A Critical Analysis on FPGA–CGRA Co-Design for Energy-Efficient Edge AI Systems

Jie Du *

Houston community college, Houston, Texas, 77449, United States

* Corresponding Author Email: W215529297@student.hccs.edu

Abstract. With the advancement of artificial intelligence (AI) in Internet of Things (IoT) devices, drones, wearables, and autonomous robots, the need for real-time processing edge AI processors has grown significantly. Architectures like field-programmable gate arrays (FPGAs) and coarse-grained reconfigurable architectures (CGRAs) have advancements in a particular field, but the stringent requirements of latency, power, and performance may not be met, respectively. Existing studies typically analyze FPGA-CGRA in isolation, leaving a limited understanding of the methodology and functionality of the hybrid architecture. Therefore, this review examines how hybrid FPGA-CGRA computing addresses problems like power efficiency and flexibility and found that it could achieve 1.93x higher peak performance and 48.5% area saving compared to a singular architecture alone. However, its limitations also need to be considered, such as performance overlay and high resource consumption. This research provides important insights for designing next-generation edge computing accelerators that balance flexibility, efficiency, and real-time processing needs.

Keywords: FPGA, CGRA, Co-Design, Energy-efficient.

1. Introduction

The rapid growth of artificial intelligence on IoT devices, drones, wearables, and autonomous robots has increased the need for real-time processing to generate meaningful information and facilitate real-time task execution [1]. The reliance on cloud devices to compute and store information in a centralized cloud may cause problems like privacy leakage, high latency, and bandwidth bottlenecks. To address these issues, edge computing introduces a low-latency, local processing element that shifts data processing, storage, and computation from cloud computing into near-terminal devices [2]. To achieve such a phenomenon, AI edge processor elements require acquiring lower energy consumption, higher performance, and smaller area.

This paper is a critical review of the combination of FPGA-CGRA co-design in the application of artificial intelligence in edge devices. First, an overview of all processing architectures, including GPUs, CPUs, FPGAs, and CGRAs, will be provided. The review will then examine how hybrid architectures integrate these two paradigms to address energy efficiency and flexibility trade-offs in edge AI.

Traditional GPUs and CPUs are insufficient to meet the demands of edge AI because they are fundamentally optimized for batch-oriented, in-memory processing rather than predictable, low-latency execution on streaming data. Both architectures are also inherently power-intensive, making them unsuitable for deployment in energy-constrained edge environments. GPUs exhibit strong sensitivity to workload size; their performance degrades significantly with small batch sizes due to kernel-launch overheads and costly memory transfers [3]. CPUs, by contrast, are built as general-purpose processors and lack the degree of parallelism required for modern AI workloads [4]. Although GPUs offer substantial spatial parallelism, their absence of temporal parallelism and limited programmability further restrict their ability to satisfy the stringent latency and power requirements of edge AI systems [3].

FPGAs are a combination of programmable logic blocks, a routing network, and configurable I/O blocks that can be used to implement virtually any digital circuit or system [5]. The usage of look-up tables for configurable logic blocks makes FPGAs prone to fine-grained reconfigurability [6]. While

this structure provides high flexibility due to LUT and parallel data path (need to add more about parallel data path), it also introduces difficulties such as routing overhead and lower energy efficiency.

On the other hand, CGRAs contain a mesh of reconfigurable cells (RCs) or processing elements (PEs) that operate in the word-level, while FPGAs operate in the bit-level LUT logic [7]. This structure provides CGRA short reconfiguration times, low delay characteristics, and low power consumption compared to field programmable gate arrays [6]. Although CGRAs provide a stronger energy efficiency and domain-specific flexibility, which is efficient for fields like machine learning and artificial intelligence, difficulties, such as less flexibility, still need to be taken into consideration [6, 8]. Furthermore, limitations like difficulties in programmability and mapping still need to be taken into action. The CGRA survey highlights that the current CGRAs are still "immature in terms of programmability, productivity, and adaptivity" [8].

While many studies have examined FPGA-based accelerators and CGRA architectures, the literature has rarely provided insights on how hybrid FPGA-CGRA co-design leverages the flexibility of reconfiguration and the efficiency of complementary strengths and weaknesses. Only a small number of papers analyzed the hybrid approach, and even fewer provided a systematic review of their performance in edge AI workload. This highlights the need for a consolidated review of current progress and identifies the challenges in FPGA-CGRA co-design for energy-efficient edge AI systems.

2. Systematic Analysis of FPGA

2.1. FPGA-based Edge AI Accelerators

2.1.1 Configurable Logic Blocks

The Configurable logic block is the fundamental logic component in an FPGA, responsible for basic logic and storage functions in an application [5]. Each logic block consists of logic elements (LEs) that are connected through local routing. Every LE includes an SRAM-based k-input LUT and an optional D-flip-flop (FF) that can implement any registered Boolean logic function from k-inputs [5]. Historically, several types of basic logic blocks have been investigated, including NAND gates, Multiplexer-based structures, PAL-style wide input gates, but commercial FPGA vendors use LUT-based CLBs as they balance between flexibility and resource efficiency.

Beyond basic logic elements, modern FPGAs contain a heterogeneous mix of specialized hard blocks such as memory, multipliers, adders, and DSP blocks, etc [5]. These hard blocks offer ASIC-like performance and are used to optimize efficiency during the FPGA fabric [5, 7].

2.1.2 Routing Network

A Routing Network is used to connect logic blocks and I/O blocks together using wires and programmable switches [5]. These programmable switches consist of SRAM-controlled multiplexers that connect block outputs to surrounding block inputs [5, 9]. These two building blocks provide fine-grained flexibility within an FPGA during run-time, allowing logic blocks to interconnect in arbitrary ways, and enabling the FPGA to optimize performance, parallelism, and scalability [3, 5, 10].

2.1.3 I/O block

The input and Output block in the FPGA connects internal logic to the outside world. They are controlled by configuration bits that set their direction (input/output), electrical standard, drive strength, and timing characteristics [5, 11].

2.2. Advantages of FPGA-based edge AI Accelerator.

Compared to CGRA-based AI accelerators, FPGA-based edge AI accelerators provide several architectural advantages toward meeting strict latency requirements for modern edge AI Systems. Its high customizability and parallel processing capacities enable users to select various algorithms depending on the current situation, enabling the FPGA to increase parallelism, resulting in decreased

latency [10]. Additionally, the flexible memory hierarchy allows designers to optimize memory access patterns, causing a decrease in latency caused by data movements [12].

FPGAs also support various data types and bit widths, which directly improve the precision and performance of each memory usage, resulting in a decrease in resource usage. Additionally, in the research paper" A comparative study of FPGA and CGRA technologies in hardware acceleration for deep learning." They found out that FPGA achieves higher throughput than CGRA in computation-intensive tasks if the task requires flexible bit-level operation and dataflow control [13].

2.3. Disadvantages of FPGA-based edge AI accelerator

While FPGAs often achieve lower latency than CGRAs, the primary drawbacks, like power efficiency when compared to CGRAs, are essential to acknowledge. Because FPGAs are more versatile and include substantial redundancy due to limited on-chip memory resources, it is difficult to optimize for specific applications. This redundancy contributes to a higher energy consumption rate compared to CGRAs [13]. Furthermore, FPGA fabrics rely mostly on LUT-based logic and programmable routing, making the area of the FPGA both difficult to predict and larger than CGRAs. The result of increasing the area leads to less efficient hardware resource utilization.

3. Systematic Analysis of CGRA

3.1. CGRA-Based AI Accelerators

For decades, the rapid development of CNNs and transformers has led to high demand for specialized accelerator hardware, with coarse-grained reconfigurable arrays emerging as a promising solution. The CGRA-based AI accelerators rely on spatially distributed compute resources to achieve high throughput and energy efficiency [8]. At its core, it consists of programmable processing elements, known as PE, arranged into two-dimensional arrays connected through a reconfigurable interconnected network [9]. High-level machine learning frameworks such as TensorFlow or PyTorch serve as the front-end input, where neural network models undergo preprocessing steps, including kernel detection, loop unrolling, and loop tiling to expose parallelism and prepare the computation for spatial mapping onto the CGRAs [8, 9].

CGRA's Processing Elements are used to operate logic on instructions given, such as addition, multiplication, logic operations, and comparison. These Processing elements provide the capability for CGRA to accumulate spatial, temporal, and parallel computation. Additionally, these processing elements create a coarse-grained operational environment within the architecture, making them capable of computing-intensive kernels [8].

3.2. Advantages of CGRA-based edge AI Accelerator

One key advantage of a CGRA-based accelerator over traditional FPGA implementation is energy efficiency. According to the paper "Coarse-Grained Reconfigurable Array Architectures," CGRAs are more energy-efficient and area-conscious due to the word-wide instruction stores (ISs), register files (RFs), and interconnections, causing them to be inherently more energy and area efficient than bit-level LUT-based fabric used in FPGAs. Additionally, there are far fewer ISs in CGRA found compared to LUTs in an FPGA. This configuration causes the number of bitstreams to be much smaller, which results in less memory requirements and faster reconfiguration abilities [14]. Together, all these architectural differences significantly reduce the power consumption, area efficiency, and overall efficiency in a CGRA design.

3.3. Disadvantages of the CGRA-based edge AI accelerator

After acknowledging the advantages of the CGRA-based edge AI accelerator, a key limitation is its programmability and flexibility. Programmability is one of the most important factors for practical adoption and usability. But because of CGRA's complex architecture, it requires programmers to

manually do tasks like scheduling, placement, and routing, as mentioned in the "Coarse-Grained Reconfigurable Array Architecture" paper, the mapping, connection, and input/output ports of the ISs and RFs need to be reserved by the compiler or programmer. Additionally, unlike FPGAs, CGRAs need to be directly programmed with interconnections between VLIWs and ISs, which is not explicitly mentioned as difficult to program, but it is still extremely complex compared to an FPGA [11].

Moreover, flexibility limitations are also a concern for edge AI systems, because CGRAs are built from a fixed number of word-level processing elements, and only certain types of loops can be mapped onto them, which limits the flexibility of CGRA, as it puts more restrictions. Additionally, unlike an FPGA, a CGRA only supports a certain kind of workflow, like computing-intensive tasks, or other types of workloads, like control-heavy or bit-level manipulation tasks [11]. These architectural differences cause CGRA to obtain limitations to flexibility and be unable to successfully perform tasks like edge AI.

4. Hybrid FPGA -CGRA Architecture

After understanding the weaknesses of each architecture separately, it is essential to examine current research on the hybrid FPGA-CGRA architecture that aims to combine its complementary advantages.

Hybrid FPGA-CGRA architecture emerges from the recognition that neither FPGAs nor CGRAs alone can optimally address all edge AI requirements [8]. It integrated FPGA reconfigurability and high flexibility for arbitrary control logic and irregular operations while utilizing CGRA regions for energy efficiency acceleration of regular AI kernels. This approach creates a reconfigurable system-on-chip (SoC) that can dynamically allocate different computation patterns to the most suitable processing substrate [8, 10, 15].

4.1. Strength of Hybrid FPGA-CGRA Architecture

The Hybrid combination of FPGA-CGRA architecture has found extraordinary improvement in the performance and area saving compared to the FPGA architecture. It has been shown to improve 1.93x in peak performance and 31.1% and 48.5% maximum area saving on Intel and Xilinx [15]. These metrics demonstrate an increase in the hardware resource utilization, proving that the hybrid approach performs better than either singular architecture alone.

Beyond the performance and area benefits, energy efficiency and workload adaptability are improved compared to a single architect alone. By partitioning computation such that CGRA regions handle regular, compute-intensive kernels while the FPGA manages irregular control, the systems minimize the energy required for consumption where it matters the most [8, 10, 15]. This enables workflow distributed to allocated architecture to maintain high utilization across diverse edge AI applications.

4.2. Disadvantages of Hybrid FPGA-CGRA Architecture

While the Hybrid FPGA-CGRA architecture accommodates multiple advantages over singular FPGA and CGRA architectures, its limitations must also be acknowledged. Performance overlay and high resource consumption are major implications for operating a Hybrid architecture in Edge AI systems, because the hybrid design creates an additional layer on top of the FPGA, causing kernel mapping to pass through multiple overlays and operations such as overlay instruction memory, PE ALU operations, overlay operation, overlay interconnect routing rather than directly to hardware components like LUTs/DSPs [10, 16]. These overlays will inevitably introduce a performance and resource consumption penalty.

Moreover, the paper "Automatic Nested Loop Acceleration on FPGA Using Soft CGRA Overlay" said that when navigating the complex architecture and compilation parameters created by the hybrid FPGA CGRA, it is a slow and non-trivial process. This means, during the design process, it is very

hard and time-consuming for humans to manually configure all the architecture and parameters, such as I/O depth, data width, and mapping constraints [16]. Additionally, the complexity of the design rises if modifications are made to any parameters, as it would affect multiple aspects of the accelerator, making the design space highly challenging.

5. Conclusion

This paper has systematically examined the advantages and weaknesses of FPGA, CGRA, and hybrid architectures for energy-efficient edge AI systems. Even though both FPGA and CGRA have advantages in their respective field, such as FPGA's reconfigurability and its support for diverse workloads, and CGRA's energy efficiency due to its architecture. It is essential to acknowledge the benefits of hybrid FPGA-CGRA toward energy-efficient edge AI systems.

By combining the advantages of both architectures, it can receive 1.93x peak performance improvement and 48.5% area saving. However, challenges remain, like performance overlay and high resource consumption. Therefore, future research must address these problems and propose solutions like advancements in mappers/schedulers or automatic design frameworks.

References

[1] Surianarayanan C, Lawrence J J, Chelliah P R, et al. A survey on optimization techniques for edge artificial intelligence (AI). Sensors, 2023, 23(3): 1279.

[2] Hua H, Li Y, Wang T, et al. Edge computing with artificial intelligence: A machine learning perspective. ACM Computing Surveys, 2023, 55(9): 1-35.

[3] Biookaghazadeh S, Zhao M, Ren F. Are {FPGAs} suitable for edge computing. USENIX workshop on hot topics in edge computing (HotEdge 18). 2018.

[4] Öberg J. An efficiency comparison of NPU, CPU, and GPU when executing neural networks. KTH Royal Institute of Technology, 2023.

[5] Farooq U, Marrakchi Z, Mehrez H. FPGA architectures: An overview. Tree-Based Heterogeneous FPGA Architectures: Application Specific Exploration and Optimization, 2012: 7-48.

[6] Biswas S, et al. FPGA based real-time object tracking system. International Journal of Computer Applications, 2014, 98(14): 15-21.

[7] Guo K, Zeng S, Yu J, et al. A survey of FPGA-based neural network accelerator. arXiv preprint arXiv:1712.08934, 2017.

[8] Podobas A, Sano K, Matsuoka S. A survey on coarse-grained reconfigurable architectures from a performance perspective. IEEE Access, 2020, 8: 146719-146743.

[9] Lahti S, Hämäläinen T D. High-level Synthesis for FPGAs-A Hardware Engineer's Perspective. IEEE Access, 2025.

[10] Sunny C, Das S, Martin K J M, et al. Standalone Nested Loop Acceleration on CGRAs for Signal Processing Applications. International Workshop on Design and Architecture for Signal and Image Processing. Cham: Springer Nature Switzerland, 2024: 83-95.

[11] Kong X, Huang Y, Zhu J, et al. Mapzero: Mapping for coarse-grained reconfigurable architectures with reinforcement learning and monte-carlo tree search. Proceedings of the 50th Annual International Symposium on Computer Architecture. 2023: 1-14.

[12] Yan F, Koch A, Sinnen O. A survey on FPGA-based accelerator for ML models. arXiv preprint arXiv:2412.15666, 2024.

[13] Wu R, Liu B, Fu P, et al. An efficient lightweight CNN acceleration architecture for edge computing based-on FPGA. Applied Intelligence, 2023, 53(11): 13867-13881.

[14] Albuquerque Ferreira G. Designing an instruction set based coarse grain accelerator. Universidade do Porto (Portugal), 2024.

[15] Huang B, Huan Y, Chu H, et al. IECA: An in-execution configuration CNN accelerator with 30.55 GOPS/mm² area efficiency. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68(11): 4672-4685.

[16] Melchert J, Zhang K, Mei Y, et al. Canal: A flexible interconnect generator for coarse-grained reconfigurable arrays. IEEE Computer Architecture Letters, 2023, 22(1): 45-48.