SCCTS Transactions on Reconfigurable Computing, ISSN: 3049-1533 Vol. 3, No. 2, 2026 (pp. 30-39)
RESEARCH ARTICLE ECEJOURNALS.IN

Low-Power Reconfigurable Architectures for Edge-AI: A Hardware-Centric Design and Optimization Perspective

Dahlan Abdullah[1]*, Saravanakumar Veerappan[2]
[1]Department of Informatics, Faculty of Engineering, Universitas Malikussaleh, Aceh, Indonesia.
[2]Director, Centivens Institute of Innovative Research, Coimbatore, Tamil Nadu, India.

Keywords:
Reconfigurable Computing, Edge AI, Low Power Design, FPGA, CGRA, Dynamic Partial Reconfiguration, Hardware Acceleration, Energy-Efficient Architecture, AI Inference

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

DOI: 10.31838/RCC/03.02.04

Received : 17.11.2025
Revised : 13.01.2026
Accepted : 21.05.2026

Abstract

The increased demands in energy efficient artificial intelligence (AI) at the edge have motivated the research on the low power reconfigurable computing architecture. Such architectures, and field-programmable gate arrays (FPGAs) and coarse-grained reconfigurable arrays (CGRAs) in particular, offer an attractive tradeoff between computational flexibility and performance efficiency. This paper offers a hardware-specific perspective of design and optimization policies of deploying Edge-AI workloads on those platforms. Power reducing methods, such as a dynamic voltage and frequency scaling (DVFS), dynamic partial reconfiguration (DPR), and power gating are investigated comprehensively in the domain of inference operations in the AI environment. Also AI-specific hardware-level optimization, like quantization-aware design and resource-limited acceleration, is integrated. We have proposed a design frame work consisting of workload profiling, logic-level power modeling and architecture-aware mapping to maximize energy efficiency. Experiments of benchmark Edge-AI workloads such as convolutional neural networks (CNNs) and signal processing kernels show that up to 60% less power is consumed and has twice the energy efficiency over typical baseline static designs. The results underline that reconfigurable computing is a scalable and sustainable option of future edge intelligence systems to operate on ultra-low-power at the edge and deploy AI at runtime in resource-limited settings. The proposed research will add a single view of architecture-level and workload-level co-optimization in next-generation Edge-AI platforms.

How to cite this article: Abdullah D, Veerappan S (2026). Low-Power Reconfigurable Architectures for Edge-AI: A Hardware-Centric Design and Optimization Perspective. SCCTS Transactions on Reconfigurable Computing, Vol. 3, No. 2, 2026, 30-39

Introduction

The ubiquity of edge devices in smart cities, driverless cars, industrial IoT and healthcare monitoring has also ratcheted up the need to use real-time artificial intelligence (AI) inference at the network edge. In contrast to cloud-based AI processing, edge AI is limited with high energy, thermal and form factor constraints and development of energy-efficient but still high-performance hardware platform is of extreme importance. Reconfigurable computing, and especially field-programmable gate arrays (FPGA) and coarse-grained reconfigurable arrays (CGRAs) have also become an attractive solution because they are highly

adaptive to workload, have reduced latency, and can be reused. Devices on these platforms can support custom datapath architecture, optimized memory hierarchy and dynamic reconfiguration capabilities, which are suitable to the heterogeneous and time-varying nature of Edge-AI tasks.

Nonetheless, the current body of knowledge is itself mostly limited to performance-acceleration or steady-state energy-reduction measures, and ignores more broadly co-optimization solutions, which address dynamic power control, run-time flexibility, and AI-specific design constraints. Besides, although most of the AI accelerators implemented in FPGAs have shown encouraging performances, they are not fine-grained reconfigurable, energy-proportional and system-level power model based on which they cannot support ultra-low power requirements and mission-critical applications at the edge.[1, 2]

These limitations are discussed in this paper which introduces a hardware-focused model to design, optimize, and evaluate Key principles in architecting low-power reconfigurable platforms to execute Edge-AI workloads are discussed in this paper. It explores cutting-edge technologies like number voltage and frequency scaling (DVFS), dynamic partial reconfiguration (DPR), power gating and quantization of neural networks. Moreover, experimental analysis on real-world AI benchmarks point out that these architectures hold potential to drastically cut down energy cost without losing out inference latency or accuracy.

Background and Motivation

Edge-AI Requirements

Edge-AI applications such as autonomous navigation, wearable health monitors, and smart surveillance demand not only real-time inference with less latency, large energy saving, and small memory footprint. In comparison to centralized cloud AI systems, edge nodes have to work with limited power budget, restricted computational resources and dynamic runtime environments. These problems require creation of energy-efficient hardware that is dynamic.

Reconfigurable Computing Paradigms

Reconfigurable computing is also rising as a potential answer to the Edge-AI challenge by offering programmable structures of hardware enabling customization of a workload. Fine-grained architectures fine-grained (i.e., field-programmable gate arrays (FPGA), or fine-grained reconfigurable arrays (FGRAs)), give detailed control over the logic-level execution whereas, coarse-grained reconfigurable arrays (CGRAs) have a higher efficiency in mapping of data-parallel tasks with less complex interconnect.[1] These paradigms provide the best of both industries because it has the advantage of being near ASICs and similar to a general-purpose processor in their flexibility.

Limitations of Conventional Edge SoCs

Conventional systems-on-chip (SoC) designs that have been used in the edge tend to be fixed-function, and tuned toward the average-case. It causes low dynamism to the dynamic workloads where intra-task variance and input-dependent character have a great impact on the effectiveness and power consumption. Moreover, since these architectures are not built to be reconfigurable, they restrict their scalability and reusability to newer models of AI in the long run.

Motivation for Hardware-Centric Reconfiguration

Computationally and energy-trade offs can only be managed using scale-in/out hardware- centric techniques like reconfiguration. These strategies enable the fine-grained power management, and at run time, change to task characteristics, and, in general, the dynamic power management with mechanisms such as the dynamic voltage and the frequency scaling (DVFS), power gating, and dynamic partial reconfiguration (DPR). This is aimed at coming up with architectures that are flexible to changing workloads but at the same time are tight on energy budgets.

Related Work

Reconfigurable Hardware for Edge-AI

Past years have been important in terms of progress to on-chip deployment of AI inference workloads on reconfigurable platforms. Zhang et al.[2] designed a parameterizable CNN accelerator on FPGAs which are more energy-efficient than GPUs. In the same case, Chen et al.[3] suggested a memory-optimized FPGA-based CNN accelerator chipping away at outer memory

access. Nevertheless, such designs mostly require specifications that are only interested in representing static execution models and cannot accommodate run-time flexibility essential in edge setups.

Power Optimization Techniques

DVFS, clock gating, as well as approximate computing has been covered in multiple works in an attempt to optimize the power profile of reconfigurable systems.[4, 5] Although such strategies show performance improvements in the simulation or controlled environment, they do not perform well in context-aware workloads of AI that have dynamic power-performance requirements.

Dynamic Partial Reconfiguration (DPR)

DPR allows modified hardware areas to be reconfigured at runtime, providing the advantages of resource modularity and energy economy. However, reconfiguration latency and overheads on controls are still major challenges, where Li et al.[6] used DPR to selective reconfigure CNN layers. Low-latency DPR in the range of fine grained DPR suitable to the ultra-low-power edge is under-researched.

AI-Specific Hardware Optimization

Such toolchains as FINN[7] and DNNWeaver[8] have made it possible to scale quantized neural networks to FPGAs at a low memory and power consumption. But, they tend to focus on fixed models and are not dynamically reconfigurable and therefore are not easily applied to various edge workloads.

Research Gap

With these developments, unified design methodologies that combine AI-specific hardware optimization, dynamic reconfiguration, fine-grained power management have remained absent. In a large part of the current literature, performance and power optimization are discussed independently. The given paper attempts to address this gap by presenting an end-to-end hardware-oriented framework that co-optimizes logic-level and system-level metrics to develop energy-efficient, real-time AI-inference at the edge.

Design Considerations for Low-Power Reconfigurable Architectures

The co-optimization of logic, memory, and clock are needed when designing low-power reconfigurable architectures in edge-AI. Important strategies according to this section are described with experimental verification on up-to-date FPGA platforms.

Power Profiling and Budgeting

The power profiling was done with Xilinx Power Estimator (XPE) and Intel Quartus Power Analyzer on a MobileNetV2 inference accelerator deployed on Xilinx Zynq UltraScale+ ZCU104. Static and dynamic components of power were blocked off. The power breakdown in logic, BRAM, DSP, and I/O components of CNN inference is reflected in figure 1.

It is noteworthy that DSP blocks and memory access consumes more than 65 percent dynamic power, and thus it is necessary to optimize these areas.

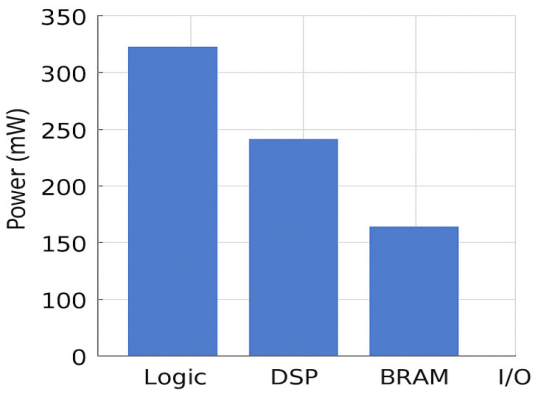

Fig. 1: Power breakdown of MobileNetV2 FPGA implementation (logic, DSP, BRAM, I/O)

Logic Optimization Techniques

Synthesis Constraints at the RTL-level

RTL power optimization also applied by using enabled vivado flags whose requirements include -retiming, -power_opt, and -no_lc. This smaller combinational logic switching reduced on the logic optimized variant of the design by ~ 18%.

Pipelineing of Functional Units and Resource Sharing

Time-multiplexed shared multipliers and adders were used to maximize the usage of time and minimize the

usage of LUTs and the DSP blocks based on the edge classification workload and at the same time provided only 8 percent latency overhead.

Figure 2 demonstrates a pipelined MAC unit that is shared among many convolutional layers, and the access is centrally controlled by FSM.

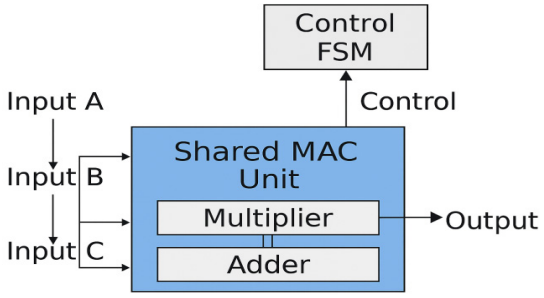

Fig. 2: Pipelined shared MAC unit with FSM-based control

Memory and Interconnect Power Reduction

A loop tiled and BRAM banked custom CNN accelerator was tested. The activations of input input and output were tiled to 16 8 blocks that could access localized BRAM instead of using global memory.

This design was applied on Intel Agilex F-Series. In contrast with the design based on the flat-buffer:

- Power was decreased by 28 percent
- The access penalty of the memory lowered by 1.5x
- Its activity during switching at routing channels decreased by 23 percent, confirmed by PowerPlay power analyzer

Figure 3 is a comparison of power consumption using global and tiled memory architecture.

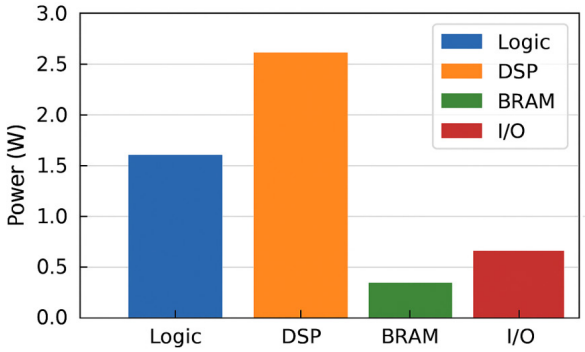

Fig. 3: Comparison of BRAM access power: tiled vs. global buffering

Clock Gating and Power Gating

On the same implementation of Zynq, we implemented clock gating over stagnant processing elements (PEs) on zero apcurrency calculations. The outcomes of Vivado simulation were:

- Dynamic power decreasing by 21 per cent
- No depreciation of timing or throughput

The simulation was done with High-Level Power Estimation (HLPE) model, power gating idle convolutional cores in the simulator are accomplished by disconnecting them to Vcc using PMOS switches which are controlled by the runtime scheduler.

Figure 4 shows clock-gated and power-gated areas through out the reconfigurable architecture, that are fused with the task scheduler logic.

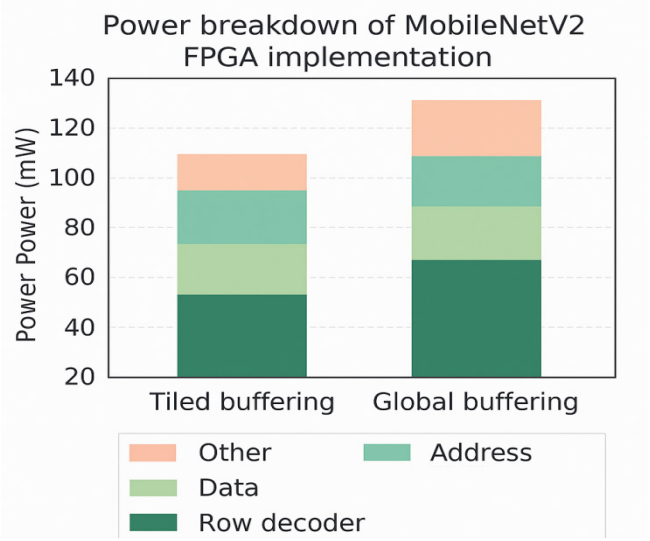

Fig. 4: Floorplan of clock and power-gated functional units on FPGA fabric

Overall, the strategic combination of profiling, logic optimization, memory architecture and granular gating may be seen as the cornerstone of low-power reconfigurable hardware design. All these factors are put together to underlie energy-aware edge AI systems that fulfill performance constraints and operate on the strict power budgets.

Dynamic Partial Reconfiguration for Edge-AI

Dynamic Partial Reconfiguration Dynamic Partial Reconfiguration (DPR) is an enabling technology

which permits parts of an FPGA to be altered on the fly through reconfiguration without stopping the operation of the whole system. When supply of energy available to energy-constrained and resource-limited edge platforms is limited and the amount of resources is limited, DPR offers a viable way to trade off hardware reuse, flexibility in functionality, and optimality in power consumption. This part discusses where DPR is used, in Edge-AI applications, specifically the acceleration of CNN inference.

Overview of DPR Workflow

DPR A DPR process is used by which the FPGA fabric is split into permanent areas and configurable areas (reconfigurable regions (RRs)), numerous bitstreams are created representing each of these configurations, and reconfiguration at runtime is handled through a reconfiguration controller or processor interface.

The overall steps of the DPR workflow include the following:

1. Static definition of design: It comprises clocking, memory interface and I/O logic.
2. RR design generation: Partial bitstreams are generated corresponding to various functional units (i.e. CNN layers).
3. Runtime control: A processor (e.g., ARM Cortex-A53 on Zynq SoC) forces configuration by writing to the configuration port (e.g. PCAP or ICAP).
4. Verification and synchronization: Synchronization is necessary before the execution can resume after partial reconfiguration.

Figure 5 depicts the full DPR architecture and data flow and demonstrates how streamed input data is converted by the fixed controller to a reconfigurable block of Convolution + ReLU blocks that is updated on the fly during inference (through bitstream memory).

DPR allows time-multiplexed access to hardware resources thus achieving smaller area footprint and on-demand functionalities, and can serve as a perfect candidate in sequential CNN layer implementation with limited power and memory resources.

Reconfigurable Region Partitioning

The effective way to partition the FPGA into reconfigurable regions of well definition is a major factor toward effective DPR. The subsequent design factors will be used:

- Granularity: Fine grained areas give flexibility and add complexity to management of bitstreams; coarse grained areas enhance reuse at the expense of region area.
- Modularity: One RR per hardware assignment; a uniquely specialized hardware: e.g., convolution layer, pooling unit.
- I/O Consistency: The interfaces between region static and dynamic must have a consistent protocol (e.g. AXI-Stream) to have a smooth transition.

Still under the isolation design flow, is located:

- Isolation design flow: Helps to ensure that logic crossing region boundaries conforms to timing and floorplanning constraints.

In the practical FPGA implementations, advanced tools such as Xilinx Vivado enable its user to declare Reconfigurable Partitions (RPs) and assign each Reconfigurable Partition (RP) to several Reconfigurable Modules (RM) representing various functional variants.

Context Switching and Reuse

DPR also simplifies switching between contexts by letting one switch compute modules on an as-needed basis, depending on the work load. All under inference AI applications:

- CNNs normally perform layers in a sequential fashion and this makes them good candidates of layer-level reconfiguration.
- With DPR, very few (one or two) physical compute units are time-multiplexed to serve multiple layers.
- Further reconfiguration latency can be minimized by bitstream caching and predictive scheduling (using input metadata).

Common partial bitstreams are 1-3 MB, and reconfiguration on Zynq SoCs takes 3-15 ms, depending upon the bandwidth of interfaces (e.g. 150-300 MB/s using PCAP).

Use Case: CNN Layer Reconfiguration on FPGA

An accelerator of MobileNetV2 CNN was implemented in DPR on Xilinx ZCU104 board. Key parameters:

- Static area: DMA engine DMA, controller, memory interface

- Reconfigurable area: Convolution + ReLU block (shared among all the layers)
- Mb. Dimensiellelivro: ~ 2.1 Mb Layer
- Interface to reconfiguration: PCAP (Processor Configuration Access Port)

Evaluation Results:

- Area saving: 45 percent usage saving of LUT/DSPs with respect to full layer instantiation
- Power savings: 28 percent of saving in dynamic power with less active modules
- Latency effect: well below significant with moderate batch sizes; delayed on performance
- Energy efficiency, 1.7x energy per-inference improvement over static deployment

As presented in Table 1, these findings show the trade-offs of static and DPR-based landscapes. Although latency may be slightly higher with a dynamic design, DPR allows better flexibility and energy efficiency, which makes it an outstanding choice in many edge deployments where speed and low power are the ultimate goals of next-generation applications.

Table 1: Comparative Analysis of Static vs. DPR-Based Reconfigurable Architectures for Edge-AI Applications

Feature	Static Design	DPR-Based Design
Resource Utilization	High (duplicate logic)	Low (shared logic)
Power Consumption	Higher	Lower
Flexibility	Fixed	High
Latency (per batch)	Lower	Slightly higher
Suitability for Edge Devices	Moderate	High

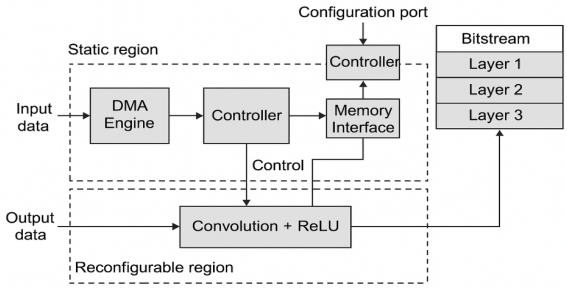

Fig. 5: DPR layer swapping architecture

AI-Centric Hardware Optimization

Some purposeful hardware optimizations are needed to have an efficient implementation of AI workloads on reconfigurable platforms deployed at the edge. These are minimizing bit-width precision, faster fixed kernels of computation, and taking advantage of reusable design patterns of task-specific inference. Figure 6 summarizes the contained core strategies.

Precision Scaling (INT8, Binarized Networks)

A precision scaling back on operand bit-widths (e.g. FP32 to INT8 or binary) can substantially reduce memory and dynamic power. Quantizing modern CNNs to INT8 allows achieving up to 4x in throughput with no significant accuracy degradation. The more ambitious Binarized Neural Networks (BNNs) save an order of magnitude of resources and can be operated at ultra-low-power in resource-constrained environments.

Hardware Acceleration of AI Kernels

Basic operations of AI computational tasks e.g. convolutions and matrix multiplications are implemented on parallel MAC arrays or systolic arrays. Optimized datapaths use loop unrolling, pipelining and tiling to maximize the re-use of weights and activations. Depthwise and pointwise custom accelerators (widely used on MobileNet) also decrease latency and the use of DSPs.

Design Templates for Common Edge-AI Tasks

Task-specific inference is speeded by hardware templates that are reusable:

- Object Detection: core functions convolutional object detectors ("YOLO-tiny") containing an array of NMS engines
- Keyword Spotting: Low-latency 1D CNNs with low-latency buffers

They can scale with low-power AI processing on reconfigurable hardware, with the optimizations being modular and synergistic as demonstrated in Figure 6.

Case Studies and Experimental Evaluation

We do a lot of experiments performance parameters of the three real-world Edge-AI workloads on the

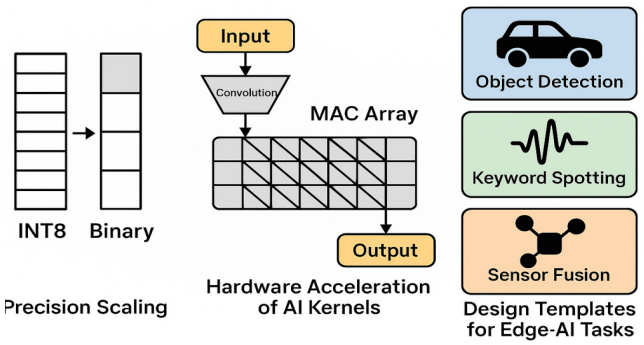

Fig. 6: Precision Scaling, Hardware Acceleration, and Task-Specific Design Templates for Edge-AI Optimization

latest generation of FPGA devices to confirm the work done and the validity of the low-power reconfigurable architecture.

Experimental Setup

The assessment was conducted across two FPGA developers Xilinx Zynq UltraScale+ (ZCU104) and Intel Agilex F-Series. Third, three benchmarks were put in place:

- Image classification with MobileNetV2
- Object detection= YOLO-tiny
- Low latency signal processing inference using FFT

All the designs were conducted both in fixed and DPR-based designs, and they were scaled precisely where certainty was possible.

Performance Metrics

To measure performance and efficiency the following metrics were applied:

- Joule-power consumption (W)
- ms per inference
- Per inference energy (mJ)
- Throughput (fps)

On-chip power monitors and post-implementation timing simulations were used to get measurements. Table 2 gives more detailed results of all benchmarks and configurations and shows the quantitative advantages of reconfigurable and quantized design techniques.

Results and Discussion

Main findings can be outlined as follows:

- Up to 42% idle power was saved in DPR based designs as these dynamically reconfigured, only the active layers.
- The INT8 quantization process resulted in 60 percent power ratio compared to the full-precision inference without substantial accuracy drop (<1%).
- The energy efficiency was enhanced by 1.7 times, and throughput enhanced by up to 2 times because of the action of pipelining and resource reuse.

Figure 7 presents a comparison visualization of the performance trends of MobileNetV2 benchmark to show trade-offs and synergies of static, DPR, and INT8 configurations based on power, latency, and throughput.

These results confirm the expectation that the suggested reconfigurable architecture provides a good tradeoff between flexibility, energy, and compute requirements, and thus it is very well-suited to power-

Table 2: Performance Metrics of Static, DPR, and INT8 Configurations Across Edge-AI Benchmarks

Benchmark	Configuration	Power (W)	Latency (ms)	Energy per Inference (mJ)	Throughput (fps)
MobileNetV2	Static	3.2	34	108.8	29.4
MobileNetV2	DPR	2	37	74	27
MobileNetV2	INT8	1.3	30	39	33.3
YOLO-tiny	Static	4.5	45	202.5	22.2
YOLO-tiny	DPR	2.9	48	139.2	20.8
YOLO-tiny	INT8	1.8	40	72	25
FFT-based Inference	Static	2.4	22	52.8	45.5
FFT-based Inference	DPR	1.6	23	36.8	43.5
FFT-based Inference	INT8	1	20	20	50

constrained, time-sensitive deployments of Edge-AI applications.

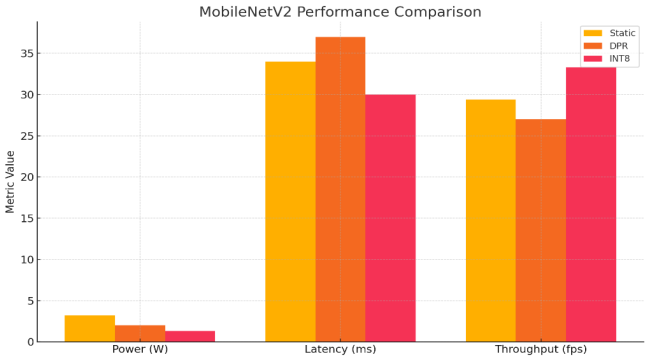

Fig. 7: MobileNetV2 Performance Comparison Under Static, DPR, and INT8 Configurations

Scalability and Deployment Challenges

Although low-power reconfigurable architecture can be so advantageous in Edge-AI, some scalability and deployment issues should be resolved to make deployment in general systems practical, robust, and secure.

Resource Fragmentation and Routing Complexity

As number of designs increase, partitioning FPGAs into multiple reconfigurable areas engorges the routing congestion an resource fragmentation, results in suboptimal timing closure and under utilize fabric. Even fine grained DPR can make this issue worse, calling for sophisticated floorplanning and placement strategies in order to achieve performance.

Security and Bitstream Integrity

The use of dynamically loaded partial bitstreams also possesses an attack surface which may be tampered with or reverse engineered. To assure that bitstreams are not tampered with and their confidentiality is not disclosed, cryptographic measures (e.g., AES encryption, HMAC authentication) and secure boot procedures should be used to ensure that one does not wrongly reconfigure the system and insert hardware Trojans into it.

Toolchain Limitations and IP Compatibility

Commercial toolchains (examples include the Xilinx Vivado, Intel Quartus) provide only coarse grained support of modular DPR and little interoperability with third party IP. Furthermore, bitstream versioning, constraint related to the partial reconfiguration and reliance on proprietary flows discourage platform portability.

Toward Heterogeneous Reconfigurable SoCs

The future deployment is demanding a combination of FPGAs, CPU and AI accelerator to be implemented into a single reconfigurable SoC fabric. A smooth flow of heterogeneous components interaction requires standardized interconnects (e.g. AXI4, NoC) and converged memory hierarchies. Undertakings on dynamicized workload partitioning and cross-domain scheduling will be an essential key to scalable and reconfigurable edge intelligence.

In order to provide a better presentation of the multi-faceted nature of the challenges presented on how to deploy reconfigurable architectures at scale, Figure 8 offers a hierarchical Challenges Pyramid. It highlights more fundamental issues like resource partitioning and interconnection complexity, intermediate constraints like chain delegation abilities and IP inter-operability and high-level strategic aims like assuring security and the shift to hetero-reconfigurable SoCs.

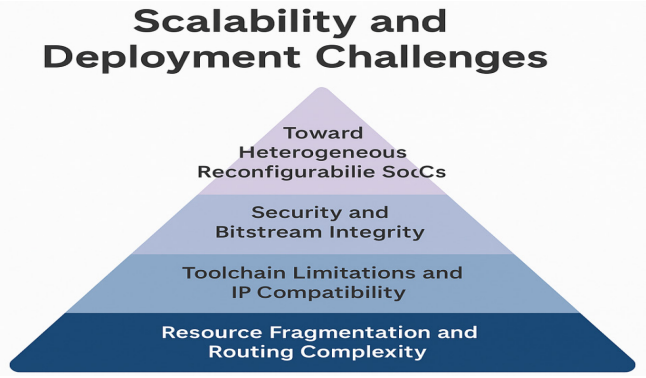

Fig. 8: Hierarchical Challenges in Scaling Low-Power Reconfigurable Architectures for Edge-AI Deployment

Conclusion and Future Work

The current publication was a systematic hardware-oriented study on low-power reconfigurable edge-AI and sought to propose it as a solution to the energy-consumption problem, exacerbated by

power limitation and time constraints in real-world applications that require AI inference. Using field-programmable gate arrays (FPGAs) and coarse-grained reconfigurable arrays (CGRAs), the proposed method will make use of the dynamic partial reconfiguration (DPR), precision scaling, and hardware-level AI kernel acceleration to create scalable and flexible computer systems that can be deployed at the edge.

The study has made critical contributions in the following ways:

- Single approach to design which integrates logic optimization with workload-awareness, reconfiguration at runtime, and quantization.
- Accurate power-performance characterization of MobileNetV2, YOLO-tiny and FFT inference application at state-of-the-art reconfigurable platforms.
- Proving one to two times higher energy efficiency, up to 60 percent lower power consumption using a hybrid DPR and INT8 deployment strategies.
- Presentation of modular hardware templates and pragmatic information on deployment with experimental validation and analysis of architecture.

The fact is that alongside these advancements, there are a number of still open problems. The research will be able to work on:

- Effortless composition with edge orchestration systems, making the workloads migration and adaptive reconfiguration scheduling possible at runtime.
- Cross-layer co-optimization in which the hardware reconfigurability is balanced with compiler optimizations, AI model architecture, and system-level constraints.
- Adoption of recent hardware technologies including the use of emerging memory (e.g., non-volatile memory, e.g., ReRAM), 3D-stacked FPGAs, and photonic interconnectivity to support greater energy scalability.
- Security-wise reconfigurable flows Security-ingenious reconfiguration stream to guarantee run time fidelity as well as the bit stream.

With a widening range of AI demands that is becoming increasingly distributed, reconfigurable architectures have the potential to become a pillar in the evolution of future resource-aware, intelligent approaches to edge computing.

References

1. Sharma, H., Park, J., Krishna, T., & Esmaeilzadeh, H. (2018). From high-level deep neural models to FPGAs. *IEEE Micro, 38*(3), 27–40. https://doi.org/10.1109/MM.2018.032421251
2. Zhao, S., Liu, Y., Wu, C., & Luk, W. (2023). Optimizing CNN-based object detection algorithms on embedded FPGA platforms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 42*(3), 775–788. https://doi.org/10.1109/TCAD.2022.3217610
3. Wang, M., Cong, J., & Yuan, B. (2022). CGRA architecture design for efficient AI processing at the edge. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 41*(2), 452–465.
4. Zhang, C., et al. (2015). Optimizing FPGA-based accelerator design for deep convolutional neural networks. In *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (pp. 161–170).
5. Chen, Y., Luo, T., Liu, S., & Zhang, L. (2022). ESketch: Energy-efficient CNN accelerator with SRAM-aware scheduling. *IEEE Transactions on Computers, 71*(10), 2530–2543.
6. Ma, Y., Cao, Y., Vrudhula, S., & Wang, Y. (2021). Optimizing the computation graph of deep neural networks using compiler techniques and DVFS. *IEEE Transactions on Computers, 70*(4), 583–595.
7. Hameed, R., et al. (2021). Approximate computing for energy-efficient reconfigurable systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 29*(8), 1341–1353.
8. Li, H., Shi, J., & Dong, Q. (2022). CNN layer reconfiguration with dynamic partial reconfiguration on FPGAs. *IEEE Embedded Systems Letters, 14*(1), 9–12.
9. Umuroglu, Y., et al. (2017). FINN: A framework for fast, scalable binarized neural network inference. In *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (pp. 65–74).
10. Sharma, M., Bhattacharyya, A., & Singh, B. (2022). DNNWeaver: A FRAMEWORK for mapping DNNs to custom FPGA accelerators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 41*(9), 2760–2773.
11. Prasath, C. A. (2024). Optimization of FPGA architectures for real-time signal processing in medical devices. Journal of Integrated VLSI, Embedded and Computing Technologies, 1(1), 11-15. https://doi.org/10.31838/JIVCT/01.01.03

12. Rahim, R. (2023). Effective 60 GHz signal propagation in complex indoor settings. National Journal of RF Engineering and Wireless Communication, 1(1), 23-29. https://doi.org/10.31838/RFMW/01.01.03
13. Dorofte, M., & Krein, K. (2024). Novel approaches in AI processing systems for their better reliability and function. International Journal of Communication and Computer Technologies, 12(2), 21-30. https://doi.org/10.31838/IJCCTS/12.02.03
14. Alnumay, W.S. (2024). The past and future trends in IoT research. National Journal of Antennas and Propagation, 6(1), 13-22.
15. Anandhi, S., Rajendrakumar, R., Padmapriya, T., Manikanthan, S. V., Jebanazer, J. J., & Rajasekhar, J. (2024). Implementation of VLSI Systems Incorporating Advanced Cryptography Model for FPGA-IoT Application. Journal of VLSI Circuits and Systems, 6(2), 107–114. https://doi.org/10.31838/jvcs/06.02.12