IAES Journal of Electrical and Electronics Engineering
Vol. 4, No. 2, December 2017, pp. 26~30
ISSN: 2338-7432, DOI: 10.11591/jeee.v4i2.pp26-30

FPGA-based artificial intelligence: from neural network acceleration to edge computing applications

Tole Sutikno, Kasman
Faculty of Industrial Technology, Universitas Ahmad Dahlan, Yogyakarta, Indonesia

Article Info

Article history:

Received Sep 13, 2016
Revised Jul 25, 2017
Accepted Sep 10, 2017

Keywords:

Artificial intelligence
Convolutional neural network
Edge computing
Field-programmable gate array
Hardware acceleration
Neural network

ABSTRACT

Field-programmable gate arrays (FPGAs) have become promising platforms for artificial intelligence (AI) workloads because they combine configurable hardware, parallel execution, and the ability to tailor datapaths to specific models. These properties are particularly relevant to neural networks, whose computational demand and memory traffic can challenge conventional embedded processors. This review examines FPGA-based AI development up to 2016, focusing on neural-network architectures, convolutional neural network (CNN) acceleration, design automation, and embedded deployment. It considers how parallelism, pipelining, data reuse, and reduced-precision arithmetic can improve throughput while managing limited logic and on-chip memory resources. The review also discusses the trade-offs between model accuracy, hardware efficiency, programmability, and system integration. Existing studies demonstrate that FPGA accelerators can support image recognition and other perception tasks, but they also reveal persistent constraints involving external-memory bandwidth, implementation complexity, and tool support. The analysis identifies hardware-aware model design and more accessible development flows as important requirements for wider adoption. It concludes that FPGAs offer a practical route toward energy-conscious edge intelligence, provided that algorithmic and architectural decisions are optimized together rather than treating neural-network software and hardware as separate stages.

This is an open access article under the CC BY-SA license.

1. INTRODUCTION

Artificial intelligence (AI) has become increasingly dependent on computationally intensive neural network models, particularly in image recognition, speech processing, and automated perception. Although central processing units (CPUs) and graphics processing units (GPUs) provide substantial computational capabilities, their general-purpose architectures may not always satisfy the latency, energy, and resource requirements of embedded applications. This limitation is particularly important when intelligent functions must operate close to sensors and users rather than relying entirely on remote computing infrastructure. Field-programmable gate arrays (FPGAs) offer an alternative because their configurable logic allows designers to construct specialized processing architectures for particular neural network workloads.

The central research problem is how to implement increasingly complex neural networks on hardware platforms with limited computational and memory resources. Neural network inference involves repeated arithmetic operations, data transfers, and intermediate storage, all of which can create bottlenecks in embedded systems. A successful implementation must therefore balance processing throughput, energy consumption, memory bandwidth, hardware utilization, and model accuracy. This review examines how FPGA architectures and accelerator designs address these requirements, how programming and design automation improve implementation productivity, and how these developments support intelligent edge applications.

The scope of this review is limited to research published no later than 2016. It focuses primarily on neural network and convolutional neural network (CNN) acceleration because these models provide

representative examples of the computational challenges associated with AI hardware. The selected literature includes FPGA-based processing architectures, accelerator optimization, embedded deployment, and automated hardware generation. By examining these developments together, the review identifies both the practical advantages of FPGA-based AI and the limitations that remained unresolved by 2016.

2. FPGA ARCHITECTURES FOR NEURAL NETWORK PROCESSING

2.1. Reconfigurable hardware for neural networks

FPGAs provide a distinctive implementation platform because their logic resources can be configured to execute multiple operations concurrently. Unlike conventional processors, which generally execute instructions through a predefined processing architecture, FPGAs allow designers to create application-specific datapaths. These datapaths can combine arithmetic units, registers, memory blocks, and control logic according to the structure of a neural network. Early work by Farabet *et al.* [1] demonstrated the feasibility of implementing convolutional network processing on FPGA hardware, establishing an important foundation for reconfigurable vision systems.

Neural networks contain computational structures that can benefit from this parallel organization. A typical network performs repeated multiplication, accumulation, activation, and data-transfer operations across multiple neurons or feature-map elements. FPGA designers can exploit parallel processing units and pipelined execution to perform several operations within a processing interval. However, the number of parallel units must be balanced against available logic resources, memory capacity, and the bandwidth required to supply data.

2.2. CNN accelerator architectures

CNNs are particularly demanding because convolutional layers repeatedly process large collections of input feature maps and filters. Their computational workload can be distributed across multiple processing elements, making them suitable for parallel hardware architectures. Chakradhar *et al.* [2] proposed a dynamically configurable coprocessor for CNNs, demonstrating how specialized hardware could support convolutional network computation while retaining configurability. This approach illustrates the value of matching accelerator organization to the computational characteristics of the target neural network.

Later research focused more directly on optimizing FPGA resources and data movement. Zhang *et al.* [3] proposed an analytical design methodology for FPGA-based deep CNN accelerators, using performance and memory-bandwidth considerations to guide architectural decisions. Their work highlighted that computational throughput alone does not determine accelerator performance because memory access can become a limiting factor. This finding is particularly relevant to embedded platforms, where external memory bandwidth and on-chip storage are often constrained.

2.3. Embedded FPGA platforms

Embedded FPGA platforms provide an opportunity to execute neural network inference near cameras, sensors, and other data sources. This arrangement can reduce dependence on remote servers and support applications that require timely responses. Qiu *et al.* [4] investigated CNN acceleration on embedded FPGA platforms and identified limited memory capacity and bandwidth as important design constraints. Their study illustrates that an effective embedded accelerator must consider the entire data path, rather than optimizing arithmetic units in isolation.

The architectural requirements of embedded AI differ from those of large data-center accelerators. Embedded devices commonly operate under stricter power, size, and thermal constraints, while also requiring predictable response times. FPGA reconfigurability allows the hardware to be tailored to these operating conditions, although the available resources may restrict the size of the supported neural network. Consequently, model selection and hardware architecture must be considered together during system design.

3. PROGRAMMING AND DESIGN OPTIMIZATION

3.1. Hardware-aware neural network implementation

Translating a trained neural network into FPGA hardware requires more than directly converting software operations into digital circuits. The network structure must be mapped onto available logic resources, arithmetic units, and memory blocks while maintaining acceptable inference accuracy. Designers must determine the level of parallelism, the number of processing elements, and the arrangement of data buffers. These decisions strongly influence the final accelerator's throughput, latency, and resource utilization.

One important optimization strategy is to reduce numerical precision where the application can tolerate it. Lower-precision representations can decrease memory requirements and simplify arithmetic

operations, allowing more processing elements to fit within the available FPGA fabric. However, aggressive precision reduction may affect the accuracy of network predictions, particularly when errors accumulate across multiple layers. Hardware optimization must therefore be evaluated alongside the accuracy requirements of the target application.

3.2. Design automation and high-level development

The complexity of manually designing neural network accelerators has encouraged the development of automated implementation tools. Wang *et al.* [5] introduced DeepBurning, a design automation framework for generating FPGA-based learning accelerators from neural network models. The framework addresses the difficulty of creating specialized hardware for different network configurations by coordinating accelerator generation and compiler support. Such tools are important because they can reduce the amount of manual hardware development required from AI application designers.

A related challenge is the need to optimize the complete CNN rather than individual computational layers. Zhang *et al.* [6] introduced Caffeine, a hardware/software co-designed framework intended to accelerate CNN execution across FPGA platforms. Its design emphasizes a unified representation of convolutional and fully connected layers, together with efficient use of computing and memory resources. These approaches indicate that automated mapping and software integration are essential for making FPGA-based AI development more accessible.

3.3. Comparison of FPGA-based AI approaches

The studies discussed above illustrate different strategies for implementing neural network computation on reconfigurable hardware. Table 1 compares representative approaches according to their main design emphasis and relevance to embedded AI. The comparison is intended to identify complementary contributions rather than rank the studies by performance, since their experimental platforms and evaluation conditions differ. It also demonstrates how FPGA research progressed from basic neural network processing toward more integrated and automated accelerator development.

Table 1. Representative FPGA-based neural network approaches published by 2016

Study	Main approach	Primary contribution	Relevance to edge AI
Farabet *et al.* [1]	FPGA convolutional network processor	Demonstrated reconfigurable CNN processing	Foundation for embedded vision
Chakradhar *et al.* [2]	Dynamically configurable coprocessor	Specialized CNN computation with configurable hardware	Adaptable neural processing
Zhang *et al.* [3]	Analytical CNN accelerator optimization	Balanced computational throughput and memory bandwidth	Resource-aware acceleration
Qiu *et al.* [4]	Embedded FPGA CNN platform	Addressed embedded memory and bandwidth constraints	On-device image recognition
Wang *et al.* [5]	DeepBurning design automation	Automated generation of learning accelerators	Simplified accelerator development
Zhang *et al.* [6]	Caffeine hardware/software co-design	Unified CNN representation and resource optimization	More integrated deployment

Table 1 shows that FPGA-based AI development involves several interconnected design priorities. Early architectures established the feasibility of implementing neural network operations in programmable logic, while subsequent work focused on improving throughput and managing memory constraints. Automated frameworks then addressed the difficulty of mapping different network models onto FPGA resources. Taken together, these studies suggest that practical edge AI requires both efficient hardware architectures and development tools capable of supporting model-specific optimization.

4. APPLICATIONS IN EDGE COMPUTING

4.1. Embedded vision and intelligent perception

Embedded vision is one of the most direct application areas for FPGA-based neural network acceleration. Cameras and imaging sensors generate data that can be processed locally for object recognition, classification, and scene interpretation. FPGA accelerators can implement convolutional operations and other network stages close to the data source, reducing the need to transmit all raw information to a remote server. The embedded CNN platform investigated by Qiu *et al.* [4] provides an example of how FPGA resources can be used to support image recognition under constrained hardware conditions.

Local processing can be valuable in applications where communication delays or network availability are important concerns. Robotics, intelligent surveillance, and autonomous sensing systems may need to respond to visual information within predictable time limits. An FPGA-based accelerator can be configured to execute selected neural network operations in parallel, while a processor handles system management and application logic. This division of responsibilities offers a practical route toward responsive embedded intelligence.

4.2. Robotics and resource-constrained devices

Robotic systems frequently combine perception, decision-making, and control within limited power and computational budgets. Neural networks can support tasks such as visual recognition and environmental interpretation, but their computational requirements may compete with other real-time functions. FPGA acceleration offers the possibility of assigning computationally intensive operations to dedicated hardware while retaining software flexibility for system coordination. DeepBurning's focus on embedded vision and robotics-related applications illustrates the relevance of automated learning-accelerator generation to such systems [5].

However, FPGA deployment does not automatically guarantee a suitable real-time solution. The overall system must account for sensor input rates, memory transfers, accelerator latency, and communication with the host processor. A high-throughput neural network accelerator may still be unsuitable if data cannot be delivered to it efficiently. Therefore, the performance of edge AI systems should be evaluated at the system level rather than through isolated accelerator measurements.

5. RESEARCH CHALLENGES AND FUTURE DIRECTIONS

One persistent challenge is the relationship between computational parallelism and memory bandwidth. Increasing the number of processing elements does not necessarily improve performance when the accelerator cannot receive input data quickly enough. Limited on-chip memory can also force frequent transfers to external memory, increasing latency and energy consumption. Future architectures should therefore prioritize data reuse, efficient buffering, and coordinated optimization of computation and communication.

Another challenge concerns the trade-off between neural network accuracy and hardware efficiency. Reduced-precision arithmetic and model simplification can make larger networks feasible on resource-constrained devices, but they may also affect prediction quality. The appropriate optimization level depends on the application, since a small accuracy reduction may be acceptable in one task but unacceptable in safety-critical perception. More systematic hardware-aware training and model-design methods are therefore important for reliable FPGA-based AI.

Programming accessibility remains equally important for broader adoption. Low-level hardware development provides detailed implementation control, but it requires specialized knowledge and can increase development time. Automated frameworks such as DeepBurning and Caffeine represent progress toward more accessible hardware/software development flows [5], [6]. Further improvements in compiler quality, performance estimation, and design-space exploration could help developers achieve efficient implementations without requiring extensive manual hardware optimization.

Finally, edge computing requires the accelerator to operate as part of a complete intelligent system. This includes sensor interfaces, communication links, host processors, memory management, and mechanisms for updating or adapting the deployed model. FPGA reconfigurability offers a useful foundation for adapting hardware as application requirements change. Nevertheless, dependable edge deployment will require improvements in system integration, software support, energy management, and verification.

6. CONCLUSION

FPGAs provide a promising hardware platform for AI because they combine reconfigurability with the ability to implement parallel, application-specific processing structures. Research published through 2016 demonstrated their potential for accelerating neural networks and CNNs, particularly in image recognition and embedded perception applications. The development of specialized coprocessors, embedded CNN platforms, and automated accelerator-generation frameworks reflects a gradual shift from isolated hardware implementations toward more integrated AI systems. These developments establish FPGA technology as a practical option for applications that require configurable processing under resource constraints.

However, successful FPGA-based AI deployment depends on more than computational throughput. Memory bandwidth, on-chip storage, numerical precision, programming complexity, and system integration all influence the final performance of an accelerator. Future progress therefore requires coordinated optimization of neural network models, hardware architectures, and development tools. By treating algorithm

design and hardware implementation as connected stages, FPGA platforms can contribute to more efficient and responsive edge intelligence systems.

REFERENCES

[1] C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun, "CNP: An FPGA-based processor for Convolutional Networks," in *FPL 09: 19th International Conference on Field Programmable Logic and Applications*, Aug. 2009, pp. 32–37. doi: 10.1109/FPL.2009.5272559.

[2] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," in *Proceedings - International Symposium on Computer Architecture*, Jun. 2010, pp. 247–257. doi: 10.1145/1815961.1815993.

[3] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *FPGA 2015 - 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Feb. 2015, pp. 161–170. doi: 10.1145/2684746.2689060.

[4] J. Qiu *et al.*, "Going deeper with embedded FPGA platform for convolutional neural network," in *FPGA 2016 - Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Feb. 2016, pp. 26–35. doi: 10.1145/2847263.2847265.

[5] Y. Wang, J. Xu, Y. Han, H. Li, and X. Li, "DeepBurning: Automatic generation of FPGA-based learning accelerators for the Neural Network family," in *Proceedings - Design Automation Conference*, Jun. 2016, vol. 2016-August, pp. 1–6. doi: 10.1145/2897937.2898002.

[6] C. Zhang, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, Nov. 2016, vol. 07-10-November-2016, pp. 1–8. doi: 10.1145/2966986.2967011.