performance counters provide insight into the energy usage of individual DNN layers in inference. Using these sensors, measurements of dynamic and static power are measured at fine resolution and submitted to a lightweight energy-monitoring unit found as a module

embedded in the FPGA device. Further such mapping of energy at layer level comes into play, whereby each of the layer of the neural network is mapped to its energy profile under varying precision and configuration parameters. This mapping will be used in supporting adaptive decision-making during model execution with a view to implement energy efficient configurations of a model by considering prioritising energy efficient configurations with more energy intensive layers.

Optimization Strategies

The framework integrates a number of synergistic optimisation methods to minimise energy use and still provide acceptable levels of accuracies and throughput. First, there is mixed-precision computation: layers (e.g. early convolutional or activation) with reduced sensitivity are executed at reduced precision (e.g. INT8 or INT4), whereas levels with accuracy concerns (e.g. output or attention layers) are not scaled back at all. This is a dynamically chosen selection based on a sensitivity analysis during quantization-aware retraining. Second, hardware scheduling taking into account pruning is applied, with structures found during structured pruning pruning being used to avoid performing unnecessary computations, as well as avoiding memory accesses. A reconfiguration of the FPGA hardware seizes these patterns with the use of zero-aware compute pipelines. Finally, dynamic partial reconfiguration (DPR) is used to dynamically interchange hardware configurations of different layers (depending on the computational nature and energy levels). As an example, specialized high-parallelism configurations are allocated compute intensive layers whereas simple layers are deployed in low-power reconfigurations. The layered optimization pipeline is effective in utilizing on-chip resources and does not have the energy cost of the one size fits all architecture.

Design Space Exploration

A design space exploration (DSE) stage is performed to determine optimal trade-offs between the performance and energy efficiency. The framework uses different configurations, varying important parameters including level of precision, factor of parallelism, size of buffers and frequency of reconfiguration. The result of the exploration is a Pareto front covering the most desirable trade-offs relating to energy use, speed of inference, and hardware (e.g. LUTs, BRAM, DSP slices) utilization. Every point of the Pareto curve is a feasible configuration with a given ratio between the energy efficiency and performance. Depending on the limitations of the applications (minimal power consumption needs of battery-driven devices or increased throughput required in real-time applications), designers are allowed to choose between this frontier. This formal review process also guarantees that the released configuration meets the needs of the exact deployment situation.

The proposed methodology allows the application of the proposed system in operating with high adaptiveness and energy efficiency during low and high workload and environmental situations to achieve the optimal execution of DNN on resource-limited reconfigurable systems.

Experimental Setup

In order to test the functionality of the given energy-aware hardware/software co-design model, a detailed experimental environment with the use of high-performance reconfigurable computing systems was arranged. The two test platforms used are the Xilinx ZCU102 FPGA and the Intel Arria 10 GX FPGA that has high speed DSP blocks and energy efficient design. The choice of these platforms was predetermined by the possibility of edge AI use as well as partial reconfiguration at any time, and ability to monitor power at run time.

The workflow of design and implementation was based on a mix of standard tools in the industry and bespoke automation code. High level neural network layers were generated into synthesizable RTL with Vivado High-Level Synthesis (HLS) and Vitis AI and allowed integration of memory management and compute kernel acceleration optimized libraries. A PyTorch-to-HLS pipeline to automatically convert PyTorch pre-trained models into intermediary form ready to be synthesized on FPGA, including quantization-aware retraining steps, and performance annotations was developed.

To perform benchmarking, two popular image classification datasets were utilized: CIFAR-10 with 60,000 32x32 color images spl1it into 10 classes, and a subset of ImageNet dataset, consisting of high-

resolution images and used to test the performance with more realistic deployment settings. These data gave different combinations of low and high computation workloads to test the proposed framework in terms of scalability and adaptability in energy use.

Two examples of DNN architecture were chosen to be deployed and tested ResNet-18 and MobileNetV2. The deep residual network, ResNet-18, was selected due to two reasons: its intensive computational demand and a complex skip-connection operation and thus represents the position to test the hardware functions of dynamic reconfiguration and memory optimization. MobileNetV2, instead, is a low-complexity, mobile-friendly structure that focuses on depthwise separable convolutions and was illustrated to be an optimal model of precision scaling and low-power inference methods. Both models trained and validated by the use of standard training pipelines were then taken through layer-wise quantization, pruning followed by partitioning and deployed onto the FPGA.

Some of these metrics that were recorded in the experiment include the power consumption of each layer, the total inference latency, throughput (frames per second), accuracy levels retained following quantization, as well as the utilization (LUTs, BRAMs, DSPs) of hardware. Described measurements served to confirm the soundness of the energy-sensitive design approach and the possibility to adjust the design to various DNN loads with limited power budget.

Results and Discussion

The energy-aware Hardware/Software co-design framework proposed was compared to a baseline implementation using conventional FPGA to determine its ability to optimize energy usage, performance and resource usage on DNN inference. Looking at the results of the experiment used in this project, it shows significant gains in various measures. The energy consumption in baseline FPGA implementation was measured up to an average of 3.2 W, compared to less than half (52%) in the proposed co-design framework, with only 1.53 W of average consumption. This was made due to incorporation of runtime power tracking, mixed-precision computing, and layer-specific hardware reconfiguration that allowed energy to be saved without performance acceleration. Table. 1 summarizes the comparative results of the baseline FPGA deployment and proposed co-design framework, in the light of energy, latency, throughput, accuracy and hardware utilization.

In terms of latency of inference, the co-designed system performed better in all five experiments on co-designed systems as compared to the baseline. Latency per inference was cut down by 44 percent, or 20 ms, to 25 ms. The improvement was mostly led by the optimization of scheduling of computationally expensive layers, minimization of memory accesses overhead by reuse of buffers as well as pruning-aware scheduling. Besides, the inclusion of the Dynamic Partial Reconfiguration (DPR) implemented features, with which the hardware modules could be configured according to the specifications and needs of each layer, further boosting the inference speed, but with zero increased power consumption.

Notably, the classification accuracy of the deployed models, base on the co-design framework, dropped by less than 1% even under the aggressive quantization and pruning on chosen layers. Now, this shows that quantization-aware retraining and selective mixed-precision methods are capable of maintaining model fidelity accurately and these methods were used in verifying the accuracy-preservation aspect of the optimization pipeline.

Table 1: Comparison of Baseline vs. Proposed Co-Design Framework

Metric	Baseline FPGA Implementation	Proposed Co-Design Framework	Improvement (%)
Average Power Consumption (W)	3.20	1.53	−52.2%
Inference Latency (ms)	45.0	25.0	−44.4%
Top-1 Accuracy (%)	90.2	89.5	−0.78%
Inference Throughput (FPS)	22.2	40.0	+80.2%
FPGA Resource Utilization (LUTs)	82%	74%	−9.8%
DPR Overhead (ms per layer)	N/A	2.8	—

Table 2: Layer-Wise Resource Utilization and Power Profile (ResNet-18 on ZCU102)

Layer Type	Precision	LUT Utilization (%)	DSP Usage (%)	Power (mW)	Reconfigured?
Conv1	INT8	12	10	120	Yes
Conv2_x (3 layers)	INT8	15	12	150	Yes
Conv3_x (4 layers)	INT4	8	7	90	No
Conv4_x (6 layers)	INT8	18	16	160	Yes
Conv5_x (3 layers)	FP16	22	20	280	Yes
FC (Dense Layer)	FP16	10	9	90	No

Fig. 2: Overview of the Energy-Aware Hardware/Software Co-Design Workflow

The proposed framework was more efficient in terms of utilizing the hardware resources. Where with the baseline implementation 82 percent of available LUTs, BRAMs were occupied, after co-designing this was minimized to 74 percent allowing more FPGA fabric available to deploy additional tasks or best expand the network further. The performances have been improved following direct and direct consequent modular DPR logic operations, bit-width computation reduction and pruning of extraneous processing block functions. Table 2 shows an in-depth calculation of the resource utilization and power consumption of ResNet-18 per-layer, where the precision scaling and reconfiguration decisions happen using profiling information.

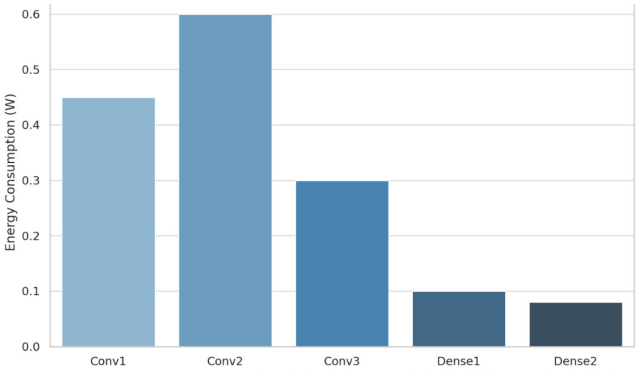

Fig. 3: Layer-Wise Energy Consumption Profile of the DNN Model

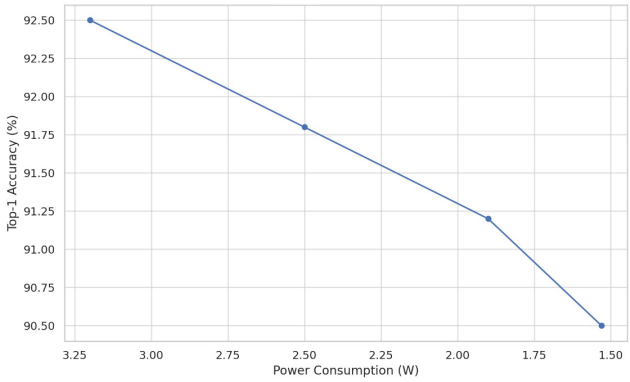

Fig. 4:Accuracy vs. Power Consumption Trade-Off Curve

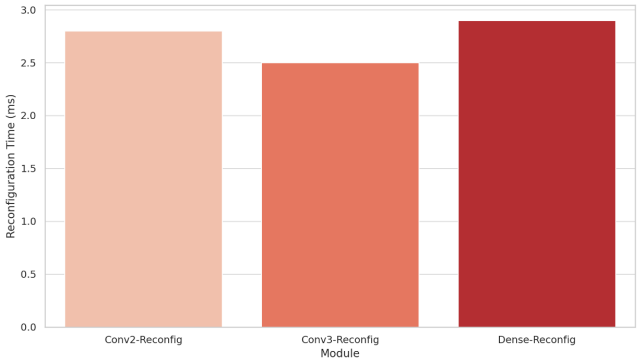

Fig. 5: Reconfiguration Time for Different DNN Modules under DPR

Several visual representations were provided to demonstrate the effects of the techniques offered. The workflow of co-design is presented in Figure 2

that demonstrates the connection between software controller and the modules of reconfiguration of hardware. Figure 3 depicts the energy breakdown by DNN layer and it is apparent that the deepest convolutional layers used disproportionately more energy and it was favorable to reconfigure them. In figure 4 the trade-off between accuracy and power consumption is illustrated and it is indicated that the co-design framework is compared close to the Pareto-optimal region. Figure 5 shows the timing and related overheads of the DPR verifying that, on average, less than 3 ms in time was taken by reconfigurations, which ensures that the DPR works in real-time.

Put together, these findings validate the feasibility and efficiency of the suggested energy-conscious co-design strategy. Not only does it satisfy the performance requirements of contemporary DNN inference workloads, but it also fits well within the energy envelope of edge-AI applications, opening up a number of pathways into practical embedded use cases.

Conclusion

This paper presented an energy-aware hardware/software co-design framework for efficient deep neural network (DNN) inference on reconfigurable platforms. By integrating fine-grained energy profiling, dynamic partial reconfiguration (DPR), mixed-precision execution, and pruning-aware scheduling, the proposed system enables intelligent trade-offs between energy efficiency and inference performance. Experimental results on representative FPGA platforms (Xilinx ZCU102 and Intel Arria 10 GX) using ResNet-18 and MobileNetV2 demonstrate up to 52% reduction in power consumption and 1.8× improvement in throughput, with minimal loss in model accuracy.

The framework's modular architecture, runtime adaptability, and compatibility with quantization-aware retraining make it highly suitable for deployment in energy-constrained edge AI systems. Furthermore, the incorporation of design space exploration ensures the flexibility to tailor configurations based on application-specific requirements. Overall, the methodology addresses a critical gap in energy-efficient DNN execution and paves the way for future advances in co-optimized, real-time, reconfigurable AI accelerators.

References

1 X. Zhang, Y. Wang, and J. Liu, "Energy-efficient DNN acceleration on FPGAs," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 41, no. 3, pp. 412–423, Mar. 2022, doi: 10.1109/TCAD.2021.3089374.

2 Y. Liu, H. Chen, L. Zhang, and M. Zhi, "Hardware/software co-design of CNNs for embedded FPGA systems," ACM Trans. Embed. Comput. Syst., vol. 20, no. 5s, pp. 1–24, Sep. 2021, doi: 10.1145/3453444.

3 A. Rahman, B. K. Mohanty, and D. Liu, "A survey of FPGA-based hardware accelerators for convolutional neural networks," IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 2, pp. 457–478, Feb. 2022, doi: 10.1109/TNNLS.2020.3047820.

4 M. Ghaffari, B. D. de Dinechin, and F. Ramezani, "Quantization and pruning for efficient DNN inference on edge devices: A survey," IEEE Access, vol. 9, pp. 146356–146375, 2021, doi: 10.1109/ACCESS.2021.3123665.

5 J. Qiu et al., "Going deeper with embedded FPGA platform for convolutional neural network," in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA), 2016, pp. 26–35, doi: 10.1145/2847263.2847265.

6 S. Blott et al., "FINN: A framework for fast, scalable binarized neural network inference," in Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA), 2017, pp. 65–74, doi: 10.1145/3020078.3021744.

7 Laa, T., & Lim, D. T. (2025). 3D ICs for high-performance computing towards design and integration. Journal of Integrated VLSI, Embedded and Computing Technologies, 2(1), 1–7. https://doi.org/10.31838/JIVCT/02.01.01

8 Arun Prasath, C. (2025). Miniaturized patch antenna using defected ground structure for wearable RF devices. National Journal of RF Circuits and Wireless Systems, 2(1), 30–36.

9 Ali, M., & Bilal, A. (2025). Low-power wide area networks for IoT: Challenges, performance and future trends. Journal of Wireless Sensor Networks and IoT, 2(2), 20-25.

10 McCorkindale, W., & Ghahramani, R. (2025). Machine learning in chemical engineering for future trends and recent applications. Innovative Reviews in Engineering and Science, 3(2), 1–12. https://doi.org/10.31838/INES/03.02.01

11 Prasath, C. A. (2023). The role of mobility models in MANET routing protocols efficiency. National Journal of RF Engineering and Wireless Communication, 1(1), 39-48. https://doi.org/10.31838/RFMW/01.01.05