Hardware Architectures of FPGA-based Accelerators for Convolutional Neural Networks

Yilin Lai

School of Electronic Science & Engineering, Southeast University, Nanjing, 210096, China

213202783@seu.edu.cn

Abstract. Convolutional Neural Networks (CNNs) have been widely used in Artificial Intelligence applications due to their unsupervised feature, which automatically identifies relevant features without human intervention. Outperforming power-hungry GPUs and inflexible ASICs in lightweight CNNs, FPGAs serve as a promising platform on balancing peak performance, energy efficiency and flexibility. In the last decade, several frameworks have been proposed to optimize the global performance of CNN on hardware platforms. This paper presents a survey on hardware architectures generated by various software frameworks designed for mapping CNN on FPGAs. Classic architectural cases of the streaming architecture and the single computation engine from traditional CNN-specific processors to end-to-end mapping using High Level Synthesis (HLS) tools which emerged in recent years are carefully analyzed in a sequential order. Moreover, adaptability of existing frameworks to upcoming challenges and future directions of FPGA-based CNN accelerators are identified, providing an in-depth evaluation on the topic of hardware architectures of FPGA-based CNN accelerators.

Keywords: FPGA, Convolutional Neural Network, Hardware Acceleration.

1. Introduction

Convolutional Neural Networks (CNN) have been widely employed in the field of deep learning, mainly due to its unsupervised feature which allows the network automatically identifies the relevant features without human intervention [1].

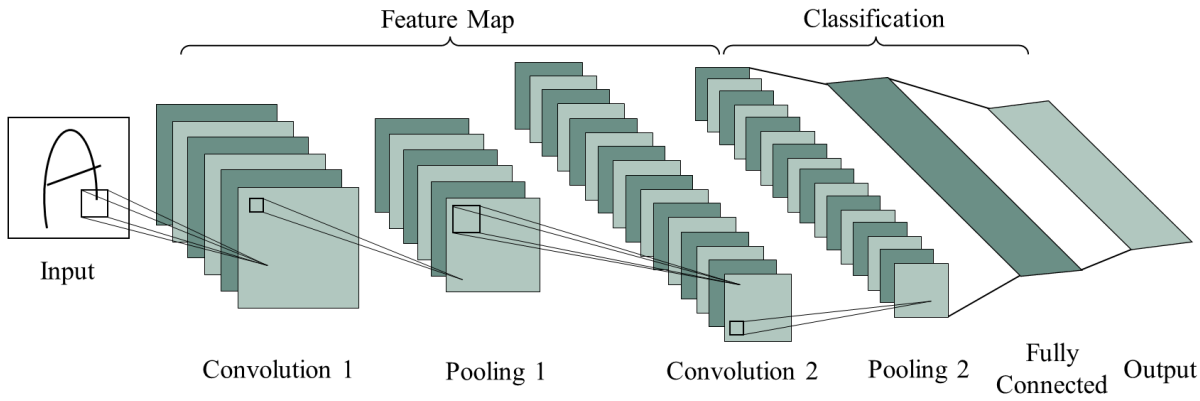


Figure 1. Illustration of the typical framework of a CNN model

A CNN model generally consists of multiple layers, namely the convolutional layer, pooling and nonlinear layer, and fully connected layer. Fig. 1 depicts the typical framework of a CNN model. Serving as the most significant component of the architecture, the convolutional layer generates output feature maps by conducting convolutional operations through a series of convolutional kernels. The pooling and nonlinear layer is responsible for sub-sampling of feature maps, shrinking bulky feature maps into small-size feature maps. Finally, non-linear activation function performs a transformation of the input, allowing the fully connected layer to conduct a classification based on previously extracted features.

While having certain merits such as high accuracy and unsupervised identification of relevant features, the increasing complexity of CNN also places greater demands on the performance of its hardware execution platform. With rapid progress in machine learning frameworks, more

heterogenous computational resource can be used to optimize the throughput, latency and minimum task power consumption. It has been well proved that FPGAs outperform other platforms such as GPUs and ASICs greatly on balancing peak performance, energy efficiency and flexibility, especially in lightweight CNNs [2].

On hardware accelerators, CNN layers can be considered as multi-dimensional loops which need three levels of adjustments, namely ordering, tilting and scheduling through space and time. Decisions taken during the process are referred to as mapping [3]. To achieve a better performance and optimize the design flow, researchers gradually abandoned using a fixed mapping for each layer in favor of a more dynamic approach by demonstrating the benefit of optimizing per-layer mappings. In recent years, mapping methodologies of FPGA-based CNN accelerators has rapidly iterated, evolving from CNN-specific processors to end-to-end mapping using High-Level Synthesis (HLS) into a customized hardware implementation.

Albeit the number of review articles published in the past researching on similar topics, there is still a lack of review dedicated specifically to analyzing hardware architectures generated by various mapping methodologies, as well as upcoming challenges and opportunities. This paper presents a systematic review on hardware architectures generated by various frameworks for mapping CNN on FPGAs. For this work, this article discusses the classic architectural cases of the streaming architecture and the single computation engine from traditional CNN-specific processors to end-to-end mapping using High Level Synthesis (HLS) tools in a sequential order. Table 1 lists the frameworks that are discussed in this paper. While there often exists multiple versions of the discussed frameworks, this article only discusses the latest revised version. Moreover, this paper summarizes the adaptability of existing frameworks to current challenges and offers future directions on the frameworks for mapping CNN on FPGAs.

Table 1. Frameworks for mapping CNN to FPGA

	CNN-Specific Processors	End-to-End Mapping
Streaming Architecture	Uni-OPU	FINN fpgaConvNet HSL4ML
Single Computation Engine	Light-OPU	Angle-Eye GAMMA unzipFPGA

2. CNN-Specific Processors

While lightweight CNN effectively shrinks down the model size, its resulting lightweight operations are not well compatible with conventional FPGA-based CNN accelerators. Evolving from the general processing purpose, embedded domain specific processors consist of algorithm-specific cores either by programming or configuration, tailoring a potential platform into application-specific requirements. Adopting different design methodologies, Uni-OPU [4] and Light-OPU [2] are typical representatives in this field.

2.1. Uni-OPU

Uni-OPU focus on optimizing specific hardware acceleration modules. Based on a unified data flow pattern and address generation mechanism, Uni-OPU achieves acceleration for both conventional convolutional networks and transposed convolutional networks by deconstructing and reformulating convolutional operations into streaming parallel vector multiplication.

2.2. Light-OPU

Proposed by the same team as Uni-OPU, Light-OPU proposed a FPGA-based general accelerator for a class of network models with high performance and power efficiency. Light-OPU decomposes

and reformulates lightweight operations by employing the single computation engine architecture. The accelerator with a specific set of instructions considers the utilization of resources between lightweight operations and conventional convolution operations among several major computation engines, greatly improves the runtime resource utilization and global power consumption.

3. End-to-End Mapping

Despite the great balance between high performance and power efficiency Uni-OPU and Light-OPU has achieved, their dependency on manual tuning which emphasizes on versatility has led the design space to be overwhelming. Due to the inaccessibility of brute-forced searching rather than poor performance, this methodology falls short of meeting the high requirements in high-speed iterative real-life applications and is gradually replaced by emerging HLS tools which present a future of highly customized and automated end-to-end mappings. Hardware architectures generated by HLS tools are divided into two categories, namely the streaming architecture and the single computation engine [5].

3.1. Streaming Architecture

Similar to Uni-OPU, the streaming architecture realizes the acceleration of target CNN by instantiating several processing units with different functions, sacrificing generality for performance. Each Processing Element (PE) only provides specific functions, such as forward calculation of convolutional layer, pooling, etc. Compared to the single computation engine architecture, streaming architecture can be customized and optimized for different network characteristics, and dynamically adjust the resource allocation of each PE to achieve better global performance. Fig. 2 depicts the framework of streaming architecture.

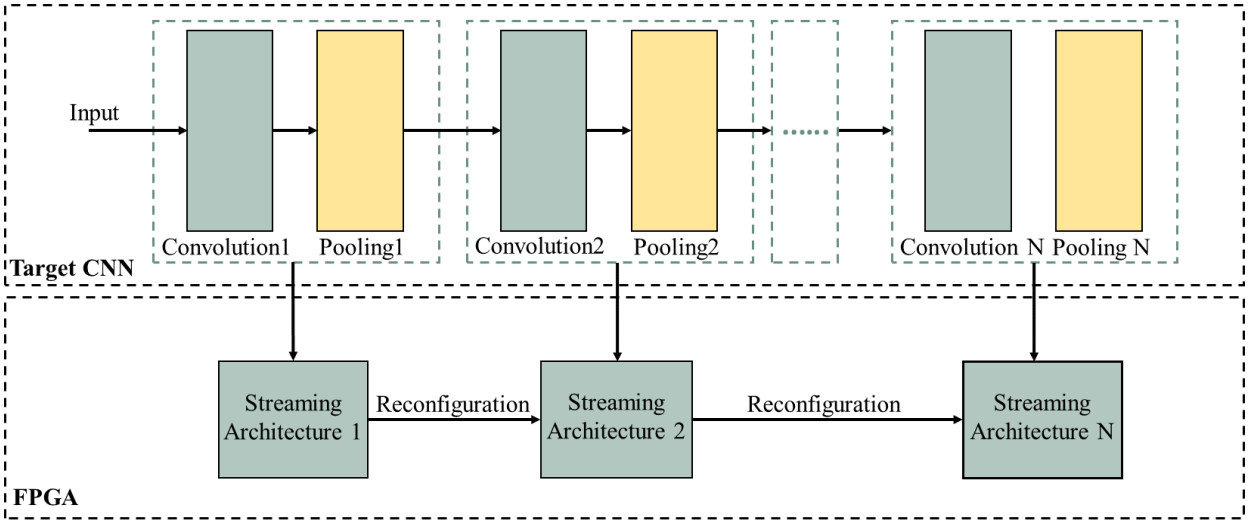


Figure 2. Illustration of the streaming architecture

3.1.1 FINN Framework

Originally customized for generating streaming architecture on low-precision networks such as Binarized Neural Network (BNN), FINN [6] meets the computational requirements of high precision CNN models. For a better coordination of workload allocation, FINN designed a special unit which consists of multiple PEs and SIMD lanes. Numbers of PEs and SIMD inside each unit are iteratively refined by allocating extra resources to the worst performing unit from the last round from a global perspective. The most recently revised version of FINN also adapted to Integer Linear Programming, enabling FINN to expand its application into multi-FPGA collaborative mapping.

However, according to experimental results, this automatic algorithm is still subject to optimization and is often outperformed by manual tuning. What's more, FINN requires that the external memory only focus on managing the input and output dataflow, and is not responsible for

the storage of the binarized weights, putting an upper limit on the scale of CNN models this framework could support. The on-chip weight storing restriction reduces the computational resources and weakens the advantages of FPGA in lightweight CNNs.

3.1.2 fpgaConvNet Framework

While FINN serves as a typical streaming architecture which sacrifices generality for performance, fpgaConvNet [7] focuses on restoring the balance by enhancing configurability. The framework explores multiple degrees of parallelism from both coarse-grained and fine-grained perspective. To alleviate limitations caused by resource constraints, the framework adopts the opposite design idea of FINN, namely, transferring weight storage from on-chip to off-chip. Furthermore, the framework facilitates bitstream reconfiguration and partial reloading of weights, overcoming resource constraints and demonstrating a potential for multi-FPGA mapping.

To avoid the design-space become prohibitively large, fpgaConvNet takes a significant step toward building a self-aware mechanism by deploying an optimizer employes based on the stimulated annealing algorithm, which has resulted in a considerable design space and availability in resource-constrained and latency-sensitive applications.

3.1.3 HSL4ML Framework

Orthogonal to FINN and fpgaConvNet, HSL4ML [8] focuses on further compressing CNN models through lossy non-structural approaches such as pruning and quantization rather than optimizing the design flow of the framework. While other streaming architectures such as fpgaConvNet focuses on improving the configurability, HSL4ML further sacrifices configurability to obtain peak performance in terms of resource saving and low latency. During the mapping, weights of the target CNN are first deployed compactly and then inflate. Lately, more capabilities for implementing HSL4ML have been introduced, including additional configuration handles for pruning and quantization as well as parallelization that allow reductions in latency and power consumption.

3.2. Single Computation Engine

Despite the efficiency of the streaming architecture, there exists redundancy in compilation due to the one-to-one correspondence between the bitstream generated and each layer of target CNN. Sacrificing customization for flexibility, the single computation engine only needs to compile once. Similar to Light-OPU, it has a specific hardware accelerator with specific instruction sets. The corresponding compilation software compiles the CNN model into a series of instructions, which are then executed by the accelerator in a sequential order. Fig. 3 depicts the framework of the single computation engine. In an ideal situation, a compiled bitstream can be tuned solely on the resources of FPGAs and be capable of targeting any CNN without taking the overhead produced by bitstream-level reconfiguration into account.

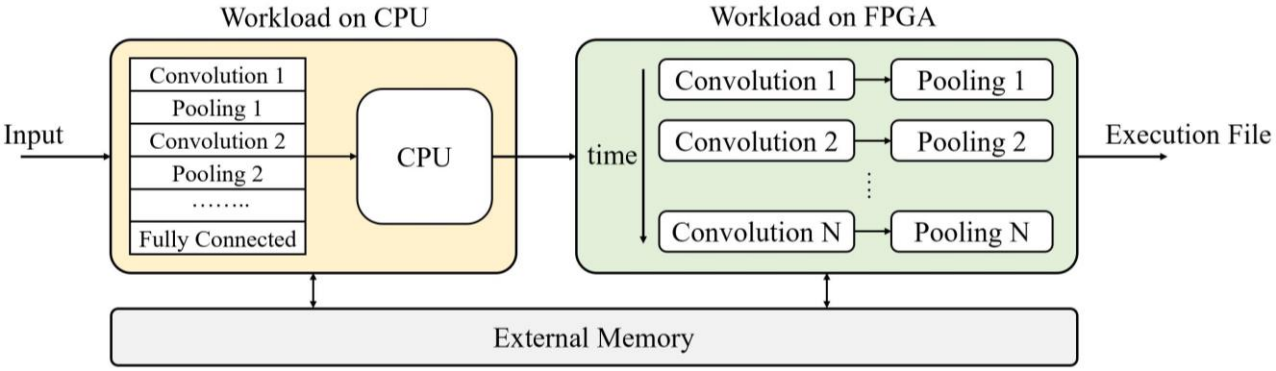


Figure 3. Illustration of the single computation engine

3.2.1 Angle-Eye Framework

Angle-Eye [9] upholds the principle of single computation engine, which ensures that hardware architectures generated by this paradigm are unaffected by characteristics of the target CNN. Each PE processes its input with a distinct set of kernels, generating independent feature maps from a shared input feature maps from the convolutional layer. As one of the earliest single computation engine frameworks, Angel-Eye serves as an excellent example for subsequent frameworks such as DNNWeaver [10], Snowflake [11], etc.

To avoid hardware constraints such as bandwidth requirements and power consumption, Angle-Eye offers a primitive selective mapping approach. Angle-Eye evaluate several possible mappings beforehand from the perspective of peak performance and hardware resource consumption, and applies it selectively. Although this is a band-aid solution under hardware resource constraints, it does enable more optimization opportunities.

3.2.2 GAMMA Framework

By using a modified genetic algorithm, GAMMA [3] conducts a complete search which takes into account all three dimensions of hardware-mapping, namely computation order, tiling strategy and parallelization strategy, outperforms earlier researches that solely focus on two levels of parallelism. In contrast to earlier research that either choose from several limited mapping options or targeted basic rigid accelerators, GAMMA constructs a flexible map-space and determines an optimum mapping. Hence, GAMMA employs specific cross operators to evolve cross-optimized mutational mappings similar to Angle-Eye.

3.2.3 unzipFPGA Framework

The so called unzipFPGA framework [12] focuses on minimizing the impact of memory-bound layers, breaking down the ceiling of resource constraints on previous single computation engines. Since FINN, most frameworks adopt off-chip memory to store weights, allocating more computational resources to the CNN model. To better utilize the computational resources while addressing the growing complexity of CNN models, unzipFPGA reproposes storing the compressed weights on-chip and recreating them instantly. The suggested method provides a weight generator with a highly optimized computation order and parallelization strategy, and a customized memory structure with high processing rates.

Meanwhile, unzipFPGA presented a novel CNN engine design that includes a variety of input selective PEs to address the issue of PE inefficiency in suboptimal mapping. As a backup plan, efficient switches have been equipped on a subset of PEs. According to experimental results, unzipFPGA outperforms other advanced FPGA-based accelerators at the same bandwidth while providing significant improvements in performance density [12].

4. Comparison of Existing Frameworks and Upcoming Challenges

4.1. Comparison of Existing Frameworks

In recent years, FPGA-based CNN accelerators have attracted much attention due to their flexible customizability and excellent parallelism. Hardware architecture still bases on the two traditional modes, namely streaming architecture and single computation engine, even though the design flow has evolved from manual tuning of CNN-specific processors to automated end-to-end mapping using HLS tools. Although not making revolutionary advances in several key performance indicators such as throughput, latency, and power consumption, end-to-end mapping has greatly simplified the design flow and design difficulties. With the integration of those frameworks, software is dominating the FPGA hardware design process. A customized and optimized implementations of CNN on FPGAs can be obtained without hardware design intervention, enhancing the integrity of FPGAs within the deep learning ecosystem.

Streaming architecture and single computation engine have their own emphasis. The former trades generality for performance by concentrating on the optimization for each CNN layer whereas the latter attempts to explore the architecture capable of accelerating a class of models. Although Uni-OPU is a CNN-specific processor, it constructs an instruction set within the streaming architecture, demonstrating a potential of combining the two design methods with its excellent performance [4].

Among the six end-to-end mapping frameworks mentioned above, designers continuously optimize the generation and storage of weights to achieve the optimal utilization of FPGA resources. The weights are binarized by FINN, partially reloaded by fpgaConvNet and stored using an off-chip technique, and proposed on-chip storage again by unzipFPGA, which includes real-time generation. Multi-FPGA collaborative mapping has also gained attention in recent years as an essential method of overcoming the restrictions of FPGA hardware resources. At the same time, when the optimization of the design flow gradually reached a bottleneck, HSL4ML puts forward a different technique to further compress the CNN model through pruning and quantization.

4.2. Upcoming Challenges

Despite the fact that current applications of lightweight CNN on FPGA are reasonably mature, the inconsistency of progresses between the field of deep learning and hardware design has also introduced new challenges.

While each framework design has always been on how to make better use of FPGA resources, the astonishing rate of evolution of the CNN model still outpaces hardware architecture. Furthermore, the growing complexity also leads to irregular layer connections, complicates the automated mapping process and necessitates additional manual-tuning [5]. While the growing workload also affects the robustness of the CNN model, support for low-precision CNN plays a crucial part in solving the problem. While most paradigms are still following approaches provided for ASIC platforms, designing dedicated low-precision CNN models for FPGA is a problem that needs to be solved.

5. Conclusion

This paper provides a review on 8 classic architectural cases of the streaming architecture and the single computation engine from traditional CNN-specific processors to end-to-end mapping using HLS tools in a sequential order, demonstrating that end-to-end mapping has gradually become the mainstream of the industry in recent years due to its great improvement in design efficiency. Through the comparative analysis of these frameworks, the author believes that automated and software-based mapping, further optimization of weight processing methods, multi-FPGA collaborative mapping and further compression of CNN model volume will be the future direction of FPGA-based CNN accelerators. This paper also argues for the case that it is necessary to address the issue of keeping up with the growing complexity of CNNs and gradually move beyond lightweight applications.

References

[1] Gu, Jiuxiang, Wang, et al. Recent advances in convolutional neural networks. Pattern Recognition the Journal of the Pattern Recognition Society, 2018.

[2] Yu Y, Zhao T, Wang K, et al. Light-OPU: An FPGA-based Overlay Processor for Lightweight Convolutional Neural Networks. FPGA '20: The 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2020.

[3] Krishna T, Kao S. GAMMA: automating the HW mapping of DNN models on accelerators via genetic algorithm. ICCAD '20: IEEE/ACM International Conference on Computer-Aided Design. ACM, 2020.

[4] Yu Y, Zhao T, Wang M, et al. Uni-OPU: An FPGA-Based Uniform Accelerator for Convolutional and Transposed Convolutional Networks. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2020, 28(7): 1545-1556.

[5] Venieris S I, Alexandros K, Christos-Savvas B. Toolflows for Mapping Convolutional Neural Networks on FPGAs: A Survey and Future Directions. ACM Computing Surveys, 2018, 51(3): 1-39.

[6] Blott M, Preusser T, Fraser N, et al. FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks. ACM Transactions on Reconfigurable Technology and Systems, 2018, 11(3): 1-23.

[7] Venieris S I, Bouganis C S. fpgaConvNet: A Framework for Mapping Convolutional Neural Networks on FPGAs. IEEE International Symposium on Field-Programmable Custom Computing Machines. 0.

[8] Fahim F, Hawks B, Herwig C, et al. hls4ml: An Open-Source Codesign Workflow to Empower Scientific Low-Power Machine Learning Devices. 2021.

[9] Guo K, Sui L, Qiu J, et al. Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017: 1-1.

[10] Sharma H, Park J, Mahajan D, et al. From high-level deep neural models to FPGAs. IEEE/ACM International Symposium on Microarchitecture. ACM, 2016.

[11] Gokhale V, Zaidy A, Chang A, et al. Snowflake: An efficient hardware accelerator for convolutional neural networks. 2017 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2017.

[12] Venieris S I, Fernandez-Marques J, Lane N D. unzipFPGA: Enhancing FPGA-based CNN Engines with On-the-Fly Weights Generation. 2021.