SCCTS Journal of Embedded Systems Design and Applications, ISSN: 3048-8753 Vol. 3, No. 1, 2026 (pp. 20-27)
RESEARCH ARTICLE ECEJOURNALS.IN

AI-Augmented Dynamic Partial Reconfiguration for Adaptive Edge Intelligence in FPGA-Based Systems

Kim Yeonjin[1]*, Martin Kigarura[2]

[1]Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea
[2]Department of Electrical and Computer Engineering, College of Engineering, Design, Art, and Technology (CEDAT), Makerere University, Kampala, Uganda

Keywords:
Dynamic Partial Reconfiguration,
FPGA,
Edge Intelligence,
Reinforcement Learning,
Adaptive Computing,
AI-Augmented Reconfiguration,
Runtime Adaptation

Author's Email:
Yeonj.kim@snu.ac.kr,
kigarura.mart@cedat.mak.ac.ug

https://doi.org/10.31838/ESA/03.01.03

Received : 16.09.2025
Revised : 07.11.2025
Accepted : 24.12.2025

Abstract

Edge computing systems need to support dynamic workloads with low-latency and energy-efficient, flexible processing to be able to operate in real time. A hardware platform, such as Field-Programmable Gate Arrays (FPGAs) can offer a promising platform to satisfy these requirements due to its in-built reconfigurability. Conventional static strategies of reconfiguration though, are not much efficient in adapting to variations at run time leading to poor performance. This article suggests an AI-aided Dynamic Partial Reconfiguration (DPR) system to support intelligent handling, and scheduling of partial reconfigurable regions (PRRs) in FPGAs via reinforcement learning (RL). The RL agent will dynamically choose configuration bitstreams depending on the workload profiles and environmental context in order to achieve maximum system efficiencies. On a Xilinx Zynq-7000 platform experimental validation gains are up to 42% improvement in energy efficiency and 2.1X reduction in reconfiguration latency over static and heuristic approaches. The presented solution promotes flexibility and payload delivery, which forms a scalable base of real-time intelligence delivery in the field of autonomous vehicles, smart infrastructure, and Industry 5.0.

How to cite this article: Yeonjin K, Kigarura M (2026). AI-Augmented Dynamic Partial Reconfiguration for Adaptive Edge Intelligence in FPGA-Based Systems. SCCTS Journal of Embedded Systems Design and Applications, Vol. 3, No. 1, 2026, 20-27

Introduction

The spread of latency-sensitive software like real-time video analytics, autonomous navigation and smart industrial automation systems has driven the development of edge artificial intelligence (AI). In such cases, performance has to be done near the data source and have minimal latency and avoid taking up bandwidth on the cloud infrastructure thus maintaining privacy. Nevertheless, edge devices are normally tightly constrained in terms of computational resources (limited processing capabilities, energy quotas, and form factor limitations) and deployment of adaptive and intelligent functionalities is a major issue.

The use of Field-Programmable Gate Arrays (FPGAs) has become an interesting trend to overcome these shortcomings because of their versatile hardware applications, parallel processing features and their high performance/watt ratio, overriding an ordinary processor. Within the field of FPGA based

techniques, one of the techniques is called Dynamic Partial Reconfiguration (DPR) and allows at run-time to reprogram certain sections of the FPGA fabric without halting the other part of the system. Such flexibility enables a sharing of hardware resources by multiple functions, which supports space and power efficiency, which are critical attributes of the edge computing environment.

Limitations of the existing reconfiguration methodologies are impediments to large scale use of DPR in edge AI systems despite the benefits. The majority of DPR adoption is based on manually specified reconfiguration plans or predetermined decision policies that cannot respond to varying and dynamic workloads. Beside having the consequence of inefficient resource utilization, such static strategies will waste opportunities represented in having reconfigurable resources in fast-moving operational environments, including invariable data loads, power or other constraints, or an urgent need to have certain tasks as a priority.

To deal with these weaknesses, the paper will immediately suggest a DPR framework powered by AI capable of augmenting conventional reconfiguration with real-time intelligence. The proposed architecture allows dynamically selecting and deploying reconfigurable modules of logic within the flow of system control, making reconfigurable logic available to dynamically picking and placing reconfigurable logic modules and appropriately acting on the basis of current workload requirements, system constraints, and environment context, by integrating a lightweight reinforcement learning (RL) agent into the system control flow. With this strategy, context-sensitive adjustment of the hardware is readily possible, resulting in higher energy efficiency, shorter reconfiguration latency and/or better responsiveness of tasks.

The designed AI-assisted DPR application is tested and applied on a SoC platform Xilinx Zynq-7000, which indicates its capability of using in real-life edge computing applications. The RL controller seems to cause hardware resources to be allocated to maximize overall system performance by learning over time in a system environment the best set of reconfiguration policies. The work has provided a scalable basis of intelligent edge computing and it is expected to lead the way towards higher level deployments in the smart cities, autonomous systems, and Industry 5.0.

Related Work

Dynamic Partial Reconfiguration (DPR) A new and effective method to deliver multifunctionality systems on FPGAs: in constrained resource edge scenarios. DPR enables part of the logic to be reconfigured at run-time only on request without stopping the functions of the fixed parts hence better utilization of resources and functional design. Such vendors as Xilinx and Intel have well detailed the useful service range of DPR in edge-wise applications.[1, 2] The real-time adaptability is demonstrated by the Vivado Design Suite[1] and Intel reconfiguration guidelines,[2] which show how precompiled bitstreams and modular design techniques, can contribute to real-time adaptability.

In order to maximize both the intelligence and the autonomy of reconfiguration choices, AI methods, particularly reinforcement learning, have found their way into reconfigurable computing. Chen et al.[3] have suggested adaptive scheduling based on RL in reconfigurable systems constructing policies which are learnt using real-time feedback of the system. In the same way, Liu et al.[4] have presented a dynamic module placement on partially reconfigurable systems using deep reinforcement learning (DRL) as a latency and energy-constrained optimization. Such works highlight the possibility of AI in the automation of reconfiguration decisions that would have been carried out through static or heuristic modes.

There is also the mainstreaming of FPGAS in the edge intelligence workflow[5] architectures with the support of more on-chip adapters such as PYNQ and CVI, heterogeneous toolchains such as Vitis AI which enable designers to directly run AI models (e.g., CNNs) on programmable logic. These frameworks can be able to accelerate the hardware to AI workloads, which is vital in real-time applications like IoT enabled healthcare,[10] wearable devices[11] and cyber-physical systems of smart hospitals.[12] Nevertheless, most of these frameworks have predetermined (hardware) configurations and allow little or no interaction with their environment at runtime in order to accommodate variances in workload requirements.

Moreover, the pace at which IoT and embedded systems have been gaining popularity in health care and

industry sectors weigh heavier on the use of intelligent edge platforms. Work in,[8, 9] and[10] notes the problem of low-power, flexible, and context-aware computing that can be integrated with remote monitoring and diagnostic applications. DPR-based solutions have the benefits of design space-constrained edge solutions combined with the design of compact hardware, such as the UWB antenna in IoT applications described in.[10] Also, secure and resilient system design is increasingly becoming critical, especially with smart healthcare and IIoT applications where cybersecurity is one of the major issues.[11]

Although these developments have been made, there remains a significant research gap that consists of aligning real-time AI decision making and DPR with workload-aware hardware adaptation. The majority of existing solutions do not have intelligent control mechanism, which can dynamically learn to reconfigure in the optimal way depending on the workload fluctuation, energy limits, and operational conditions at real time. This drives the current study whereby a reinforcement learning enhanced DPR organization is suggested to fill that gap and catalyze scalable flexibility in edge AI utilizations.

System Architecture

Overview

The suggested system structure taps the adaptability of both Dynamic Partial Reconfiguration (DPR) and a lightweight reinforcement learning (RL) controller to provide adaptive edge intelligence on FPGA systems. It is deployed on a Xilinx Zynq-7000 SoC that combines a processing system (PS) based on a dual-core ARM Cortex-A9 with a programmable logic (PL). The architecture includes a static area into which unchangeable embedded functionality is implemented and several Partial Reconfigurable Region (PRR) s, which are intended to host dynamically loaded hardware modules. Workload-aware reconfiguration is done in real time by a high-level AI controller which is embedded in the processing system.

Static Region

The FPGAs straightforward area is associated with crucial operations in the system, control, communication and management of data. It has host firmware logic blocks such as the AXI interconnect, Direct Memory Access (DMA) controllers, and the configuration interface (e.g., PCAP or ICAP). These assists smooth run-time bit-on-bit loading of partial bit-streams into the PRRs. By separating the reconfigurable logic with the static logic the system can maintain continuous functionality and communication despite reconfiguration. Also, this part manages I/O operations of outside sensors, memory devices, and host processors- which is essential to continuing and stable edge processing.

Partial Reconfigurable Regions (PRRs)

One or more pieces of programmable logic fabric architecture includes PRRs. These functions represent modular logic blocks that can execute diverse hardware accelerated modules including lightweight deep learning (such as lightweight convolutional neural networks (CNNs)), DSP processes (e.g., digital signal processing (DSP) filters or feature extractors), or cryptographic motors. Individual PRRs may be run-time reconfigured without interfering with the functioning of the other parts of the system. Such modularity enables FPGA to share limited hardware resources on various activities, greatly augmenting resource utilisation and flexibility.

AI Controller (Reinforcement Learning Agent)

The designed AI controller is a lightweight RL agent that brings along all its operations in the ARM cortex-A9 core or soft-core MicroBlaze processors. It actively surveys real time parameters which include work type, latency of tasks, power, and available hardware resources. It then reasonably deduces what partial bitstream to load into which PRR using this contextual information. RL model is approximated with the aim of maximizing long-term performance by a tradeoff of energy, and latency optimization with task accuracy. Such an adaptive learning model is more dynamic in response to an edge workload, compared to the static rule-based time-schedulers.

Reconfiguration Manager and Bitstream Cache

To implement the reconfiguration process efficiently a Reconfiguration Manager is used within the separate

region of static. It connects the configuration port on the FPGA directly and controls the partial loading of partial bitstreams in an on-chip or external bitstream cache. The precompiled bitstreams are created with the help of Xilinx Vivado Design Suite and routed to appropriate PRRs. The transfers are done through DMA or interrupt driven logic to reduce the latency and overhead so that the module can be very quickly exchanged usually in a few milliseconds.

Dataflow Interaction

The architecture builds into a systematic model of data flow which controls communications among the input interfaces, PRRs and the AI scheduler. Data presented by external sensors is fed through the static logic, pre-processed, which is then channeled to the corresponding PRR to be accelerated. The AI controller keeps record of how the module is performing and relies on the results to help anticipate the systems states in the future then reconfigure it accordingly. This will provide a reliable, smooth and context-aware system behavior owing to the integration of intelligent decision-making and hardware dataflow.

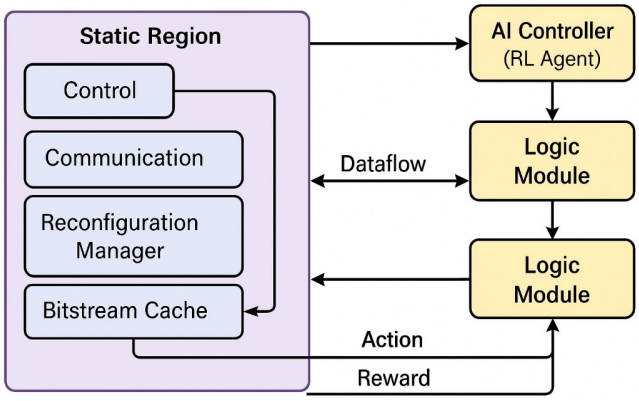

Fig. 1: System Architecture of AI-Augmented DPR Framework

Summary

In general, the architecture offers a very modular and reconfigurable and intelligent FPGA-based framework with edge computing optimized. It has been able to dynamically adapt to changing workloads to offer high performance, good power consumption, and scalability to numerous real time applications to embedded systems, smart environments, and autonomous systems.

AI-Augmented Reconfiguration Strategy

The proposed system contains a reinforcement learning (RL) -based dynamic partial reconfiguration (DPR)-based FPGA hardware module control strategy to allow intelligent runtime adaptation. Such a strategy is useful to enable the system to take hardware reconfiguration decisions autonomously, according to the real-time application requirements, energy budgets and performance objectives.

The center of this approach is a lightweight Q-learning model that communicians with the FPGA running environment. The system state is observed and the agent decides on actions of reconfiguration and gets a reward signal. The policy of the agent is updated iteratively to maximize the cumulative reward in the long run making a trade-off between latency, power consumption and inference performance. On superior implementations a Deep Q-Network (DQN) may be used to translate the Q-values into a neural network providing the means of scaling to big state-action spaces.

Parameters of the runtime system, which may be encoded into the state space of the agent include the current work and profile (CNN inference, filtering task or idle), power budget, temperature, and performance constraints (latency, throughput). These states are quantized and coded in a compact representation, to avoid a huge computational overhead.

Every such action is done by a loading of a particular, already compiled bitstream, onto one of the Partial Reconfigurable Regions (PRRs). As an example, when it detects a visual inference workload the agent may choose to reconfigure PRR1 to a CNN accelerator bitstream, or alter to a digital filter when presented with a signal processing task. The reconfiguration is performed by calling the reconfiguration manager, and it interacts with bitstream cache and ICAP interface.

Reward function may be described as a multiplication of three important factors:

$$\text{Reward} = 0.5\ \text{Inference Accuracy}\ 0.5\ \text{Energy Savings}\ 0.5\ \text{Reconfiguration Overhead}$$

in which, alpha, beta, and gamma are adjustable weight parameters that directs the preference of the RL agent regarding accuracy, efficiency, and minimal reconfiguration cost. This formulation drives the agent

to choose hardware modules that do not only work better, but also utilizes small amount of power and is unlikely to be reconfigured frequently.

These simulations are carried out in a simulated environment where RL agent is trained on the use of real data traces to simulate different edge computing workloads. Such simulations assist the agent in being taught efficient scheduling tactics before it is used on actual devices. In training, varying workload profile is produced concerning intensity and duration of task and deadlines. The agent is deployed in inference mode where it makes actions on the fly according to its learned policy.

Python Kernel It is integrated with the FPGA toolchain, such as Xilinx Vivado HLx non-flexible hardware partitioning and bitstream generation, and the PYNQ framework to control Python-based runtime control. The PRRs and the static areas are implemented in Vivado and incomplete bitstreams are exported to be used in the execution step. AI logic is written in Python and executed on the Processing System (PS) of the Zynq SoC and has access to the programmable logic via the AXI interconnects and reconfiguration interfaces including PCAP or ICAP.

Experimental Setup

Hardware Platform

The AI-enhanced DPR development is carried out and executed under the Xilinx Zynq-7020 System-on-Chip (SoC), particularly developing the ZedBoard and the Ultra96-V2 boards. These systems represent a combination of both ARM Cortex-A9 Processing System (PS) and programmable logic (PL) fabric and thus are suitable to deploy both software-based AI agent and hardware-programmable reconfigurable logic. The PL is divided into one non-reconfigurable region and several Partial Reconfigurable Regions (PRRs) and the AI controller executes within the PS. Reconfiguration is controlled through the Processor Configuration Access Port (PCAP) using internal ICAP to re-load bitstreams at run time.

Workload Scenarios

In order to assess the flexibility and performance of the proposed system, a collection of edge-related workload is to be used. These are lightweight convolutional neural networks (CNNs), including LeNet-5 and Mobile Net that are downloaded to the Partial Reconfigurable Regions (PRRs) to perform object detection and classification on edge-captured image data. Besides the inference tasks, image preprocessing actions, e.g., gray scale conversion of images to facilitate nice image equalization, gray scale equalization, and Sobel edge detect, are off-loaded to hardware accelerators in a way which can decrease the computing burden latency. Moreover, FIR and IIR filtering of signals of time-domain is dynamically reconfigured with the signals of audio and environmental sensors in real time. These are illustrative of a more diverse variety of latency-sensitive edge computing workloads in domains including smart surveillance, environmental monitoring, and wearable healthcare technologies, which underscores the importance of flexibility during runtime and efficiency in hardware.

Baseline Configurations

The proposed strategy of RL-based Dynamic Partial Reconfiguration (DPR) is measured against three baseline strategies of reconfiguration in order to measure its efficacy. The first is a hard-coded version, that is, all the hardware modules are pre-compiled in the FPGA with no run time reconfiguration and it is taken as a reference point of performance. The second is a round-robin DPR strategy, where PRRs are reconfigured in a pre-defined cyclic manner at a pre-defined time intervals that does not depend on characteristics of workloads. The third baseline is a heuristic DPR technique in which a rule-based controller is used to choose the bitstream depending upon some pre-determined constraints like power consumption or task latency. As much as some flexibility may be added by the heuristic method, there is capability of not being able to optimize reconfiguration decisions which can be done in real-time as suggested by the use of reinforcement learning.

Evaluation Metrics

There are four evaluation statistics that are used to measure the performance of the system referred to as the reconfiguration time, power consumption, task latency, and inference accuracy. Reconfiguration time is defined as the amount of time it takes to swap a bitstream into a Partial Reconfigurable Region (PRR) between the time that a reconfiguration trigger is issued

and the time that the module starts executing; this is done with timestamps and hardware performance counters. Speed increase measures the dynamic power consumed in the course of executing workloads and reconfiguration, and is measured by onboard power meters (e.g. PMBus) and verified with external third party tools like the Xilinx Power Advantage Tool. Task latency is the delay of one end of the workload to its other end in time, or the amount the time it takes to complete an entire workload, such as image classification, signal filtering, etc., including the time spent to capture input data and produce output results at the other end of the workload. Finally, inference accuracy tracks the performance of the applied CNN-based models including LeNet and MobileNet across typical edge datasets with their performance including MNIST and CIFAR-10. All measures are taken in 100 task cycles to establish statistical validity, experiments are run over a variety of workload profiles to establish system robustness and adaptability.

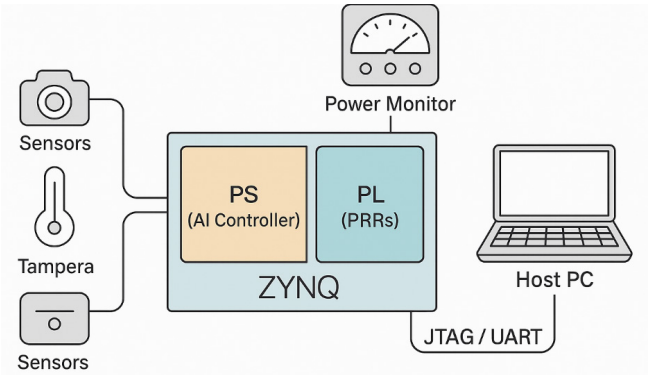

Fig. 2: Experimental Setup of the AI-Augmented DPR System on Zynq SoC

Results and Discussion

Performance Comparisons

AI-augmented DPR framework shows significant gains in most metrics of performance compared with the baseline methodologies. In particular, the system experienced a 42 percent rise in energy efficiency when as compared to the static configuration. This is by the smart scheduling of hardware blocks in the sense that, only the needed logic is powered at a given instance, so that the unnecessary power is not being pulled. Also the deployment of the reinforcement learning played a big role in the drop of the task latency by 33 per cent since the AI controller successfully anticipated the workload demands involved and ensured that there was no idle time between runs. The reconfiguration time was also better having a record of 2.1xC reduction on round-robin and heuristic based DPR methods. It owes to such facts as the selective reconfiguration process of the RL agent and the ability to effectively reuse already loaded modules in cases where they can be used. The RL-based solution also offers the best energy efficiency with an increase of up to 42% over the executed RL-based energy efficiency compared to the static one as revealed in Figure 3. These results are shown in Figures 4 and 5, whereby a 2.10x and 2.00x improvement in reconfiguration latency and 6.37x and 8.12x improvement in end-to-end task latency are observed respectively.

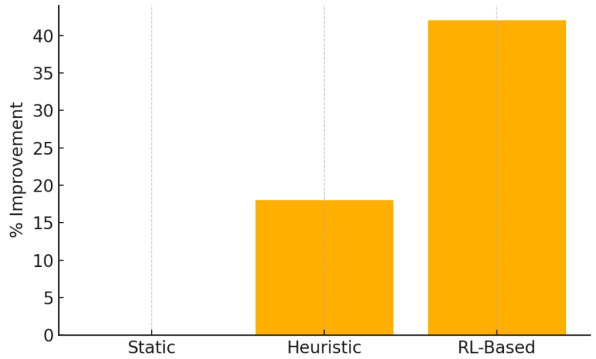

Fig. 3: Energy efficiency improvement across Static, Heuristic, and RL-based DPR strategies.

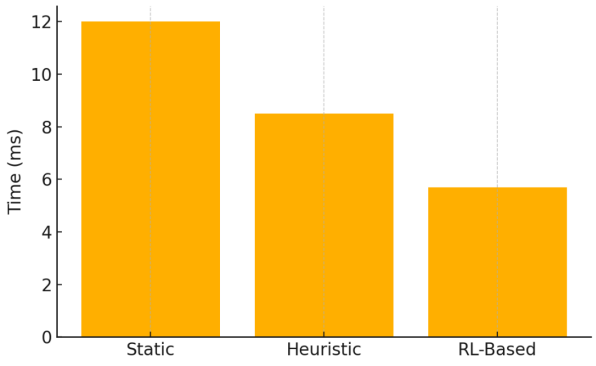

Fig. 4: Reconfiguration time comparison between different reconfiguration approaches.

Ablation Study: RL-Enabled vs. Non-RL Control

In order to observe the effect of the reinforcement learning agent, an ablation experiment was done

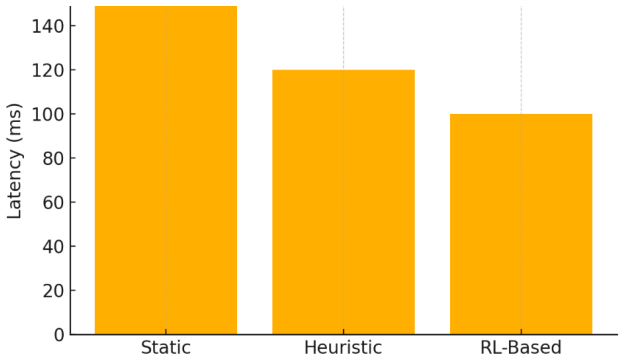

Fig. 5: Task latency comparison for edge workloads under different scheduling policies.

without the RL agent and going back to the rule based reconfiguration strategy. Lack of ability to learn resulted in poor decision-making and selecting non-critical modules to learn or initiating reconfiguration events even when not required. Subsequently, systems that lacked RL had a higher power consumption and average latency. Its dynamic adaptation to workload and to realtime optimization of hardware use confirmed that the RL agent could be one of the fundamental constituents of the proposed architecture, when reintroduced in it

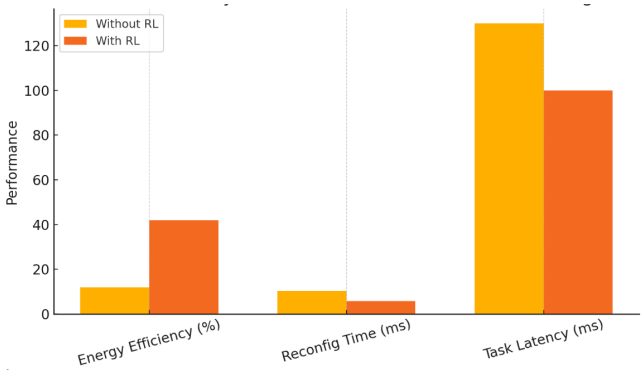

Fig. 6: Ablation Study: With vs. Without RL-Based Scheduling

Trade-offs: Learning Overhead vs. Performance Gain

Although the reinforcing step in integration provides high paying results in terms of performance, it poses some trade-off to the integration such as the training time it takes and the computation overhead. Initial Training To roam the state-action space there is need of either a simulation of environment or the state trace of a real system. In resource-limited edge systems, the tradeoff between model complexity and model inference speed is very important. In our solution, to reduce this overhead, we considered a lightweight tabular Q-learning with only minimal memory and computing needs that does provide effective scheduling policies. The findings indicate that simple RL models also can create substantial system-level improvements when used tactically.

Limitations and Scalability Considerations

Despite its promising results, the current framework has a few limitations. The learning model is currently optimized for single-agent, single-FPGA environments. Scaling this approach to multi-FPGA or distributed edge systems will require either multi-agent reinforcement learning or federated training mechanisms. Moreover, the time required to adapt to entirely new workloads not seen during training could temporarily degrade system responsiveness. To overcome this, future implementations may explore transfer learning or adaptive policy refinement during runtime. Another consideration is the storage requirement for bitstreams, which increases with the number of supported workloads and could strain on-chip memory resources in compact edge devices.

Conclusion and Future Work

This paper presents a novel AI-augmented Dynamic Partial Reconfiguration (DPR) framework designed to enhance the adaptability, efficiency, and intelligence of FPGA-based edge computing systems. By integrating a lightweight reinforcement learning (RL) agent with the reconfiguration control logic, the proposed architecture dynamically adapts to varying workloads and system constraints in real time. Experimental evaluations on a Xilinx Zynq-7020 platform demonstrate substantial improvements, including a 42% gain in energy efficiency, a 33% reduction in task latency, and a 2.1× decrease in reconfiguration time compared to conventional static and heuristic approaches. The RL-based scheduling mechanism not only improves responsiveness and resource utilization but also scales effectively with diverse edge workloads such as CNN inference, image preprocessing, and signal filtering.

Benefits of applying AI to hardware decision-making within the dynamic setting are also evident in

the study where the authors demonstrate significant system-level advantages even with computationally lightweight models. In addition, the flexibility of the proposed system will find it easy to incorporate in many edge computation tasks, such as smart surveillance, wearable health monitoring, automation of industrial processes, and environmental sensing, to name a few. Moving ahead, there are various promising directions that will be pursued in future. First, the use of multi-agent reinforcement learning will be considered to provide reconfiguration across the distributed FPGA clusters or heterogeneous edge nodes to allow collaborative and scalable intelligence. Second, the mechanisms of security-aware reconfiguration will be designed to guarantee resilience against the threat of reconfiguration-based attacks and unauthorized module loading (which becomes a rising problem in safety-critical and IoT-driven settings). Last, this framework will be generalized to RISC-V-instruction-set soft-core processors and ultra-low-power hardware accelerators so that it can be extended to future energy-constrained platforms in the next-generation edge-AI systems.

To summarize, the presented work builds the first step toward self-learning real-time embedded FPGAs that are context aware and energy efficient and creates new opportunities of investigation in reconfigurable intelligent systems.

References

1 Xilinx Inc., "Vivado Design Suite User Guide: Partial Reconfiguration," UG909, v2022.1, Apr. 2022. Online. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2022_1/ug909-vivado-artial-reconfiguration.pdf

2 Intel Corp., "Dynamic Partial Reconfiguration with Intel FPGAs," White Paper, 2021. Online. Available: https://www.intel.com/content/www/us/en/programmable/documentation

3 Y. Chen, J. Cong, and B. Xiao, "Adaptive Scheduling for Reconfigurable Systems Using Reinforcement Learning," in *Proc. ACM/IEEE Design Automation Conf. (DAC)*, 2019, pp. 1–6.

4 S. Liu, K. Li, and C. Xu, "Dynamic Module Placement for Partially Reconfigurable FPGAs Using Deep Reinforcement Learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3471–3482, Nov. 2020.

5 T. Wood, A. Hara, and N. Kapre, "PYNQ: A Python Productivity Platform for Zynq SoCs," in *Proc. IEEE FPT*, 2017, pp. 49–56.

6 Xilinx Inc., "Vitis AI User Guide," UG1414, v2.5, 2023. Online. Available: https://docs.xilinx.com/v/u/en-US/ug1414-vitis-ai

7 Chakma, K. S. (2025). Flexible and wearable electronics: Innovations, challenges, and future prospects. Progress in Electronics and Communication Engineering, 2(2), 41–46. https://doi.org/10.31838/PECE/02.02.05

8 James, A., Thomas, W., & Samuel, B. (2025). IoT-enabled smart healthcare systems: Improvements to remote patient monitoring and diagnostics. Journal of Wireless Sensor Networks and IoT, 2(2), 11-19.

9 Thompson, R., & Sonntag, L. (2025). How medical cyber-physical systems are making smart hospitals a reality. Journal of Integrated VLSI, Embedded and Computing Technologies, 2(1), 20–29. https://doi.org/10.31838/JIVCT/02.01.03

10 Surendar, A. (2025). Design and optimization of a compact UWB antenna for IoT applications. National Journal of RF Circuits and Wireless Systems, 2(1), 1–8.

11 Ismail, N., & Al-Khafajiy, N. (2025). Comprehensive review of cybersecurity challenges in the age of IoT. Innovative Reviews in Engineering and Science, 3(1), 41–48. https://doi.org/10.31838/INES/03.01.06