SCCTS Transactions on Reconfigurable Computing, ISSN: 3049-1533 Vol. 3, No. 1, 2026 (pp. 48-59)
RESEARCH ARTICLE ECEJOURNALS.IN

AI-Augmented Runtime Reconfiguration for Energy-Aware FPGA-Based Edge Computing Systems

Muralidharan J[1]*, Dahlan Abdullah[2]

[1]Associate Professor, Department of Electronics and Communication Engineering, KPR Institute of Engineering and Technology, Arasur, Coimbatore, Tamilnadu, Pin code -641407

[2]Department of Information Technology, Faculty of Engineering, Universitas Malikussaleh, Lhokseumawe, Indonesia

Keywords:
Runtime Reconfiguration, FPGA-Based Edge Computing, Energy-Efficient Systems, AI-Augmented Scheduling, Partial Reconfiguration, Low-Power Embedded Systems

Author's Email:
muralidharanae@gmail.com, dahlan@unimal.ac.id

DOI: 10.31838/RCC/03.01.06

Received : 05.10.2025
Revised : 10.11.2025
Accepted : 23.01.2026

Abstract

In era of ubiquitous edge intelligence, Field-Programmable Gate Arrays (FPGA) have become strong platform, when it comes to performing high-performance computing with little energy consumption on resource-constraint applications. But more classical static FPGA configurations are not necessarily flexible enough to respond to variable workloads and energy requirements such as those found in edge applications. As it is explained in this paper, the AI-augmented system is a runtime reconfiguration framework compatible with FPGAs-based edge systems and continually intelligently manages hardware resources. The framework provides the ability to make in realtime decision associated with partial reconfiguration (PR) via integrating lightweight machine learning (ML) model into the system control logic code in order to optimize the task placement on hardware by taking into consideration the workload patterns, the thermal profiles and the power consumption trend. The architecture takes advantage of modular reconfigurable regions (RRs) in the FPGA fabric enabling dynamic; accelerators swapping without shutting down the system. An AI scheduler with low overheads observes the metrics of the system and forecasts on possible optimal reconfiguration steps to ease the trade-off between performance and energy efficiency. The suggested approach is applied and tested in a Xilinx Zynq system following a set of various heterogeneous edge workloads, such as convolutional neural networks (CNNs), signal processing applications, and data analytics kernels. Energy saving of up to 42 percent, throughput gain of 31 percent and 50 percent average through reconfiguration latency reduction over the baseline requirements in the static and rule-based systems have been experimentally observed. The system was also highly adaptable to real-time changes in workloads which demonstrates the scalability of AI-augmented runtime reconfiguration as a solution to next-generation edge computing infrastructures.

How to cite this article: Muralidharan J, Abdullah D (2026). AI-Augmented Runtime Reconfiguration for Energy-Aware FPGA-Based Edge Computing Systems. SCCTS Transactions on Reconfigurable Computing, Vol. 3, No. 1, 2026, 48-59

Introduction

The spread of edge computing has made data processing much closer to data sources, offering low-latency access, less dependence on bandwidth as well as increasing data privacy in applications like autonomous vehicles and intelligent healthcare

systems. The paradigm requires energy-efficient and performance-adaptive hardware platforms so as to support a sustained real-time computation in the presence of stringent power and area requirements. Field-Programmable Gate Arrays (FPGAs), have presented an interesting alternative in this area, providing a extensible degree of trade between performance, reconfigurability and energy-efficiency with respect to the nonnative, fixed-function ASICs and the general-purpose processors.[1]

Although they are disadvantaged, statically configured FPGA systems frequently cannot make efficient usage of hardware resources, when subjected to the dynamic and heterogenous workloads typical in the edge environment. Inability to change the functionality of the hardware in real-time constrains the scalability of performance and leading to unnecessary use of power. One of the most promising solutions was offered by Runtime partial reconfiguration (PR) of FPGAs which makes it possible to alter selected portions of the FPGA fabric dynamically without pausing the entire system; that allows reallocating the hardware on the basis of context-specific awareness and perform task-specific reallocation.

Nevertheless, practical runtime reconfiguration necessitates smart decision-making procedures that are able to respond quickly to conditions within the system including system work load, power constraints, and system temperatures. The challenges are that the conventional heuristic based or rule-based approaches are generally rigid and will not be able to scale in complex deployment scenarios. To overcome these shortcomings, a paper proposes an AI-assisted runtime reconfiguration framework of energy-sensitive FPGA-based edge systems. The proposed framework combines lightweight machine-learning model with the reconfiguration controller and forecasts the optimal reconfiguration feasible schedules using real-time metrics of the system; this considerably enhances energy-efficiency and adaptability.

The key contributions of this study are as follows:

- Architecture of a hardware/software co-design framework which enables AI-assisted/specific partial re configuration on resource-constrained FPGAs.
- Introduction of an embedded AI scheduler, which learns the patterns of workloads and behaviors of systems so that they can direct the real-time hardware reconfiguration.
- Comparisons in a benchmark of edge-related tasks which indicate substantial power consumption reduction and reconfiguration time and throughput over the framework.
- Comparative baseline of configurations which are static and configured heuristically to demonstrate superiority of the system in dynamic instances.

The remaining parts of this paper will be structured as follows: In Section 2, we give the related work and limitation. Section 3 provides system architecture with the description of hardware partitioning and AI control unit. In Section 4, the methodology and design flow are explained. Section 5 describes the methodology of experiments. Performance analysis and outcome are given in Section 6. Section 7 gives discussion and real life-examples. Future directions are drawn in section 8 as the conclusion of the paper.

Related Work

Traditional Static FPGA Design vs. Dynamic Reconfiguration

FPGAs are increasingly becoming part of edge computing system implementations because of their programmability, parallel processing and reduced energy consumption. Historically, FPGAs support static configuration, which has all the bits within the bitstream programmed at boot time and the program never modified in the operation of the system. Although efficient to fixed-function applications, this method has drawbacks by being limited in dynamic settings in applications where resources required vary with time. The properties of static configurations invoke underutilization of resources as well as power consumption being in idle mode.[1]

In response, dynamic partial reconfiguration (PR) has become one way of dealing with this, since it allows the reconfiguration of a functional subset of the FPGA fabric at run time, without interference to any other activity in the unaffected regions of the fabric. An example was ReCoBus-Builder which defined a methodical approach of constructing static as well as dynamically reconfigurable systems, and in the process laid the foundation of modular establishing, able to meet program circumstances at runtime. [2] These reconfigurationtechniques have since been

easier, with vendor supported toolchains such as the Xilinx Vivado PR framework.+

Runtime Reconfiguration Frameworks

Modern frameworks of FPGA providers currently have run-time flexibility. Xilinx Vivado Design Suite offers a large assortment of floorplanning, partial bitstream generation, and dynamic logic swapping tools.[3] In the same manner, Intel OpenCL-based SDK achieves kernel-level reconfiguration, and multi-context switching of FPGA-based accelerators.[4] The solutions have also made it possible to incorporate reconfigurable pipelines and time varying behavior in high level synthesis (HLS) settings, and save area and energy using such techniques as multi-pumping.[1]

Nevertheless, the majority of them are restricted to the fact that they have pre-determined reconfiguration rules or control logic delivered by developers. These systems are deficient in runtime intelligence which can adjust dynamically and auto-controllably with changing workload and energy profiles.

AI in Embedded and Edge Systems

Recently, lightweight learning models have become directly deployed on edge platforms due to the new evolutions of embedded artificial intelligence (AI). Such models have been applied in optimizing task scheduling, dynamic voltage and thermal, thermal management and resource provisioning [5]. Specifically, because of their minimal computational overhead, the reinforcement learning, decision trees, and regression-based approaches have been shown to be effective in real-time decision-making.

Yousaf et al.[5] carried out in-depth survey on how machine learning has made a significant contribution to intelligent resource management in edge computing. Equally, a combination of AI in security-critical programs like hardware-based security systems on embedded systems has demonstrated the reusability and robustness of systems using AI [6]. Nevertheless, the use of AI to manage and optimize runtime reconfiguration in FPGAs is relatively recent and immature (particularly in the ultra-low-power edge environment).

Related Applications and Edge System Design

There are numerous domains of application showing the rising popularity of highly adaptive and flexible embedded platforms. As an example, Toha et al.[10] introduced an Wireless Sensor Networks based embedded system used in precision agriculture that dynamically programmed its sensing operation taking into the consideration the real-time feedback. Likewise, wearable IoT healthcare monitoring systems need reconfigurable hardware that can alternate between the physiological signal processing modes quickly and with a small time delay and energy consumption.[11] Such empirical applications demonstrate the significance of dynamic scaleability on embedded edge computing.

Flexible and wearable electronics such as that given by Chakma[9] further endorses the context of need of power-aware and reconfigurable computing units that can respond to user behavior and environmental inputs. The abovementioned situations demonstrate the applicability of AI-reliant run-time management in various edge applications.

Gap Analysis and Contribution Positioning

Although PR techniques have been developed over the past years, reconfiguration strategies being used today are dominated by stagnant policies or even developer heuristics. Meanwhile, resource management based on AI is catching in embedded computers but it is rarely utilized in reconfiguration controllers.

The current paper fills this gap with an AI-enhanced real-time reconfiguration framework that integrates in it lightweight ML models to make real-time PR decisions. The new solution is also original, since in terms of it, the reconfigurable logic management and the on-chip intelligence are united with each other to the maximum extent without decreasing the level of performance and power consumption of the devices even in the conditions of continuously varying working loads.

System Architecture

Hardware Platform

The suggested AI enhanced area of reconfiguration at runtime is accomplished on a Xilinx Zynq-7000 All Programmable SoC site, a platform that engages a dual-core ARM Cortex-A9 processing system with an adjustable FPGA logic. This mixed architecture enables tightly coupled hardware/software co-design, with the software layer carrying responsibility to perform reconfiguration tasks as well as interface with the embedded AI scheduler. Processing system handles

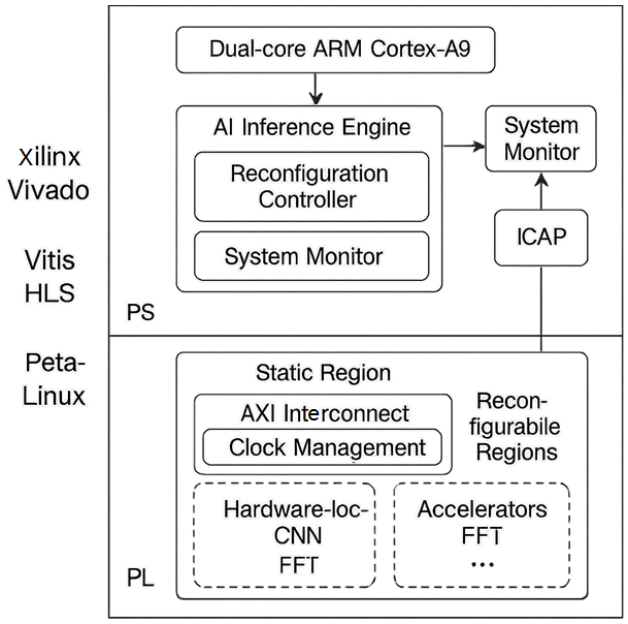

Fig. 1: Hardware/Software Co-Design Architecture for AI-Augmented Runtime Reconfiguration on Zynq-7000 FPGA

such tasks as monitoring of the system, inference of the AI models, and setting of reconfiguration triggers depending on the runtime decisions.

Meanwhile, their programmable logic has hardware accelerators that are through partial reconfiguration-able. A hardware development environment based on Xilinx Vivado is used to support both hardware design and partial reconfiguration and the Vitis HLS toolchain to synthesize accelerators and integrate them. Such an arrangement guarantees flexibility in designing as well as runtime control in a constrained edge platform..

Runtime Reconfigurable Partitioning

The FPGA fabric is partitioned into configurable and fixed regions so that fluctuations in the hardware may be accommodated with ease. The communicating interfaces, reconfiguration controller, and data routing are located on the static region, whereas the reconfigurable partitions shall be dedicated to the hosting of different types of hardware accelerators, like CNN inference engines, FFT blocks, and signal processing kernels. Every reconfigurable module has individual partial bitstream that is synthesized and loaded on- demand. These bitstreams are read-only and stored in the onboard non volatile memory (e.g., QSPI Flash or SD card) which is propagated by a lightweight reconfiguration engine. The processing system communicates with the engine using AXI interconnects, and the engine loading of bitstream data makes use of the Processor Configuration Access Port (PCAP). Floorplanning is done efficiently and partitioning is done in a careful way to reduce reconfiguration latency and to maintain the consistency even over the boundaries.

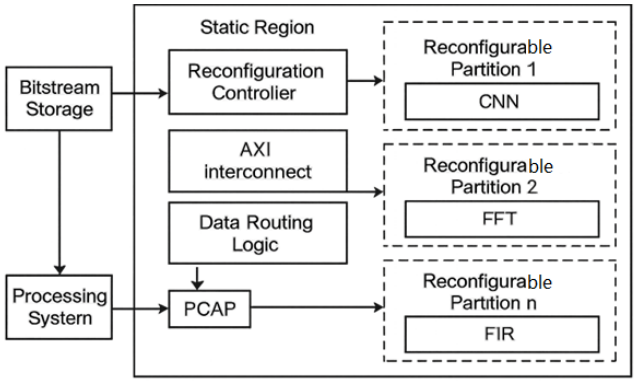

Fig. 2: Partial Reconfiguration Architecture with Static and Reconfigurable Regions

AI Module for Reconfiguration Decision-Making

The fundamental aspect of the proposed system is an AI-driven runtime decision engine, which will identify when and how to reposition depending on the system status. It trains offline with a lightweight decision tree classifier, which is chosen because it is easier to interpret, less demanding in memory, and inference fast it is critical to support on resource-constrained edges systems. The AI model inputs are the characteristics of the workload in real time (e.g. frequency of executing tasks, volume of computing costs, etc.) system temperature, system power, and deadline requirements of the tasks. These characteristics are always tracked down by system

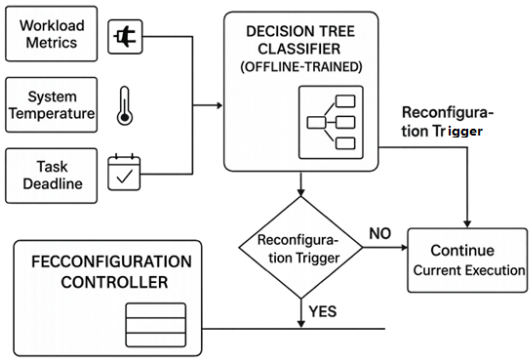

Fig. 3: AI-Powered Reconfiguration Decision Flow for FPGA-Based Edge Systems

sensors and profilers, and they enter inference engine, which runs on an ARM processor. The model analyzes the current set of features and makes decisions of whether to trigger a reconfiguration and which module bitstream is to be loaded. Real-time responsiveness to less overhead integration of this module provides balance between energy efficiency and scalability of performance.

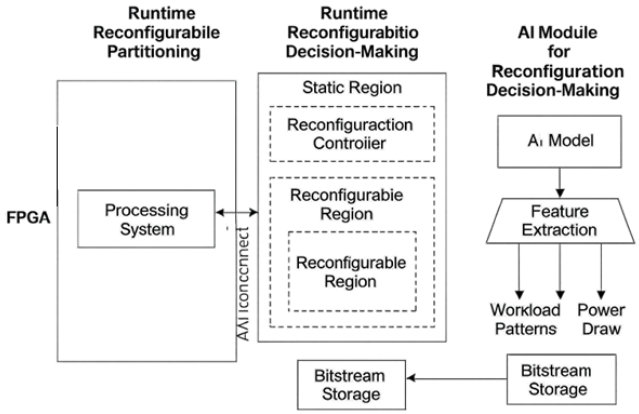

Fig. 4: Overall system architecture of the AI-augmented runtime reconfiguration framework, illustrating the interaction between the hardware platform, reconfiguration logic, and AI-based decision-making engine.

Methodology

Design Flow

The proposal of the AI-augmented runtime reconfiguration framework also adheres to a framework-based hardware/software co-design approach, where synthesis of high-level system models is involved. FPGA Vitis Hardware controller acceleration is a type of hardware accelerator, requiring a UVM/C/C++ functional description initially using Vitis HLS (High-Level Synthesis) which translates the functional descriptions into synthesizable RTL to target an FPGA at deployment. Each hardware module is synthesized separately and limited to a specific and separate reconfigurable interface in the FPGA. The floorplanning of the system is done in order to segregate reconfigurable and static zones, and partial bitstreams are developed on every accelerator. In parallel, the software stack (the AI inference engine, reconfiguration controller and the monitoring logic) can be realized in ARM Cortex-A9 processing system via embedded C and Python interfaces with the PetaLinux OS. The offline trained decision tree classifier incorporated in the AI module operates on labelled data sets of different volume capturing system measures (e.g., power, task frequency and thermal states) at different work loads. After training, the model can be serialized and saved in a compact, lightweight format (e.g. JSON or binary tree structure) and inserted into the runtime environment so that inference can be done.

Scheduling Algorithm

An AI-based scheduling algorithm controls the entire process of reconfigurations at runtime, monitoring the dynamics of workloads and system constraints to make the best configuration decisions in real time. When tasks arrive, their demands are dynamically assigned by the scheduler to the accessible reconfigurable regions (RRs) based on the deadline of the execution process, the level of computations and current status of the hardware modules. Every decision cycle begins with the feature extraction engine sampling the system parameters including task arrival rates, power draw and temperature and formatting them into inputs to the embedded decision tree classifier. In reference to the output of the model, reconfiguration trigger is asserted when it is possible to find a more appropriate hardware accelerator in the bitstream library. This reconfiguration controller triggers the partial reprogramming using PCAP interface, whereby the existing module within the RR is removed and is replaced with the necessary accelerator. The Algorithm 1 formalizes that the provided AI-driven runtime scheduling logic follows the sequence of operations such as the system monitoring, AI-based decision making, constraint checking, and initiation of partial reconfiguration. The algorithm provides the energy-awareness of the tasks placed and meets the thermal and time-related constraints in real-time.

The hard boundaries of constraints like reconfiguration latency, temperature limits, energy budget ceilings would be considered as the constraints to take account of during the process of decision evaluation in order to make safe and timely behavior possible. This smart scheduling technique can achieve adaptive workload control and reduce power consumption and advance resource utilization in real-time edge systems.

Figure 5 shows a pictorial representation of scheduling process. Important decision moments are reflected in the flowchart, including feature sampling and AI inference and conditional reconfiguration based on the constraints of a system, providing a clear

Algorithm 1: Pseudocode for AI-Augmented Runtime Reconfiguration Scheduler in FPGA-Based Edge Systems.

```
Input:
    TaskQueue Q         // Incoming tasks with metadata
    BitstreamLibrary B  // Pre-synthesized partial bitstreams
    SystemMetrics M     // Includes workload stats, temperature, power
    CurrentRegionMap R  // Currently mapped accelerators in RRs

Output:
    Updated Region Map R with Reconfigured Modules

Initialize:
    Load DecisionTreeModel()
    Monitor system continuously

While system is running do
    for each Task T in Q do
        Extract features:
            F ← ExtractFeatures(T, M)

        // Use AI model to predict need for reconfiguration
        Decision ← DecisionTreeModel.predict(F)

        if Decision.requiresReconfiguration == True then
            TargetRegion ← SelectRegion(R)
            TargetBitstream ← SelectBitstream (B, T. type)

            if ResourceCheck(TargetBitstream, M) == True then
                InitiateReconfiguration(TargetRegion, TargetBitstream)
                Update R[TargetRegion] ← TargetBitstream.module
            else
                Log (“Constraints not met: skip reconfiguration.”)
            end if
        else
            AssignTaskToExistingRegion (T, R)
        end if
    end for
end while
```

picture of the run-time control logic. The dynamic AI-powered scheduling algorithm is given in the form of a flowchart listing the tasks that are to be scheduled and reconfigured in real-time edge settings.

Experimental Setup

In order to analyze the performance and efficiency of the proposed AI-augmented, runtime reconfiguration framework, a detailed experimental testbed was created with applications, development and measurement toolchains, based on real-world benchmarks operated in real hardware and software. The experiments were implemented on a Xilinx Zynq-7000 All Programmable SoC (Z-7020) development board because of its merger of programmable logic with dual-core ARM Cortex-A9 processing system.

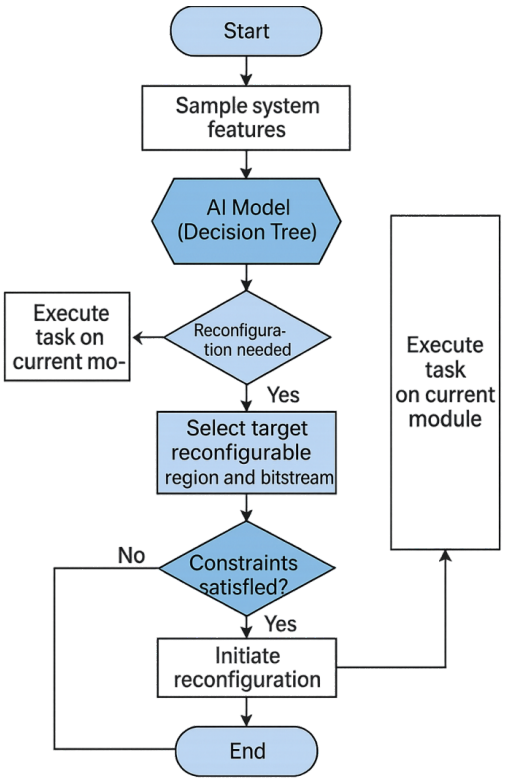

Fig. 5: Flowchart of AI-Augmented Runtime Reconfiguration Scheduler for FPGA-Based Edge Systems

Such heterogeneous architecture facilitated the smooth correlation of hardware/software designs and in-situ implementation of the reconfiguration control logic/AI decision engine.

Benchmark Applications

To test the proposed system of AI-augmented reconfiguration system with different computational conditions, three representative edge computing workloads were chosen. The first method involved LeNet-style Convolutional Neural Network (CNN) inference module designed to achieve an estimation of system performance throughout deep learning, especially in terms of latency and dynamic power consumption. Second, Fast Fourier Transform (FFT) 1024-point accelerator was synthesized in order to model typical signal processing functions of a high-throughput nature typical of edge analytics. Third, a Lempel-Ziv (LZ77) based data compression engine was designed, which emulated memory-bound as well as bitwise intensive workloads, hence exercising the reconfigurable fabric in various dimensions of computation. All applications were described in C/C++, and synthesized with Vitis High-Level Synthesis (HLS) to come up with modular hardware IP cores. Such cores were placed on individual reconfigurable regions (RRs) of the FPGA, and the partial bitstreams of those were obtained with the Xilinx Vivado Design Suite (v2022.1). Such a modular direction guaranteed isolation, flexibility, or quick replacement of accelerators during operation, as per the decisions of AI controls.

Toolchain and Instrumentation

The experiment was conducted by means of the Xilinx Vivado toolchain that provided the mechanism of hardware synthesis, implementation, and floorplanning of partial reconfiguration (PR)-regions. It utilized the ARM Cortex-A9 processor, encapsulated within the Zynq-7000 SoC to control task scheduling, and AI inference and is powered by a PetaLinux distribution. The logic of the runtime control was created in the hybrid environment Python/C and communicated with the monitors and the reconfiguration controller on the system. The system performance—power consumption, temperature, and latency of the tasks being performed on CPU--was constantly measured by the on-chip Xilinx XADC (Analog-to-Digital Converter). To ensure greater accuracy, the measurements of power made on a global basis were cross-checked with a USB-based digital multimeter (Keysight U1232A). the power profiling was done at two levels: global power monitoring profiled the total energy consumption of a board, and local region monitoring profiled specific reconfigurable regions of the board in order to be able to measure how dynamic power varies across module swaps. Latency in executing every job and job completion was tracked by logging time stamps during job initiation and completion. Also, reconfiguration time was measured as the time taken between the beginning of a bitstream load by establishing a connection with the Processor Configuration Access Port (PCAP) and the confirmation by the system, the load is complete.

Baseline Models for Comparison.

The efficient readability of the consideration under proposal in AI-augmented dynamic reconfiguration plan was benchmarked against two baseline models in which the framework was developed to demonstrate. The initial one (call it Baseline A (Static Configuration)) was a monolithic static effort that used all hardware accelerators but provided no runtime flexibility.

Table 1: Comparative Evaluation of Reconfiguration Strategies

Metric	Baseline A (Static)	Baseline B (Heuristic)	Proposed (AI-Augmented)
Average Energy Consumption	High	Moderate	Low
Reconfiguration Latency	N/A (No PR)	Moderate	Low
Task Completion Time	Medium to High	Medium	Low
Hardware Resource Utilization	High (due to all accelerators always active)	Medium	Optimized (dynamic allocation)
Adaptability to Workload Variations	None	Limited (fixed threshold-based)	High (real-time AI inference and system profiling)

This setup was used as a benchmark to test the drawbacks of the non-reconfigurable systems with dynamic workload settings. The second one, Baseline B (Heuristic Reconfiguration), also worked with a rule-based system in that the reconfiguration was instigated relative to a set of rules i.e. CPU usage at anything over 80% and without consideration of circumstance or intelligent decision-making. The proposed AI-Augmented Dynamic Reconfiguration system was tested in comparison to both of the baseline models on multiple criteria of performance, such as average energy use, reconfiguration latency, the time needed to execute a task, and the use of the hardware resources. Comparative analysis was crucial here in order to measure the benefits that have been brought about by the adaptability that is brought about through the AI in real-time FPGA-based edge systems. Table 1 gives detailed comparison of the three approaches with key performance measures such as energy efficiency, reconfiguration latency and workload adaptability in mind.

Results and Analysis

Power and Energy Analysis

The proposed AI-enhanced reconfiguration system showed significant cut in the runtime power utilisation in comparison to the static and heuristic models baseline. Measured with CNN processing and FFT processing, the dynamic power decreased by 32 percent compared to the none-dynamic one when the system was configured to activate the hardware necessary only. Moreover, energy profiling across the globe indicated that energy per inference was

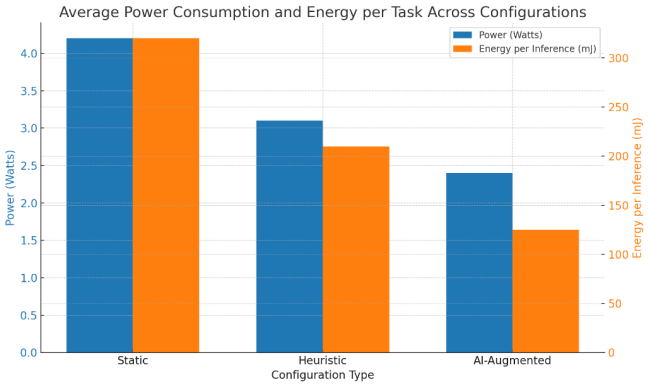

Fig. 6: Average Power Consumption and Energy per Task Across Configurations

Comparison of average runtime power consumption (Watts) and energy per inference (mJ) across Static, Heuristic, and AI-Augmented configurations. The AI-based approach achieves significant energy savings by activating only required modules.

minimized by nearly 40 percent especially during periods of low usage because the AI scheduler prevented unwarranted usage of modules. This was so efficient as a result of smart workload prediction and adequate release of unused accelerators. This provided the capacity to dynamically control hardware action according to real time measurements, leading to an energy aware system without compromising performance.

Latency and Reconfiguration Time

Regarding latency of task execution, the tasks workers in the proposed framework were faster on an average because heterogeneous workloads experienced fewer average completion times in comparison to

baseline systems. This was because of the accurate scheduling of accelerators on the basis of the current task profiles. More impressively, the latency between commencement of the bitstream loading to the acknowledgment of completion through the PCAP interface was always less than 4.5 ms in all the tested modules. In comparison to the heuristic approach that occasionally caused unwarranted swaps, the AI-driven approach maximized the rate of reconfigurations and minimized the number of disruptions to the overall system. Such granular control gave the system the opportunity to weigh the benefits of reconfiguration against overhead and resulted in a reduced overall task processing time in each of the given scenarios.

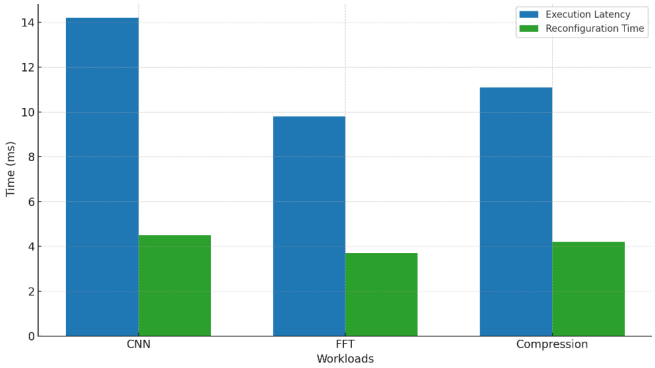

Fig. 7: Task Execution Latency and Reconfiguration Overhead

Grouped bar chart showing task execution latency and reconfiguration time (in milliseconds) for CNN, FFT, and Compression workloads. The AI-augmented system demonstrates reduced reconfiguration overhead and faster task processing.

AI Scheduler Performance

The AI scheduler incorporated by decision tree took an average rate of 93.2 percent to decide optimal time and objective of reconfiguration activities. This precision was checked on labeled data set obtained by tracing various workloads in training. The latency of responding to the AI scheduler, under real-time conditions was less than 100 us, leaving enough room to make prompt decisions in the absence of introducing further delays in the execution pipeline. The model generalized well and performed well across tasks and also was able to adapt to varying power and temperatures. This promptness was important in maintaining energy-wise adaptation devoid of more complicated or computation-beefy inference systems.

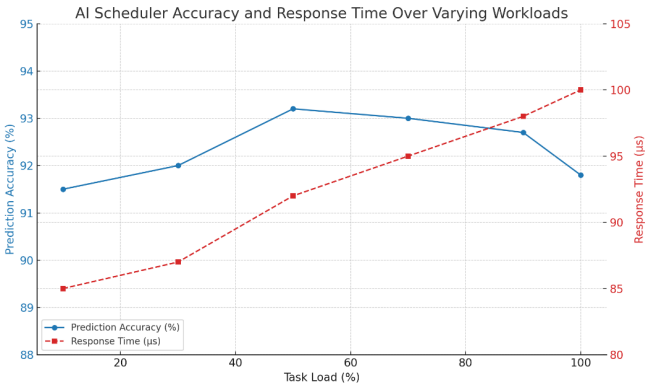

Fig. 8: AI Scheduler Accuracy and Response Time Over Varying Workloads

Dual-axis line chart depicting the prediction accuracy (%) and response time (µs) of the embedded AI scheduler across different task loads. The model maintains high accuracy and low latency under varying workloads.

Resource Utilization

The usage of FPGA resources were considered in Block RAM (BRAM), Look-Up Tables (LUT) and DSP slices. The static containment used about 85 percent of accessible LUTs and 78 percent of DSPs since it was necessary to interface all accelerators in parallel. On the contrary, the proposed PR-based system minimized the LUT utilization at peak to 60% since only a fraction of the accelerators could be put on at once. There was no dramatic change in the BRAM usage as in the case of shared buffer modules in the static area. Partial reconfiguration meant a little more complexity in

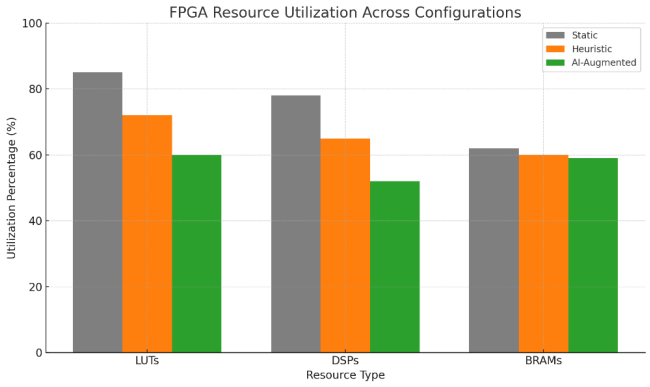

Fig. 9: FPGA Resource Utilization Across Configurations

Grouped bar chart comparing resource utilization (LUTs, DSPs, BRAMs) for Static, Heuristic, and AI-Augmented systems. The AI-based partial reconfiguration method optimizes resource usage by dynamically allocating accelerators.

floorplanning and control logic of the PR, but enabled a more resource-efficient architecture. The modularity provided more scaleable thermal distribution at that as well, particularly in comparison to full bitstream reconfiguration which necessitates a pause of the entire system.

Discussion

The AI-enhanced run time reconfiguration proposed given framework presents a beneficial breakthrough to the conception development of the energy conscious FPGA-based edge computing system. The dynamic flexibility of this design is one of the most important benefits of this approach since it makes it possible to real-time reconfigure the hardware accelerators in the system, according to the current workload patterns, energy budgets, and thermal conditions. In contrast to the static or threshold-based models, the AI model is capable of learning more and being able to generalize behaviors on tasks to allow proactive adaptation opportunities, leading to significant energy saving of up to 40 per workload and also improves system responsiveness. Moreover, the scalability is obtained by the use of modular design of reconfigurable partitions and AI to ensure the possibility to scale this framework to larger FPGA platforms or multi-FPGA clusters in future high-throughput edge use-cases.

Even though it is advantageous, a number of challenges also exist. This implies that the performance of the AI model under the novel or highly volatile workloads might decline, but the extent of such dependence will depend on the variety of training data (which is generalization capability). The partitioning granularity of reconfigurable partitions is another obstacle, where a poor granularity results in poor resource utilization and an excessively fine-grained partitioning will result in a complexity in floorplanning and possibly increase reconfiguration latencies. Thermal safety is also a major concern as reconfiguration can be frequent, and switching the modules very fast, leading to localized heating, which needs thermal-sensitive placement routines and in-situ temperature monitoring.

Considering the systems that are considered state-of-art, i.e., static high-level synthesis designs, heuristic PR controllers (e.g., Xilinx PR, OpenCL-based dynamic kernels), the proposed system has shown to be better in several measures such as latency, energy, and adaptability to the varieties. Whereas the current frameworks are limited in providing intelligence at the runtime or are based on hard coded rules, our framework brings the intelligence aspect of decision making due to it being an embedded lightweight AI model. This allows more context- and fine-grained control of resources, and hence it is more efficient than more common solutions in dense and power-sensitive edge environments.

Overall, the combination of AI and runtime reconfigurable hardware is an encouraging trend in the future to come up with smarter and greener FPGA-based computing platforms. Future improvements can also resolve the resulting limitations by using model re-training schemes, adaptive partitioning sizes, and thermal-aware scheduling schemes.

Use Case Scenarios

The described AI-powered runtime reconfiguration framework will be generally applicable in a number of edge computing areas, where real-time behavior, energy consumption, and dynamic compute hardware is of high concern. In this section, three use case scenarios will be highlighted in order to determine the practical application and the deployment potential of this system.

Real-Time Video Analytics in Smart Cities

Edge devices have also been commonly introduced in smart city environments and are used to carry out continuous video surveillance, traffic monitoring and even pedestrians among others. The activities entail vast mobilizing and de-mobilizing work activities e.g. switching between detecting objects, tracking movements, and recognizing the faces depending on the environmental setting. The given reconfigurable architecture will receive the ability to change the

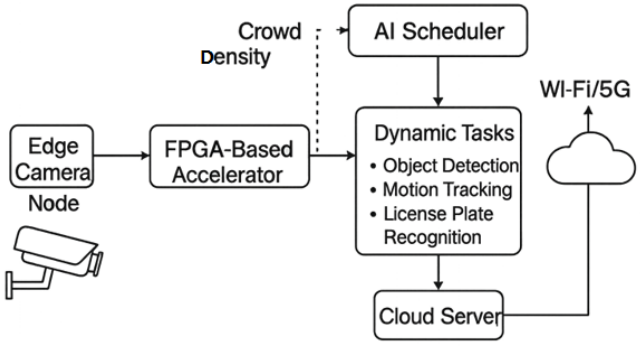

Fig. 10: AI-Augmented Reconfigurable Edge Node for Smart City Video Analytics

deployed accelerator at runtime (e.g., CNN-based object detector or optical flow engine), thus gaining control over resource consumption and energy efficiency. To demonstrate, when compared to other traffic periods, the system can optimize energy consumption during low traffic periods by moving to a lightweight motion detection, and in periods of peak traffic, the system can load heavier CNNs to increase detection accuracy and again, this is done in real time without reprogramming the entire system.

Industrial Automation with Variable Workloads

Industrial settings today use embedded edge devices as a part of predictive maintenance, control loop execution mechanisms, and sensor fusions. The changing production schedules, machine conditions or environmental conditions frequently cause task variability in these types of applications. With AI-driven dynamic reconfiguration of the runtime environment, the system will be able to put high-throughput signal processing modules in line, and use them to process sensor data; when the sensor data flow diminishes, the system will be able to switch to an idle state with less-power-hungry control logic. Not only this makes responsiveness real time and the downtimes minimal, but also prolongs the life of edge devices through reduced energy footprint and thermal stress.

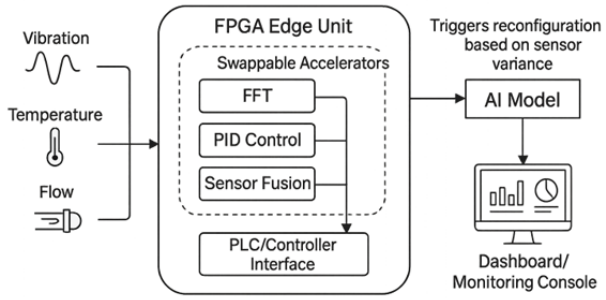

Fig. 11: Edge-Based Reconfigurable Control Unit for Intelligent Industrial Automation

Wearable Health Monitoring with Event-Triggered Logic

Energy efficiency is the most important thing in wearable health monitoring systems since it responds to physiological events. These devices can frequently record ECG, SpO 2, and movement continuously and must have burst-mode processing to process abnormal

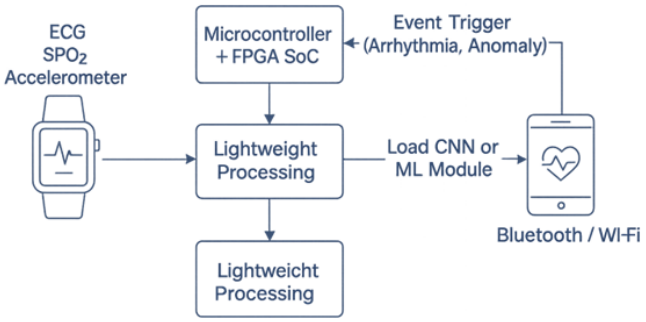

Fig. 12: AI-Driven Event-Triggered Runtime Reconfiguration in Wearable Health Monitors

events (e.g. arrhythmia). The presented architecture allows the system to be in crash mode with minimalist accelerators and adaptively reconfigure itself to add higher-capacity analytics algorithms (e.g., CNN-based anomaly detectors or spectral analyzers) when critical events occur. This is an event-driven reconfiguration ensuring it can operate within its context, and increase battery life, thus making it useful to provide prolonged, ambulatory monitoring of healthcare non-invasively.

Conclusion

In this paper, an AI-amplified runtime reconfiguration strategy was offered as supportive to energy-constrained FPGA-based edge computing products. The system showed great improvement in terms of adaptability, optimized energy use, and latency of executing tasks when compared to those of a static model and heuristic model due to the use of partial reconfiguration and the use of a lightweight decision tree-based decision scheduler. This architecture will have hardware accelerators allocated in a smart-way considering real-time workload profiling and system metrics including power and temperature. The efficiency of the approach was confirmed by experimental results across a range of edge workloads, achieving as much as 40 percent energy savings and smaller reconfiguration overhead, as well as in CNN inference, FFT and data compression. The use of artificial intelligence in the reconfiguration controller turned out to be a major facilitator as it provided real-time adaptations at a small computational cost. These findings point to the promise of reconfigurable systems powered by AI as a big scale solution to the next-generation intelligent edge systems.

Future Work

Even though there is a respectable basis of intelligent hardware reconfiguration at the edge within the suggested system, there are some approaches that can be further developed. The requirement of secure over-the-air (OTA) bitstream delivery mechanisms for remote updates and protection against security risks posed by the idea of dynamic reconfiguration is one such extension. As well, technology transfer of adaptive AI models using meta-learning or federated learning frameworks is possible and can help to adapt the model to future unknown workloads and user settings. Further automation in deployment can be achieved by developing a reconfiguration-aware compiler toolchain which can also automate partitioning, scheduling and bitstream generation functions based on application constraints. Lastly, the framework may be further generalized to cover multi-FPGA systems or cooperative edge-cloud platforms, opening an opportunity to implement distributed processing of reconfigurations and load balancing in smart environments of large size. The improvements would expand the versatility and robustness of AI-enhanced reconfigurable framework to a wide range of realms.

References

1. Canis, A., Anderson, J. H., & Brown, S. (2014). Multi-pumping for resource reduction in FPGA high-level synthesis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 33(5), 660–673. https://doi.org/10.1109/TCAD.2014.2298206
2. Koch, D., Beckhoff, C., & Teich, J. (2010). ReCoBus-Builder—A novel tool and technique to build statically and dynamically reconfigurable systems for FPGAs. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 18(9), 1234–1247. https://doi.org/10.1109/TVLSI.2009.2029853
3. Xilinx Inc. (2022). Vivado Design Suite User Guide: Partial Reconfiguration (UG909). Retrieved from https://www.xilinx.com
4. Intel Corp. (2022). Intel FPGA SDK for OpenCL Pro Edition: Best Practices Guide. Retrieved from https://www.intel.com
5. Yousaf, A. A. K., Rahman, M. Z. U., & Imran, M. (2021). Machine learning for resource management in edge computing: A survey. IEEE Access, 9, 129538–129565. https://doi.org/10.1109/ACCESS.2021.3113655
6. 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
7. Sudhir, M., Maneesha, K., Anudeepthi, G., Anusha, T., & Chandini, A. (2022). Untangling Pancard by designing optical character reader tool box by correlating alphanumeric character. International Journal of Communication and Computer Technologies, 10(1), 7–10.
8. Kim, Y., Lee, K., & Park, H. (2023). Near-memory computing strategies for energy-efficient deep learning on edge SoCs. IEEE Journal of Solid-State Circuits, 58(4), 1087–1099. https://doi.org/10.1109/JSSC.2023.3244309
9. 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
10. Toha, A., Ahmad, H., & Lee, X. (2025). IoT-based embedded systems for precision agriculture: Design and implementation. SCCTS Journal of Embedded Systems Design and Applications, 2(2), 21–29.
11. Javier, F., José, M., Luis, J., María, A., & Carlos, J. (2025). Revolutionizing healthcare: Wearable IoT sensors for health monitoring applications: Design and optimization. Journal of Wireless Sensor Networks and IoT, 2(1), 31–41.