FPGA-Based Deep Learning Acceleration: Analysis of Dataflow and Tiling Strategies

Xingke Wang*

School of Precision Instrument and Opto-Electronics Engineering, Tianjin University, Tianjin, 300072, China

*Corresponding author: 13623491111wxk@gmail.com

Abstract. With the fast development of deep learning models, especially Convolutional Neural Networks (CNNs) and Large Language Models (LLMs), Central Processing Units (CPUs) and Graphics Processing Units (GPUs) struggle to meet the high-performance and low-power requirements for deep learning acceleration. As a kind of reconfigurable hardware, Field-Programmable Gate Arrays (FPGAs) have exhibited great potentials for deep learning acceleration because of its efficient parallel computing and flexible nature. This paper is on the design and optimization strategies for FPGA based Deep Learning (DL) accelerators, specifically dataflow and tiling strategies. FPGA accelerators applied to deep learning architecture such as CNNs or Transformers; also consider the effects of different data flow strategy on the performance. This paper also presents ongoing bottlenecks like memory bandwidth, computational resource allocation and maturity of toolchain and future research directions like dynamic data flow, partial reconfiguration and heterogeneous computing platform integration. Lastly, this paper give an outlook for the future prospects of FPGAs in deep learning acceleration and offer some suggestions for further optimization.

Keywords: Performance Optimization; FPGA Accelerators; Tiling Strategies; CNNs.

1. Introduction

As Deep Neural Networks (DNNs) are increasingly being used across many areas, the demand for hardware acceleration is rising, especially when it comes to low-power, high-throughput inference. Using reconfigurability and parallel computing of FPGAs to realize the application of deep learning inference has a good performance [1].

But there are some problems too, like not having enough room for stuff, places where information gets stuck waiting to move on and using those computing bits in not-so-smart ways. In this paper, the author examine how different dataflow mapping and tiling strategies can be used to solve these problems. Dataflow mapping decides what kind of computing jobs will be done and where to put data away, tiling strategy makes sure there is enough room in memory by breaking big calculations into smaller pieces that fit in better.

In recent years, many FPGA designs have improved their performance by improving the dataflow and tiling. Take FINN and Flight LLM accelerators for example, and we compare their designs and distinct dataflow applications in low-precision and sparse computation.

The paper gives an overall account and analysis about the usage of dataflow and tiling strategy used by FPGA-DNN accelerators in last five years. It mainly focuses on decreasing bottlenecks due to memory bandwidth as well as increasing the resource utilization of the available computing power.

2. Balancing elbow force analysis

2.1. FPGA Architecture Basics

FPGAs have reconfigurable logic resources, which include Look - Up Tables (LUTs), flip - flops, and Digital Signal Processing (DSP) units. These can then be programmed to create any type of logic circuit, providing both parallelism and adaptability for various different tasks [1].

FPGAs have different types of memories like Distributed Random - Access Memory (RAM), Block RAM (BRAM) and Ultra RAM. In the devices of now, HBM, high - band with memory, is used too. and these parts of the building makes FPGAs able to solve math problems as big as multiplication but using much less power [2].

FPGA's ability to be flexible, which can help tailor the data path and memory hierarchy for a certain DNN. Unlike the fixed-function ASICs, FPGAs' ability to be reconfigured makes them very suitable for a broad set of AI workloads. Such ability to adapt makes FPGAs suitable for edge devices and data centers.

2.2. DNN Computation Models: CNN and Transformer

DNN models like CNNs and Transformers have different ways of doing work. CNNs have convolution operations, which are performed on input feature maps through sliding multi-dimensional filters. The convolution layer does multiply accumulate operations, and each weight will be used a lot of times in many spatial positions so it has a lot of data reuse [2]. It also makes them appropriate for certain specializations, such as buffering data in on-chip memory to make use of spatial locality, or organizing the MAC units in a systolic array for parallelism.

On the contrary, Transformer models, particularly those used in NLP and Computer Vision, rely on attention, which involves matrix-intensive operations like multiplying the query and key matrices for attention scores. CNNs have much more localized data access patterns compared to Transformers which makes the computation irregular. High dimensional tensor algebra in the Transformers is memory intensive. Transformers on FPGAs can be optimized by tiling large matrix operations, using systolic arrays, often with quantization and sparsity techniques to fit the work to the parallelism of the FPGA.

2.3. Dataflow Mapping Strategies

Dataflow mapping is a fundamental concept in FPGA accelerators, as it dictates how neural network computations are mapped onto hardware resources. Specifically, dataflow mapping defines the processing methods of data such as weights, activation values, and partial sums within Processing Elements (PEs), as well as the flow of data within the architecture.

Typical dataflow strategies include Weight Stationary (WS), Output Stationary (OS), and Row Stationary (RS). In the WS strategy, the weights are stored in the local buffer of PEs, which reduces the access to the off-chip memory. OS strategy: PEs keep the partial sums in a stationary state to minimize the write-back overhead of intermediate results. The RS strategy is first introduced by the Everiss accelerator. It tries to balance the reuse of weights and that of activations, it retains the "rows" of activations in PEs and applies weights [3].

These strategies all have to make do-possibilities in different aspects like memory usage, repeating the use of data and implementation level. In more recent research, hybrid dataflow strategies are used in the design, or the dataflow is adapted to the computational characteristics of individual network layers. Things like MAESTRO help look at and improve how scholars match things up so that the data gets reused well, and doesn't read a lot from the place outside the chip (off-chip memory access). This is necessary for improving the performance and energy efficiency of FPGA - based neural network accelerators [4].

2.4. Tiling/Blocking Strategy and Data Reuse

Tiling strategy is important for DNN accelerators, it aims to solve the on-chip memory limitation of FPGAs. The key is to split large tensors into smaller sub-tensors, called "tiles", and process them sequentially or in a pipeline. Each tile is loaded into the on-chip memory such as BRAM, URAM, and then the next tile is loaded when the previous one is finished. It can maximize the use of data in on-chip memory and reduce the frequency of off-chip memory access. Double buffering is also used for memory latencies. When doing some calculation on one tile, the other tile can already be downloaded from an external memory cache to a free memory buffer. Once the current tile's

computation is completed, the accelerator immediately switches to the newly loaded data. The tile size must balance on-chip memory capacity and processing parallelism; if the tile is too large, it cannot fit into the memory; if it is too small, frequent data loading will reduce performance. Multi-level tiling further enhances data reuse by partitioning data among different memory hierarchy to optimize the use of memory hierarchy and ensure minimal data transfer across memory boundaries.

3. Literature Analysis and Frontier Exploration

3.1. Dataflow and Tiling Strategies in FPGA DNN Acceleration

In the past five years, researchers have devoted significant effort to optimizing the dataflow and tiling strategies of FPGA-based DNN accelerators. “Dataflow” refers to the order and manner in which data (weights, activations, partial results) is moved and reused on-chip, while “tiling” refers to partitioning large tensors into smaller blocks that fit into on-chip memory levels [1]. The choice of dataflow is critical: common paradigms include WS, where each PE keeps filter weights locally while input data streams through; OS, where partial sums for an output remain in registers until the output is complete; RS, which balances reuse of weights and activations; and No-Local-Reuse (NLR), a streaming approach with minimal on-chip buffering. Recent FPGA designs often adopt or hybridize these canonical dataflows to maximize data reuse and minimize off-chip memory accesses. For example, weight-stationary schemes avoid re-fetching weights from DRAM, which is beneficial for convolutional layers with high weight reuse. Output-stationary designs keep partial sums on-chip to reduce writing intermediate results back to memory. Row-stationary dataflow, though initially an ASIC concept, has inspired FPGA implementations that try to maximize both weight and activation locality across 2D convolution windows. The optimization focus for dataflow is to exploit on-chip BRAM/URAM to hold frequently reused data, thereby improving throughput per memory bandwidth.

In tandem with dataflow, multi-level tiling strategies are employed to map large DNN layers onto the hierarchical memories of FPGAs. Due to limited on-chip memory, input activations, weights, and outputs are tiled into smaller blocks that can be loaded into faster memory and processed in chunks. Effective tiling often goes hand-in-hand with double buffering to hide DRAM latency by overlapping data transfer with computation [5]. For example, an accelerator might split a convolution’s input feature map into tiles that fit in BRAM; while one tile is being processed by the PE array, the next tile is concurrently fetched into an alternate buffer. Researchers have created analytical tools like MAESTRO and Time loop for exploring the dataflow and tiling configuration space systematically. They can model the effect of different loop tiling and unrolling choices on performance, resources and data reuse. From a short comparison, it can be observed that MAESTRO uses an explicit data-centric DSL to specify the spatial/temporal reuse, while Time loop enumerates the loop nests and searches for the best tiling automatically. Both have been used to help FPGA designers pick the best data movement plans. In summary, technical route of recent FPGA DNN accelerators: select an appropriate dataflow (for maximized on-chip data reuse), determine a tiling scheme at each level of the memory hierarchy (fit layer data to limited on-chip memory with minimum performance penalty). focuses on throughput vs. resources, fine tile reduces on chip mem (decrease reuse, hits), while coarse increase but need BRAM/external mem BW [5]. So, designers carefully size tiles per layer to get close to the sweet spot of high performance with good resource efficiency.

3.2. Comparative Performance and Resource Efficiency

FPGA based DNN accelerators are analyzed through the lenses of through put, latency, resources used, energy efficiency. Different data flows and design strategies can lead to significant differences in these metrics. The following summarizes several representative FPGA accelerators from recent literature, listing their hardware platforms, computational precision, architectural highlights, achieved throughput, resource occupation, and energy efficiency, etc. For instance, the FINN accelerator targeting binarized neural networks achieved about 5 TOPS on a small 28nm Zynq-7020 device by streaming data through a layer-pipelined architecture. Notably, FINN uses only LUTs by leveraging

binary operations, reaching ~90% LUT utilization. Its successor FINN-R extended this to quantized networks on a larger FPGA, automating the unrolling/tiling for each layer [6]. FINN-R demonstrated up to 50 TOPS for 1-bit networks at 75% LUT utilization, and an energy efficiency around 100 GOPS/W. In contrast, accelerators for large models like Transformers exploit high-end FPGAs and advanced features: Flight LLM on an Alveo U280 sustains ~55 token/s throughput on a 7-billion-parameter language model using 8-bit sparsity-aware computing, with an impressive energy efficiency of ~300–400 GOPS/W. Another state-of-the-art design on Xilinx's 7nm Versal ACAP, SSR, uses the on-chip AI Engine array plus programmable logic to reach up to 26.7 TOPS throughput and 246–453 GOPS/W energy efficiency. These comparisons illustrate a trade-off: smaller, low-precision networks can achieve extreme throughput on low-cost devices, whereas larger models require newer FPGAs to deliver high performance within reasonable power envelopes. The resource efficiency tends to improve when exploiting low precision or sparsity, as seen by the DSP-free FINN achieving competitive throughput, and Flight LLM and SSR using sparsity and AI Engines to boost ops/W significantly.

Precision and dataflow settings can completely change how well something works. FINN's binary network approach gets high throughput-per-resource by avoiding multipliers, but designs like Flight LLM and SSR take the other end: very large models, with much more complicated dataflows, and much more total resources. Directly compare the energy efficiency with today's best GPUs to reveal FPGAs special advantages. FQ-BERT stands out, outperforming a NVIDIA K80 GPU by over 12 times performance/watt [7]. FTRANS framework for Transformers delivered about 8.8x higher energy efficient model through model compression, and a custom architecture for FPGAs [8]. Multi-FPGA systems such as DFX achieved around 4×GPU energy efficiency on GPT-2 text generation by distributing the work across FPGAs and using FPGA-friendly dataflows [9]. And more flexible accelerators such as Flex Run showed nearly 2.7x better perf/watt for NLP, by using resource allocation that was more flexible and fitting of the model's structure [10]. In general, these comparisons demonstrate that FPGAs are best at achieving superior energy efficiency for DNN inference-often 1-2 orders of magnitude more efficient than CPUs and 2-3 times more efficient than GPUs-especially when the accelerator is very specialized to the network. But those gains need carefully working on algorithms and hardware to fully use FPGA resources.

3.3. Innovations in Recent FPGA DNN Accelerators

Research Methodologies and Design Strategies: Over 2020 - 2025, many prominent studies have suggested new ways to make FPGAs perform better as accelerators for CNNs and Transformers. One of the most obvious directions is model compression, low-precision computing. FTRANS is a framework which introduces an FPGA-accelerated transformer that uses block-circulant weight matrices to compress the model with little accuracy loss, combined with a specialized architecture [8]. By compressing Transformer weights up to 16× and scheduling computation carefully, FTRANS achieved a 27× speedup and 81× energy efficiency gain over CPU, and about 8.8×higher energy efficiency than a GPU. In 2021, FQ-BERT extended the low-precision approach to Transformers by demonstrating a fully quantized BERT inference on FPGA [7]. FQ-BERT quantized not only weights and activations but also SoftMax and layer norm operations to 8-bit or less, and implemented a custom accelerator with bit-level configurable multipliers.7.9 times model size, 3.18fps/W, 12.7×higher perf/W than Intel i7 CPU, 28.9× higher perf/W than NVIDIA K80 GPU. These works show that aggressive quantization and structured sparsity can be effective on FPGAs, which have custom data paths with bit-level parallelism and no wasted computation on zeros.

Another research focus is enhancing dataflow and memory efficiency for CNN accelerators. Petrica et al. proposed the Frequency Compensated Memory Packing (FCMP) method for alleviating on-chip memory fragmentation in custom dataflow CNN accelerators [11]. They have put the weight into BRAM, they also changed the memory format. It could save 30% of BRAM while the throughput could stay the same. This allowed us to port a quantized ResNet-50 accelerator running on a bigger FPGA over to the smaller FPGA (U280) without much reduction in performance. Also in 2020, Yan

et al. introduced Atheena, which explored a spectral domain approach to CNN acceleration [12]. Atheena does convolutions via FFT-based operations, that is to say in the frequency domain, showing a flexible dataflow which reached low-latency inference on image-processing CNNs. These works indicate that it is more than just raw computing, and that how data are stored and moved, like memory packing and frequency-domain operations to make convolution cheaper, is key for FPGA efficiency. In addition, researchers also developed some automation framework to support those optimizations: Huang et al., DNN Explorer, which is a design space exploration tool that balances the layer-wise pipeline and the reusable compute units, which can automatically find an optimized architecture for a given DNN. DNN Explorer's paradigm outperformed prior layer-pipelined approach on deep CNNs by 4.2x, and improved DSP efficiency over both academic and commercial designs by 2-4x [13]. Such tools accelerate the design process by navigating the complex trade-offs in tiling, parallelism, and resource allocation.

For Transformer and NLP accelerators, researchers have tackled the challenges of sequence processing and model size [14]. Besides FTRANS and FQ-BERT mentioned above, another notable work is DFX-a multi-FPGA system for low-latency text generation. DFX partitions a GPT-2 model across four FPGA boards, using model parallelism and a custom instruction set to coordinate the devices [9]. It specifically addresses the sequential nature of autoregressive generation (word-by-word output) by splitting the workload and keeping all intermediate data on HBM-enabled FPGAs. DFX achieved a 5.58× speedup and 3.99× energy efficiency gain over a 4-GPU (V100) setup for end-to-end GPT-2 inference. This demonstrated FPGAs' ability to deliver better scalability and latency in datacenter NLP tasks. In 2023, Flex Run presented a modular FPGA accelerator for diverse NLP models [10]. Instead of targeting one network, Flex Run identifies common building blocks (e.g. attention, feed-forward, embedding) and includes them in a reconfigurable pipeline. It performs design space exploration to tailor the architecture to each model and can reconfigure at runtime accordingly. Flex Run's versatility allowed it to run both BERT and GPT-2 efficiently, outperforming a prior state-of-art FPGA design by up to 2.7× and even a Nvidia V100 GPU by ~2.7× in throughput on these models. Finally, the year 2024 saw accelerators targeting Large Language Models (LLMs): Flight LLM combined 8-bit weight sparsity (pruning less important weights) with a "sparse DSP chain" architecture to handle LLM workloads on FPGAs. It also introduced an "always-on-chip decode" scheme to maximize memory bandwidth usage and a length-adaptive compilation to manage long sequences [15]. On a Xilinx VU9P and the new Versal VHK158, Flight LLM was able to surpass a V100S GPU in energy efficiency by 6× and even modestly beat an NVIDIA A100 in throughput (1.2×) for batch-1 inference.

Similarly, Spatial-Sequential Reconfigurable by Zhuang et al. targeted transformer acceleration on Versal ACAP, using nearly all 400 AI Engines in parallel for matrix multiply while sequencing tasks to balance throughput and latency. SSR's hybrid spatial-sequential design yielded a programmable trade-off between high throughput and low latency, which is critical for different deployment needs. Collectively, these innovations have pushed the frontier of FPGA-based DNN acceleration: from compressing and quantizing models, optimizing dataflows, scaling across multiple FPGAs, to new architectures for ever-larger models [16]. Each work brought its own innovation, targeting some bottleneck-memory bandwidth, or transformers' irregular computational patterns, or even how researchers can get a better productivity in design-so as to push forward the state-of-the-art in this very fast-moving area [17].

3.4. Gaps, Challenges and Future Directions

There are still several open challenges even with the rapid development in FPGA-based DNN acceleration, which can be regarded as the directions for future research. One is partial reconfiguration and adaptivity: FPGAs can be reconfigured during runtime unlike other accelerators, but very few take advantage of this to adapt to different DNN layers or models on-the-fly. Future designs could use hybrid dataflows that choose the mode to be weight-stationary or output-stationary on a per-layer basis, based on shape or sparsity.

When paired with fast partial reconfiguration, the hardware could have potentially re-optimized itself by layer for efficiency, but this adds complexity and reconfiguration overhead. Another challenge is the design productivity for quickly changing models [18]. It implies future domain specific compiler that takes high level network description and automatically performs the optimal dataflow mapping, tiling and scheduling on those many levels. like MAESTRO / Timeloop to have unified code + hardware generation flows across CNN/ RNN/Transformer etc. architectures which makes the use of FPGAs a lot more straightforward [19]. Also model flexibility: FPGA needs new designs that can flexibly handle sparse or dynamic NN (dynamic sequence length, conditionally computation graph) with the low hardware overhead to support those new models like GShard etc [20]. There's also a research push for FPGAs for training, not just inference, but training comes with precision and on-chip memory for gradients challenges. hardware trends: 3D stacked memories (e.g. HBM2E, HBM3) and FPGAs with AI cores offer new possibilities as well as headaches: working out how to best take advantage of them is non-trivial. To conclude, FPGAs have achieved impressive performance gains for deep learning inference; but the frontier of FPGA-accelerated DL is in runtime adaptability, automated design, and handling increasingly large, irregular models [21]. But these challenges can be viewed as potential chances as well; by tackling them, future research can continue to reduce the difference between flexible FPGA solutions and requirements of AI models of the next generation.

4. Conclusion

Development in FPGA Based DL Accelerators has seen significant advancement but many challenges are still present. There are resource limitations, there are limits to how much bandwidth the memory can have, and then there's just trying to figure out how to design an FPGA. Even though there are these obstacles, the improvements in FPGA architecture, such as HBM, are removing some of the limits.

Transformer models and LLMs have created new demands on FPGA accelerators. These models that have large matrix multiplication inside multi-head self-attention are demanding for both the compute and memory subsystems. In the recent FPGAs, ProTEA and HPTA, have made huge improvement in speedup and energy efficiency by using optimized tiling strategies and memory subsystem. They are much faster and more energy efficient than GPUs.

Dynamic dataflow control, and reconfigurable FPGA arrays represent some of the more interesting and promising means for making things versatile. These methods give FPGAs the ability to change to fit various workloads that may be needed at any moment. It could transform how FPGA accelerators are used in many different kinds of computer environments.

Coarse-grained reconfigurability and partial reconfiguration are particularly expected to play considerable roles in such a transformation. They make it feasible for FPGAs to be deployed in a flexible and efficient way in both cloud-based and edge-device applications.

In a word, FPGAs still be the driver of the AI system. This is even more so in heterogeneous architectures working alongside CPUs, GPUs, and ASICs. Although there are still some challenges to be faced, it seems that the combination of the FPGAs' reconfigurability, their high-performance ability, and the specialized hardware support will push the envelope on what can be achieved with deep learning. This is particularly evident in systems that prioritize energy efficiency and adaptability.

References

[1] Li R. Dataflow & Tiling Strategies in Edge-AI FPGA Accelerators: A Comprehensive Literature Review. arXiv preprint arXiv:2505.08992, 2025.

[2] Jiang J, Zhou Y, Gong Y, et al. FPGA-based Acceleration for Convolutional Neural Networks: A Comprehensive Review. arXiv preprint arXiv:2505.13461, 2025.

[3] Rati G, Costa R, Ishikawa L. High-Performance FPGA Acceleration for Transformer-Based Models. 2025.

[4] Domagała Ł, van Amstel D, Rastello F. Generalized cache tiling for dataflow programs. ACM SIGPLAN Notices, 2016, 51(5): 52-61.

[5] Yi Q, Sun H, Fujita M. FPGA based accelerator for neural networks computation with flexible pipelining. arXiv preprint arXiv:2112.15443, 2021.

[6] Gao L, Luo Z, Wang L. Convolutional Neural Network Acceleration Techniques Based on FPGA Platforms: Principles, Methods, and Challenges. Information, 2025, 16(10): 914.

[7] Liu Z, Li G, Cheng J. Hardware acceleration of fully quantized bert for efficient natural language processing. 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021: 513-516.

[8] Li B, Pandey S, Fang H, et al. Ftrans: energy-efficient acceleration of transformers using fpga. Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design. 2020: 175-180.

[9] Hong S, Moon S, Kim J, et al. Dfx: A low-latency multi-fpga appliance for accelerating transformer-based text generation. 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2022: 616-630.

[10] Hur S, Na S, Kwon D, et al. A fast and flexible FPGA-based accelerator for natural language processing neural networks. ACM Transactions on Architecture and Code Optimization, 2023, 20(1): 1-24.

[11] Petrica L, Alonso T, Kroes M, et al. Memory-efficient dataflow inference for deep CNNs on FPGA. 2020 International Conference on Field-Programmable Technology (ICFPT). IEEE, 2020: 48-55.

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

[13] Zhang X, Ye H, Wang J, et al. DNNExplorer: a framework for modeling and exploring a novel paradigm of FPGA-based DNN accelerator. Proceedings of the 39th International Conference on Computer-Aided Design. 2020: 1-9.

[14] Du C, Wen Q, Wei Z, et al. Energy efficient spike transformer accelerator at the edge. Intelligent Marine Technology and Systems, 2024, 2(1): 24.

[15] Zeng S, Liu J, Dai G, et al. Flightllm: Efficient large language model inference with a complete mapping flow on fpgas. Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays. 2024: 223-234.

[16] Zhao S, Gao S, Wang R, et al. Acceleration and implementation of convolutional neural networks based on FPGA. Digital Signal Processing, 2023, 141: 104188.

[17] Rati G, Costa R, Ishikawa L. High-Performance FPGA Acceleration for Transformer-Based Models. IEEE, 2025.

[18] Liu Z, Yin P, Ren Z. An efficient fpga-based accelerator for swin transformer. arXiv preprint arXiv:2308.13922, 2023.

[19] Zeng S, Liu J, Dai G, et al. Flightllm: Efficient large language model inference with a complete mapping flow on fpgas. Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays. 2024: 223-234.

[20] Wojcicki F, Que Z, Tapper A D, et al. Accelerating transformer neural networks on fpgas for high energy physics experiments. 2022 International Conference on Field-Programmable Technology (ICFPT). IEEE, 2022: 1-8.

[21] Nechi A, Groth L, Mulhem S, et al. Fpga-based deep learning inference accelerators: Where are we standing?. ACM Transactions on Reconfigurable Technology and Systems, 2023, 16(4): 1-32.