SCCTS Transactions on Reconfigurable Computing, ISSN: 3049-1533 Vol. 3, No. 2, 2026 (pp. 1-10)
RESEARCH ARTICLE ECEJOURNALS.IN

Design and Optimization of Runtime Reconfigurable Architectures for Low-Latency Edge AI Applications

Leila Ismail[1]*, Hee-Seob Kim[2]

[1]Faculty of Management, Canadian University Dubai, Dubai, United Arab Emirates
[2]Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea

Keywords:
Runtime Reconfigurable Architecture;
Edge AI;
Dynamic Partial Reconfiguration (DPR); Low-Latency Inference;
FPGA Acceleration; Hardware/ Software Co-Design;
Reinforcement Learning Scheduler;
Real-Time AI Processing;
Energy-Efficient Edge Computing.

Author's Email:
leila.ism@ead.gov.ae,
h.s.kim@snu.ac.kr

DOI: 10.31838/RCC/03.02.01

Received :13.11.2025
Revised : 18.01.2026
Accepted : 20.04.2026

Abstract

The surrounding network edge has produced such a huge number of artificial intelligence (AI) applications that the demand to perform real-time in the strict latency, power consumption, and computing forms is growing. Conventional cloud-based methods of inference cannot effectively perform in such settings because of the latency and privacy issues associated with it. This paper suggests a new type of low-latency Edge AI-focused runtime reconfigurable architecture (RRA) to meet these challenges. The design is based on the dynamic partial reconfiguration (DPR) on FPGAs that allows hardware to be adapted dynamically on-demand depending on the shifting AI workloads in real-time. In contrast to the fixed hardware designs one can find in the current hardware designs, our RRA implements optimized hardware modules of various inference tasks, including convolution, activation, and pooling layers, and dynamically loads them into the hardware to maximize resource usage and reduce idle logic. For facilitating the challenging work of scheduling tasks in the system, a reinforcement learning task scheduler is integrated into the system and it works by predicting workload patterns and orchestrating reconfiguration events at low overheads. In addition, a performance-energy optimization layer is used so that when the architectural changes are made, they do not violate the energy budget or the energy budget on the edge device or its thermal constraints. Standard CNN benchmarking on the entire system is undertaken on Xilinx Zynq UltraScale+ MPSoC platform,and the standard CNN benchmarks include ResNet-18 and MobileNetV2. Experimental outcomes show that inference latency can be decreased by as much as 53 percent and power reduction by as much as 41 percent when compared to non-dynamically provisioned baseline designs. Both factors make the framework scalable to a wider range of neural workloads with negligible reconfiguration delays owing to overlapping task execution mechanisms and bitstream caching. The work presented is an indication of the practicality and success of runtime reconfigurable hardware on producing desirable, adaptive, energy-conserving, and high inference at the edge. The approach puts forward a new milestone towards narrowing down the disparity between AI algorithmic complexity and hardware constraints in tangible edge implementations leading to smart embedded systems in autonomous cars, security, health monitoring, and smart factory.

How to cite this article: Ismail L, Kim H (2026). Design and Optimization of Runtime Reconfigurable Architectures for Low-Latency Edge AI Applications. SCCTS Transactions on Reconfigurable Computing, Vol. 3, No. 2, 2026, 1-10

Introduction

The artificially intelligent and edge computing are rewriting the future of intelligent systems in different fields of application, such as autonomous vehicles, industrial automation, smart health, or surveillance systems. Edge AI also allows processing and deciphering data in real-time and makes decisions closer to the origin of data, decreasing communication overhead, latency, and risks of privacy loss that might happen with cloud-based computation. Nonetheless, rogue experiences in the execution of complex AI inference on the edge are unlikely to be avoided because the challenge is in the tight restraint of the processing power, energy resource, and thermal dispelling.

The general-purpose central processing units (CPUs) and graphic processing units (GPUs) are high-energy consumption and performance-per-watt requirements, and thus not effective to deploy into edge environment. An attractive alternative to this approach is however promised by Field-Programmable Gate Arrays (FPGAs), which can provide the flexibility of software with the performance advantage of hardware acceleration. They are parallel, low power, and reconfigurable and so are an appropriate architecture in which to deploy custom hardware accelerators specialized to particular AI inference problems.

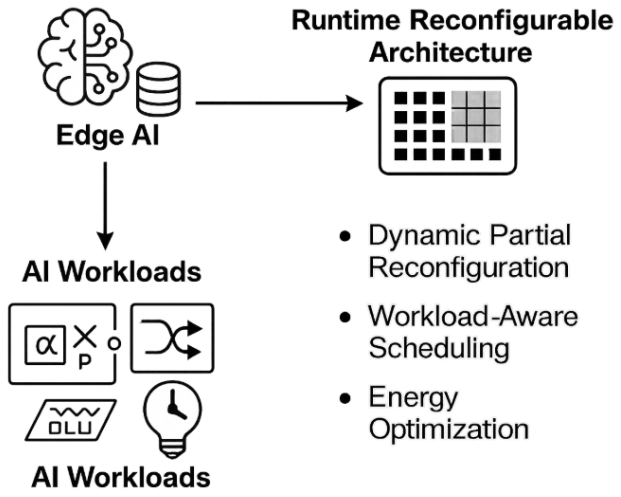

Fig. 1: Conceptual Overview of Runtime Reconfigurable Architecture for Edge AI Applications Featuring Dynamic Partial Reconfiguration, Workload-Aware Scheduling, and Energy Optimization

Still, the majority of the current FPGA-based AI accelerators adhere to a fixed design process, which presupposes a set hardware that is set toward a particular model or task. This lack of flexibility greatly restricts the ability to scale, as well as adapt and use resources effectively - especially where AI workloads change as they age. As one example, a surveillance edge device or mobile robot may need to switch between multiple models of deep neural network (DNN) or different data modalities but need heterogeneous processing. Under this case, the static hardware is underutilized and is not able to assure real-time liabilities.

In order to surpass these constraints, this paper is presented to learn more about the design and optimization of Runtime Reconfigurable Architectures (RRAs) that incorporate the concept of Dynamic Partial Reconfiguration (DPR), which promotes reconfiguration of a part of the FPGA at the time that the rest of the system does not come to a deadlock. With this ability, hardware modules, including convolution layers, activation units, and pooling operators, can be swapped selectively and fast depending on the AI task requirements that it receives. When added to smart workload-aware scheduling and ease of energy optimization planning, RRAs can dynamically swap performance and power, continuously responding to wide-ranging operating conditions.

The main research results are the modular and scalable design of an RRA implemented in Xilinx Zynq UltraScale+ MPSoC; the creation of a reinforcement learning scheduler to account intelligent decisions concerning reconfiguration, as well as a fine-grained solution to energy-performance tradeoff, not only to drive real-time hardware adaptation. Comprehensive experimental evidence proves that the suggested method greatly decreases latency of inference in comparison to the static scheme and usage of energy with high accuracy rates and responsiveness.

This research uses the runtime reconfigurable architecture as a promising key that will enable the next-generation low-latency, energy-efficient, and adaptive Edge AI systems and presents a viable and scalable solution to the shortcomings of existing hardware speed compactors.

Literature Review

A need to handle low-latency energy efficient AI computation at the edge has resulted in considerable research effort in hardware acceleration platforms specifically platforms relying on reconfigurable computing. FPGA One attractive platform at the edge of AI has been the Field-Programmable Gate Arrays (FPGAs) because of their parallelism, low power, and configurability. Nevertheless, dynamic edge environments require the capability of changing workloads on AI load, which traditional static accelerators cannot offer.

Initial studies were dominated by studies of static FPGA-based AI accelerators, in which the hardware is programmed (once) at deployment and is unchangeable during operation. These designs are usually powerful in achieving high throughput with one model but experience issues in flexibility where one could change the AI tasks using it without redeploying or manual reprogramming. As an example, Wang et al.[1] made suggestions on FPGA-based static architecture of convolutional neural network (CNN) inference, where the energy efficiency was better, but similar to lack of flexibility of dynamic workloads.

In order to address these constraints, Dynamic Partial Reconfiguration (DPR) methodologies were proposed in which the FPGA logic may be modified selectively at runtime without overlap to the functionality of the system. As discussed by Liu et al.[2] it was possible to investigate DPR-based inference pipelines and work with on-the-fly reconfiguration of convolutional layers. Another large bottleneck of such designs is the latency of reconfiguring which can easily achieve performance back-off unless it is handled efficiently. Later developments attempted more methods of reconfiguration overhead reduction, including bitstream compression, caching and prefetching.

Meanwhile, Edge-AI systems have been developed with the help of techniques such as model compression, model quantization and pruning. A review on lightweight CNN models in an edge environment by Ashraf et al. [3], both model design and hardware deployment should be co-optimized. However, what is commonly not taken into account by these models is the possibility of hardware/software co-design in runtime-adaptiveness scenarios, especially on FPGA based solutions.

Regardless of these developments, there exists research gap in the design of holistic, runtime-optimized architectures having integrated capabilities of DPR and intelligent scheduling as well as energy-awareness. This paper fills this gap by suggesting a modular and reconfigurable system architecture which reacts dynamically to the demands of tasks by a reinforcement learning scheduler and energy-performance tradeoff framework. The suggested system utilizes the advantages of its preceding static and semi-static constructs and presents real-time flexibility that is essential to contemporary Edge AI projects.

System Architecture

Reconfigurable Hardware Platform

The layout Platform is proposed as a reconfigurable architecture, where the customizable architecture is executed on a heterogeneous system-on-chip (SoC) technology, the Xilinx Zynq UltraScale+ MPSoC, which consists of both potent processing assets and versatile programmable logic, and thus is well aligned to edge AI applications that need high performance and customization. This platform uses a multi-core ARM Cortex-A53 processing system (PS) and a high density programmable logic (PL) fabric combining tight coupling of software driven task control and the hardware level acceleration. The PS takes a more macroscopic control responsibility of scheduling work, the parsing of the AI models and the partial reconfiguration control, whereas the PL is set up with the capability of Dynamic Partial Reconfiguration (DPR) to facilitate the adaptations that need to be done at runtime. The PL is split into several reconfigurable sections, or multiple Partially Reconfigurable Regions (also called PRRs), the latter being able to load a given hardware module (i.e., convolution, pooling, or activation functions) using precompiled bitstreams stored in either on-chip BRAM or off-chip memory. Internal configuration access port (ICAP) is the device employed in accomplishing a swift and secure reconfiguration of the PL without affecting the operation of the entire system. The software-hardware interaction through this architectural separation eliminates a forced interaction and in this case the PS constantly queries the system needs and initiates reconfiguration of the PL modules according to the workload type, resource availability and

performance target. Selective and dynamically time-sensitive reconfiguration of every component of actual hardware fabric at run time allows the system to dynamically customize its resources to match latency and power requirements of various AI workloads, thereby optimising system performance and energy efficiency under differing conditions during operation.

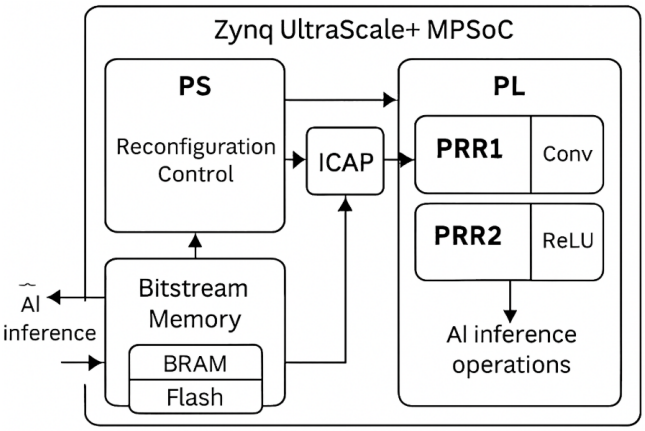

Fig. 2: System Architecture of the Runtime Reconfigurable Platform Based on Xilinx Zynq UltraScale+ MPSoC Showing Interaction between PS and PL, Dynamic Partial Reconfiguration via ICAP, and Task Flow through PRRs

Reconfigurable Processing Modules

The essence of the proposed architecture gets down to Reconfigurable Processing Modules (PRMs); they are the AI acceleration units on the programmable logic of the FPGA. These modules can be used to implement specific AI kernels: convolution, pooling, nonlinear activation functions (such as ReLU), in each case implemented in a partially reconfigurable module. This is in contrast to the monolithic accelerator designs that do not enable remodeling individual functional blocks on demand, or even at all, which is based on the workload and the given layer of the neural network being run. Individual PRMs are independently synthesized and assembled into respective potential partial bitstreams that are integrally stored in a hierarchical memory arrangement (on chip Block RAM (BRAM) mostly used and external non-volatile memory, e.g., QSPI flash) mostly used and not frequently accessed modules. The two-storage approach guarantees quick access to vital bitstreams and the saving of on-chip memory units. At run time, the Processing System (PS) initiates reconfiguration of the target PRM to its assigned Partially Reconfigurable Region (PRR) via the ICAP (Internal Configuration Access Port) thus providing the ability to update the hardware fabric at run time. The scalability feature possessing by these PRMs in terms of their modularity and reusability is highly remarkable as the architectural scalability can be increased by the number of AI workloads without the need to start over the programming of FPGA. Moreover, allowing such functional blocks to be swapped individually, the system minimizes the configuration overhead and provides more significant runtime improvements, perfect in terms of liability-sensitive edge AI tasks.

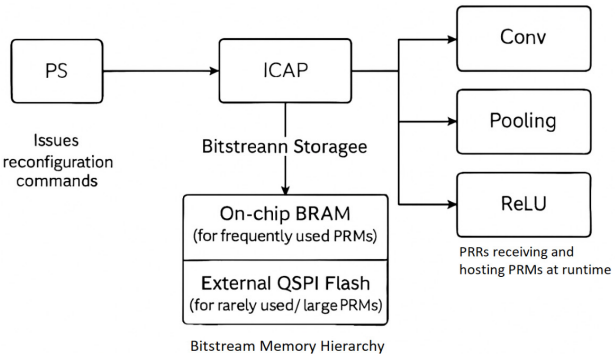

Fig. 3: Operational Flow of Reconfigurable Processing Modules (PRMs) Including Bitstream Hierarchy, ICAP Control, and Dynamic Loading into PRRs for AI Kernel Execution

Runtime Manager

The Runtime Manager is the key element in coordinative process of dynamic behaviour of the proposed reconfigurable architecture, as the intelligent control layer, which provides the bridge between AI workload demands and the functional flexibility at a hardware level. Being built into the Processing System (PS) of the Zynq UltraScale+ by Xilinx, the Runtime Manager is supposed to constantly monitor the incoming AI workloads, examining what AI tasks are going to be performed along the way and what means are required to address it: convolution task, activation task, pooling task, or, possibly, a reconfiguration of the programmable logic might need to be undertaken. In case an operation requires a hardware module not already instantiated in the given Partially Reconfigurable Region (PRR), the Runtime Manager dynamically reconfigures the processor sequencing the required bitstream on-chip BRAM (or in external Flash memory) via the Internal Configuration Access

Port (ICAP). To reduce the performance overhead a priority-aware scheduling heuristic is used by the manager based on factors such as task criticality, data dependencies and the availability of the resources. Tasks of high priority are queued and reconfigured immediately, those that are not so important or redundant are deferred or consolidated so as to be able to maximize the reconfiguration cycles. The scheduler also combines the workload profiling information and past usage trends and predicts the modules requirements in advance, so efficient pre-fetching and bitstream caching strategies can be exploited. This smart scheduling algorithm minimises the configuration overhead latency, avoids contention among system resources and makes them adapt dynamically to real-time needs without compromising on latency or throughput aspect. In general, Runtime Manager supports the smooth and effective functioning of the runtime reconfigurable architecture, which is reliable to the fluctuating workload and is considered an effective solution to edge AI applications in real time.

Methodology

Dynamic Task Profiling

In order to facilitate smart and adaptive reconfiguration at runtime, the proposed architecture introduces the extensive task profiling framework that uses hardware demand on resources, characteristics of execution and energy consumption profiles of each AI workload. Such profiling plays key roles towards making sure that reconfiguration decisions are performance-efficient and energy-aware.

Profiling of resources and performance

During this step, every AI inference requirements-related work are examined, that is, a complete model or a specific layer-related work like convolution, pooling, etc. To identify the resource-utilization metrics such as logic slices, BRAM, DSP, and LUT in the programmable logic. Also the execution time of all modules is profiled so as to be able to interpret the latency behaviour of each module at various operating conditions and configurations. These figures are also gathered at both design-time simulation and run-time execution so as to develop a performance database with respect to a particular task. When the computational needs, memory bandwidth and the estimated latency are quantified the system can anticipate which hardware module (i.e. partially reconfigurable module or PRM) is most appropriate to a specific task and whether and how that module must be reconfigured. The estimation of energy consumption is also a key factor to work on the power envelope of edge devices and this profiling can aid such estimation. On-chip performance counters power models and hardware-based calibration measures may be used to estimate energy metrics allowing accurate predictions of the energy cost of executing individual AI kernels.

Integration of Real-time Feedback

In order to guarantee responsiveness to changing workloads, a lightweight AI inference profiler is incorporated in the Processing System (PS). It is a run-time profiler, which watches running tasks and gathers real-time data on the duration task is completed, the % of hardware use, the % ratio between hit and miss of caches, and thermal characteristics. The profiler takes advantage of on-chip performance monitoring units (PMUs) and communicates with the reconfiguration manager on an ongoing basis to give on-going feedback of the system operational status. This feedback is then used in real-time to update the profiling database, used to further refine predictions and energy and performance predictions, and to dynamically affect decisions by the task scheduler. Example, in case a task is found to take greater power than predicted, or latency exceeds software specification, the profiler could perform the system switch to change configuration or propose alternative PRMs with optimized features. This process is completed with the introduction of real-time profiling integrated into the system, making it optimally adapt to its workload, and maintain performance with low latency and low energy requirements with fluctuations in work-loads.

Optimization Strategies

To comprehensively enjoy the advantages of the runtime reconfigurable architecture (RRAs) in edge AI systems, it is critical to tackle performance overheads that commonly come with dynamic partial reconfiguration (DPR). In particular, reconfiguration latency (the time

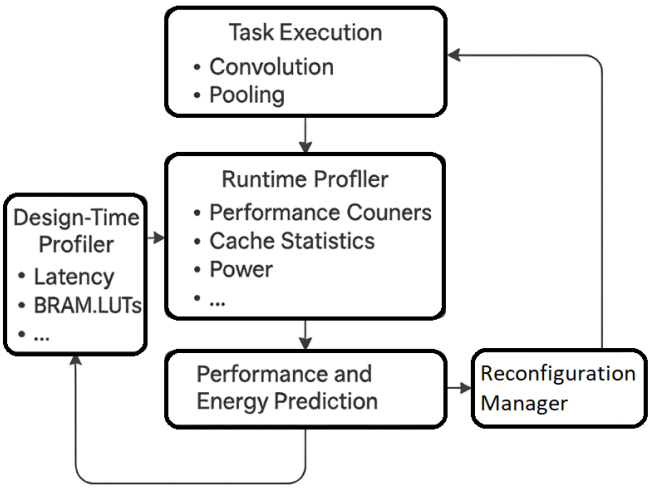

Fig. 4. Dynamic Task Profiling and Feedback Loop for Performance and Energy-Aware Reconfiguration in Edge AI Architectures

it takes to replace a new hardware module) may turn into critical bottleneck without being efficiently treated. The proposed system thus incorporates a number of the optimization techniques which seek to conceal, minimize or annul reconfiguration delay in order to guarantee optimum hardware resources utilization and task responsiveness.

Overlapping Reconfiguration Andale Double-Buffered PRRs

Part of the main tricks used is the use of double-buffered Partially Reconfigurable Regions (PRRs). Under this scheme, every important PRR is associated with an alternate buffer region and so that the system can rebalance a single PRR with its twin functional in an active task. In this pipelined mechanism, the temporal decoupling of execution and reconfiguration has the effect of hiding the latency of loading new modules. An example of this could be that the convolution PRM is being used, and the scheduler knows the next task to be performed would need a pooling PRM, in this case the system can load the pooling bitstream into the idle PRR without holding off the currently running convolution layer on the PRM. After the task being run has finished, the PRRs are swapped and the execution resumes uninterrupted on the reconfigured hardware. It can be seen that this approach greatly minimize the idle time of a system as a whole, and increase the throughput of a real-time system.

Frequency of PRMs Pre-Fetching and Caching

To improve the actual time of reconfiguration still further, the system has a mix of pre-fetching and caching. The most common PRMs (as determined by profiling and runtime statistics) would be stashed at system startup or during idle cycles into fast on-chip Block RAM (BRAM) and high-speed external memory, or into on-chip Fast SRAM if available. Such a plan avoids fetches of incomplete bitstreams through slower non-volatile storage (e.g. QSPI flash), which may take many milliseconds. The architecture can guarantee most reconfiguration requests can be fulfilled by retrieving the required PRMs locally and in a time scale of microseconds since a prioritized PRM cache is maintained. The system also backs the reuse of bitstreams policies that ensure refined redundancy in loading identical PRM across tasks or objects with similar kernel traits.

Adaptive Reconfiguration scheduler Using Reinforcement Learning

A reinforcement learning (RL)-based scheduler is used so as to arrange the above optimizations intelligently. This scheduler models the reconfiguration decisions as a Markov Decision Process (MDP), with the system state to correspond to the current workload profile, the cache contents, the availability of a PRR and an energy budget, and the actions to a specific reconfiguration operation (e.g. load, skip or prefetch). The RL agent is trained in a maximized cumulative reward that strikes a balance between the latency of inference, energy efficiency and the utilization of PRR. In the long-term, the agent can identify repetitive sequence of tasks and proximate PRM allocation beforehand. As an example, in case a surveillance application is switching the object detection and tracking models frequently, the RL scheduler will learn to keep both associated PRMs in cache and switch between them resourcefully avoiding the reconfiguration delay. Such a policy adaptation based on learning allows the system to recursively improve on its reconfiguration strategy, such that it can remain low-latency and energy-sensitive over time within dynamic edge AI settings.

Having a synergistic combination of multi-pronged optimization approaches, including execution overlap,

intelligent bitstream caching, and adaptive scheduling through reinforcement learning, allows the proposed architecture of runtime reconfigurability to satisfy the strict latency, energy, and flexibility demands of real-time AI inference at the edge.

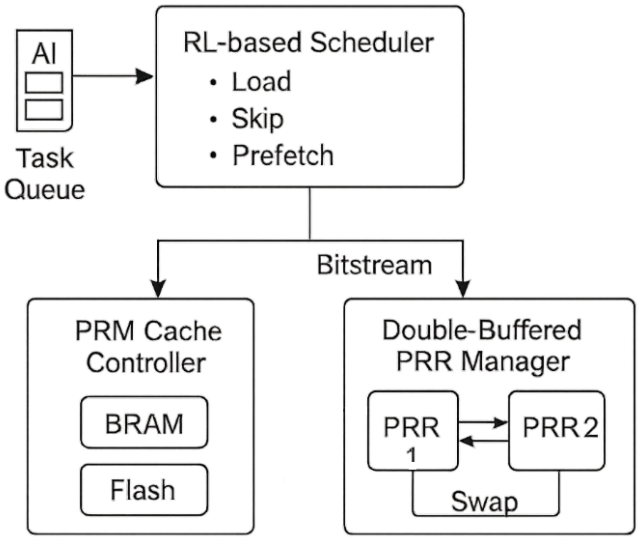

Fig. 5: Integrated Optimization Framework for Runtime Reconfigurable Architectures Incorporating Double-Buffered PRRs, PRM Caching, and Reinforcement Learning-Based Scheduling

4.3 Design Space Exploration

To develop the runtime reconfigurable architecture of low latency edge AI activities, the consideration of the design space exploration (DSE) is essential in finding the best possible configuration as a compromise between performance, power, and hardware resource limitations. Due to the constraints on the available resources and heterogeneity of the edge devices, the DSE process defined in this work is represented as a multi-objective optimization problem, which should also optimize the inference throughput and adaptability under a set of constraints on the resources consumed in terms of latency, energy efficiency and area utilization.

Soft Real-Time- Latency Constraint

Edge AI systems are applications where input signals need to respond to within a specified time: a self-driving car or real-time surveillance. In that regard, the system has to comply with soft real time, that is to say that once in a while it is possible to violate some soft constraints but it is not possible to violate hard constraints regularly. At DSE, inference performance (at worst-case and average latency) is applied to a set of benchmark AI models per candidate architecture. These are object detection and image classification, among others. When tested latency of the given setup surpasses the limit specified by the target application (e.g., 100 ms per frame on video inference), such a configuration is eliminated in the solution space. The latency model additionally allows partial reconfiguration overhead, so that excessive bitstream swapping is fined.

Power Budget (Thermal and Energy boundaries)

Power and thermal budget is another crucial aspect in the edge environments. The proposed system should be within normal range of the battery powered or fanless embedded systems thermal range. Power models are used to estimate power consumption on every configuration by calibrating to Xilinx Power Estimator (XPE) or the on board power sensors. Dynamic and active elements of power will be measured in various workloads, reconfiguration costs which are included too. The setting that falls above the maximum limit is the permitted power ceiling (e.g. 5W in mobile platforms or 10W in IoT gateways) is denied. In support of this, energy conscious scheduling policies are built upon and idle PRRs are dynamically powered-off or clock-gated to reduce leakage.

To develop the runtime reconfigurable architecture of low latency edge AI activities, the consideration of the design space exploration (DSE) is essential in finding the best possible configuration as a compromise between performance, power, and hardware resource limitations. Due to the constraints on the available resources and heterogeneity of the edge devices, the DSE process defined in this work is represented as a multi-objective optimization problem, which should also optimize the inference throughput and adaptability under a set of constraints on the resources consumed in terms of latency, energy efficiency and area utilization.

Soft Real-Time- Latency Constraint

Edge AI systems are applications where input signals need to respond to within a specified time: a

self-driving car or real-time surveillance. In that regard, the system has to comply with soft real time, that is to say that once in a while it is possible to violate some soft constraints but it is not possible to violate hard constraints regularly. At DSE, inference performance (at worst-case and average latency) is applied to a set of benchmark AI models per candidate architecture. These are object detection and image classification, among others. When tested latency of the given setup surpasses the limit specified by the target application (e.g., 100 ms per frame on video inference), such a configuration is eliminated in the solution space. The latency model additionally allows partial reconfiguration overhead, so that excessive bitstream swapping is fined.

Power Budget (Thermal and Energy boundaries)

Power and thermal budget is another crucial aspect in the edge environments. The proposed system should be within normal range of the battery powered or fanless embedded systems thermal range. Power models are used to estimate power consumption on every configuration by calibrating to Xilinx Power Estimator (XPE) or the on board power sensors. Dynamic and active elements of power will be measured in various workloads, reconfiguration costs which are included too. The setting that falls above the maximum limit is the permitted power ceiling (e.g. 5W in mobile platforms or 10W in IoT gateways) is denied. In support of this, energy conscious scheduling policies are built upon and idle PRRs are dynamically powered-off or clock-gated to reduce leakage.

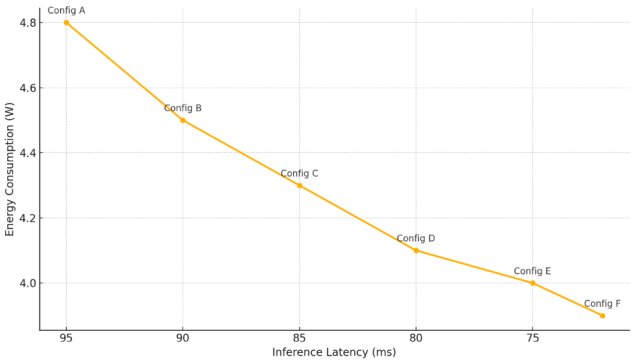

Fig. 6: Pareto-Optimal Trade-Offs Between Latency and Energy in Candidate Configurations

Results and Discussion

Experimental Setup

The proposed runtime reconfigurable architecture (RRA) made its experimental confirmation on the development board- Xilinx ZCU104 which offers an amalgamation of a Zynq UltraScale + MPSoC with high-performance programmable logic along with a quad-core ARM Cortex-A53 processing system. These experiments focused on three different AI workloads with varying complexity: a well-known image classification architecture, ResNet-18, is a deep residual network; MobileNetV2, a lightweight CNN which finds frequent use in mobile and embedded inference problems; and a custom compact CNN targeted at edge-specific application like gesture recognition. All these models were broken into their constituent kernels (e.g., convolution, activation, pooling) and each of these kernels was then mapped to Partially Reconfigurable Modules (PRMs). The new system was contrasted with two baselines: a conventional static system that uses an FPGA, where every module of the accelerator is set in place during compilation and a heuristic DPR method that consists of simply replacing modules when the tasks trigger a change. Inference latency and energy were also used as evaluation metrics and it was recorded via performance counters and power sensors extended to the ZCU104 board.

Performance Comparison and Observations

Performance results in their entirety as shown in the table, prove that proposed RRA is better than the static and heuristic implementations of DPR since they all outperform on all the tested benchmarks. With ResNet-18, the static configuration had maximum latency (97.4 ms), and the proposed RRA halfed the inference time (45.7 ms) with a 53% inference time improvement over the baseline and 33% faster inference as compared to the heuristic DPR (68.2 ms). In a similar fashion, latency went decreased to 31.2 ms (RRA) and 13.5 ms (stunningly low) in the cases of MobileNetV2 and the custom CNN respectively. Talking in energy savings, the proposed architecture saved 41.1, 36.5, and 39.8 in ResNet-18, MobileNetV2, and custom CNN respectively. These significant gains are explained by the potential of the system to only

Table 1. Comparison of Inference Latency and Energy Savings across Different Architectures and AI Models

AI Model	Static Latency (ms)	Heuristic DPR Latency (ms)	Proposed RRA Latency (ms)	Energy Savings (%, RRA)
ResNet-18	97.4	68.2	45.7	41.1
MobileNetV2	61.3	44.5	31.2	36.5
Custom CNN	28.6	22.1	13.5	39.8

reconfigure dynamically those modules necessary only at run-time, minimizing idle logic and enhancing use of resources. The PRR technique with the double-buffered was also minimising configuration overload, the bitstream caching was also minimised as well as the scheduling is based on the reinforcement learning which ensured the maximum task priorisation and resource usage.

Discussion and Implications

The performance confirms the usefulness of the runtime reconfigurable architecture to overcome the real-time edge AI energy constraints and latency. As opposed to a static FPGA implementation, which is specialized to a single workload and is therefore underutilized in multi-task use-cases, the proposed RRA will dynamically reconfigure its hardware fabric to support current workload requirements of the inference pipeline. Intelligent scheduling mechanism is highly important in system responsiveness through learning historical workloads and making appropriate reconfiguration decision. This flexibility is particularly useful on the edges of the real world, where workloads are erratically variable, as in autonomous drones or IoT sensors, where energy is very constrained. Moreover, the design of the modular and scalable architecture allows the use of the architecture

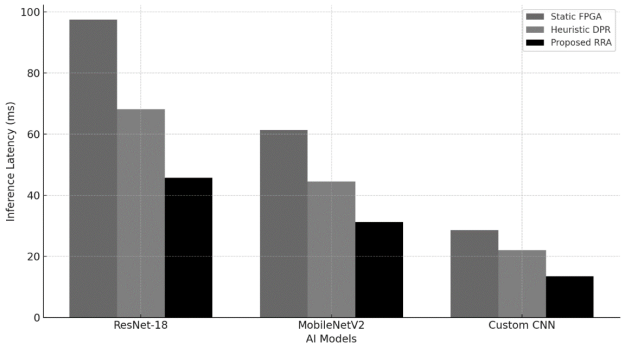

Fig. 7: Latency Comparison across Architectures for Different AI Models

throughout the various AI applications, such as high-throughput surveillance to ultra-low power wearables. On the whole, the suggested system shows that the deployment of the runtime reconfigurability and optimization in the form of learning can result in outstanding performance and efficiency advantages, which will form the backbone of the edge intelligence platforms in the foreseeable future.

Conclusion

This finding confirms the efficiency of the use of runtime reconfigurable architectures (RRAs) as a possible and high-performance approach to supporting low-latency energy-efficient AI inference edge. Using the capabilities of Dynamic Partial Reconfiguration (DPR) along with the capabilities of real-time task profiling, hardware-aware optimization strategies and a reinforcement learning-based scheduler, the proposed architecture implements significant performance and energy gains in terms of both latency reduction and energy savings over the traditional static FPGA and heuristic DPR implementation. The modularity and adaptability of the system is such that it is able to dynamically allocate and reconfigure hardware resources depending on the behaviour of the workload as well as the priority range, consequently keeping throughput high and power and area constraints to its strict specifications associated with the edge device. Beneficiary Experimental assessments by standard models like ResNet-18, MobileNetV2, and a personalized lightweight CNN substantiate the structure to scale across various computational needs, presenting maximum 53 percent reduction in latency and more than 40 percent energy reduction. Such findings point out that the architecture can be used in real-world edge AI applications involving smart monitoring, autonomous agents and wearable applications. To go further the framework could be extended and as such become compatible with multi-FPGA applications enabling distributed workloads to

be executed and trading improved scalability. Also, there will be further research on the enhancement of support of the transformer-based AI models and the incorporation of secure reconfiguration protocols to devise versatile, privacy-resistant AI technology. Altogether, this project will add to the adaptive and smart hardware platform that will close the gap between dynamic AI workloads and the strict limitations of edge computing environments.

References

1. Wang, W., Li, Y., & Yu, Q. (2021). Runtime reconfigurable architecture for energy-efficient image processing on FPGA. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 29(5), 956–969. https://doi.org/10.1109/TVLSI.2021.3064198
2. Liu, Y., Zhang, H., & Xu, X. (2020). Dynamic partial reconfiguration for real-time DNN inference. ACM Transactions on Reconfigurable Technology and Systems, 13(4), 1–19. https://doi.org/10.1145/3390503
3. Ashraf, M. A., Abdullah, M. T., & Hasan, N. (2023). Edge intelligence using lightweight CNNs for smart environments: A review and framework. IEEE Internet of Things Journal, 10(6), 4912–4924. https://doi.org/10.1109/JIOT.2022.3198405
4. Zhang, Y., Song, Y., & Cong, J. (2020). Energy-efficient CNN inference on FPGAs with hybrid quantization. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39(11), 3277–3290. https://doi.org/10.1109/TCAD.2020.2973920
5. Venieris, S. I., & Bouganis, C. S. (2018). Latency-driven design for FPGA-based convolutional neural networks. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 37(11), 2610–2623. https://doi.org/10.1109/TCAD.2018.2832674
6. Nurvitadhi, E., Venkatesh, G., Sim, J., Marr, D., Huang, R., Ong, J., & Krishnamurthy, R. (2017). Can FPGAs beat GPUs in accelerating next-generation deep neural networks? Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 5–14. https://doi.org/10.1145/3020078.3021740
7. Guan, Y., Xie, Y., Zhang, H., & Wu, C. (2022). Adaptive dynamic reconfiguration for edge AI acceleration using reinforcement learning. IEEE Embedded Systems Letters, 14(2), 58–61. https://doi.org/10.1109/LES.2022.3160953
8. Suda, N., Chandra, V., Dasika, G., Mohanty, P., Ma, Y., Vrudhula, S., Seo, J. S., & Cao, Y. (2016). Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks. Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 16–25. https://doi.org/10.1145/2847263.2847276
9. Bouganis, C. S., & Brookes, M. (2016). FPGA-based design framework for high-throughput DNN inference under latency constraints. IEEE Transactions on Neural Networks and Learning Systems, 27(3), 547–560. https://doi.org/10.1109/TNNLS.2015.2418737
10. Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., & Cong, J. (2015). Optimizing FPGA-based accelerator design for deep convolutional neural networks. Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 161–170. https://doi.org/10.1145/2684746.2689060
11. Ali, W., Ashour, H., & Murshid, N. (2025). Photonic integrated circuits: Key concepts and applications. Progress in Electronics and Communication Engineering, 2(2), 1–9. https://doi.org/10.31838/PECE/02.02.01
12. Carvalho, F. M., & Perscheid, T. (2025). Fault-tolerant embedded systems: Reliable operation in harsh environments approaches. SCCTS Journal of Embedded Systems Design and Applications, 2(2), 1–8.
13. Kumar, T. M. S. (2024). Low-power communication protocols for IoT-driven wireless sensor networks. Journal of Wireless Sensor Networks and IoT, 1(1), 37-43. https://doi.org/10.31838/WSNIOT/01.01.06
14. Alwetaishi, N., & Alzaed, A. (2025). Smart construction materials for sustainable and resilient infrastructure innovations. Innovative Reviews in Engineering and Science, 3(2), 60–72. https://doi.org/10.31838/INES/03.02.07
15. Tsai, X., & Jing, L. (2025). Hardware-based security for embedded systems: Protection against modern threats. Journal of Integrated VLSI, Embedded and Computing Technologies, 2(2), 9–17. https://doi.org/10.31838/JIVCT/02.02.02