Implementing Convolutional Neural Networks on FPGA: A Survey and Research

Abdelilah Haijoub[1*], *Anas* Hatim[2], *Mounir* Arioua[1], *Slama* Hammia[2], *Ahmed* Eloualkadi[1] *and Antonio Guerrero-González*[3]

[1]LabTIC, ENSA, Abdelmalek Essaadi University, Tangier, Morocco
[2]TIM Team, ENSA, Cadi Ayyad University, Marrakech, Morocco
[3]Universidad Politécnica de Cartagena, Cartagena, Spain

Abstract. The implementation of CNN FPGA is of increasing importance due to the growing demand for low-power and high-performance edge AI applications. This paper presents a comprehensive survey and research on the topic, with a focus on comparing and evaluating the performance of two main FPGA architectures, streaming and single unit computing. The study includes a detailed evaluation of the state-of-the-art CNNs, LeNet-5 and YOLOv2, on both FPGA architectures. The results provide useful insights into the trade-offs involved, limitations, challenges, and the complexity of implementing CNNs on FPGAs. The paper highlights the difficulties and intricacies involved in implementing CNNs on FPGAs and provides potential solutions for improving performance and efficiency.

1 Introduction

Convolutional Neural Networks (CNNs) have become an important tool in the field of computer vision, image and speech recognition. Their ability to learn hierarchical features from raw data and make accurate predictions has led to numerous breakthroughs in various applications. However, CNNs are computationally intensive, requiring high computational power and energy consumption. This becomes a challenge in real-time applications, especially when dealing with large data sets and limited computational resources.

Several research works have been performed in the area of implementation of CNN FPGA. These works have explored various approaches and architectures to tackle the challenges posed by this task. These approaches range from utilizing the streaming architecture to utilizing the single unit computing approach. The choice of a method depends on the needs and limitations of the intended application, and each approach has its own advantages and disadvantages. Nguyen, Tuan Nghia, Duy Thanh, KIM, Hyun and Hyuk-Jae Lee [1] propose a high-performance and hardware-efficient streaming architecture for YOLOV2 [2]. The design achieves this by quantizing the network and optimizing the data path to reduce off-chip access for intermediate data. With batch processing of the streaming data, the proposed architecture boasts a throughput of 1.877 TOPS without incurring additional hardware costs. Erwei Wang [3] proposes a fast and efficient FPGA prototyping

* Corresponding author: Haijoub.Abdel@gmail.com

framework for deploying CNN LeNet-5 [4] on the PYNQ platform. The implementation boasts a frame rate of 3.81 frames per second and an average power consumption of 1.951 W.

While remaining within the PYNQ platform and utilizing a single-unit computing approach, SHARMA, Aman, SINGH, Vijander, et RANI, Asha [5] demonstrated a successful deployment of a CNN on a Xilinx Zynq FPGA using the Xilinx PYNQ framework. They analyzed the software-hardware co-design tradeoff and evaluated four object detector algorithms based on mean average precision and frame per second metrics. The results showed that the SSD + Inception v2 model had the optimal accuracy with near real-time frame rate, making it suitable for real-time applications on embedded platforms. Researchers use these frameworks to implement CNNs on FPGA boards, KALAPOTHAS, Stavros, FLAMIS, Georgios, and KITSOS, Paris [6] proposed a qualitative method for evaluating the capabilities of a reconfigurable platform for computer vision applications. They compared the legacy DNNDK framework on the Xilinx ZedBoard with the new Kria SoM platform and found the latter to have better computing performance and power efficiency. HASSAN, Rania O. and MOSTAFA, Hassan. [7] proposed a solution to balance speed and power consumption for CNNs and FPGA in real-time image classification using parallelism and pipelining, and introduced a new methodology using Xilinx SDSOC tool.

Other researchers have suggested co-designing hardware and software using the single-unit computing approach. Most current studies on implementing Convolutional Neural Networks (CNNs) on Field-Programmable Gate Arrays (FPGAs) aim to reduce the size of the CNN and fit it entirely onto the FPGA to eliminate off-chip data access. This is usually done by utilizing low-precision or binary CNNs. However, this strategy demands a lengthy optimization process, which can lead to substantial accuracy reduction compared to 8-bit CNNs. Moreover, implementing the entire CNN on the chip is limited by the available on-chip memory in one example, NGUYEN, KIM, Hyun, Duy Thanh, and LEE, Hyuk-Jae [8] stored the entire CNN in the on-chip memory of a Xilinx Virtex-7 VC707 FPGA, but this approach is only feasible if all the weights can be stored on-chip. However, this leads to high power consumption and may not be feasible for low-resource IoT edge devices with limited on-chip memory [9].

For other models, we found ZHAI, Sheping, QIU, Cheng, YANG, Yuanyuan, Jing Li and Yiming Cui [10] propose a design that preserves the accuracy of the Lenet-5 model despite the precision of floating-point operations in the hardware implementation. With sufficient training, the network recognition accuracy rate reaches 97%, comparable to that of CPU and GPU platforms. WANG, Zixiao, Shuaixiao, LI LiU, XU, Ke, WU and Dong Wang . [11] proposed new methods for low-resource, low-power CNN implementation on FPGA for IoT edge devices to handle data transfer between on-chip and off-chip memory. Existing optimization techniques can't fit into low-resource FPGAs, as demonstrated by a study which showed a consumption of 26W power and 67 M bits of on-chip memory. Afzal Ahmad, Ghulam Jilani Raza and Muhammad Adeel Pasha [12] propose a hardware/software co-design approach to accelerate Tiny YOLOv3 [13], a popular lightweight object detection neural network. They measure the complexity of various operations and use hardware acceleration for the most complex operation, convolution. The results show significant performance improvements, ranging from 3.9 to 21.3 times better than existing state-of-the-art CNN accelerators for image classification and object detection. Remaining on YOLOv3, DING, Caiwen, LIU, Ning,Kaidi Xu,Yanzhi Wang ,WANG, Shuo, and Yun Liang [14] proposed REQ-YOLO, a resource-aware quantization framework for object detection with block-circulant matrix method and ADMM, equal-distance and heterogeneous quantization methods. It was implemented on FPGA with efficient PE, memory optimization, and a synthesis framework, and showed improvement over state-of-the-art FPGA designs. BOUGUEZZI, Safa, FREDJ, Tarek, Carlos Valderrama, Hana Ben BELABED, Hassene

Faiedh and Chokri Souani [15] proposed a low-resource network design for FPGA with high accuracy by improving MobileNet-V1 architecture. The FPGA implementation on Virtex-7 board achieved 88.76% recognition rate with the MAd-MobileNet model and 225 MHz speed with the RAd-MobileNet network.

Furthermore, in the state-of-the-art, several research studies have been carried out for YOLOv2 implementation on FPGA, which will be the basis for comparison in this paper. LI, Shuai, LUO, Yukui, SUN, kuangyuan, nandakishor yadav and kyuwon ken choi [16] proposed an efficient FPGA implementation to reduce data transfer with a pipelined MAC operation. This resulted in reduced loop dependency and increased parallelism. However, the high resource utilization caused high power consumption (27.2 W) on the Intel Arria10 GX1150 FPGA, making it unsuitable for low-resource IoT devices. XU, Ke, WANG, Xiaoyun, LIU, Xinyang, Huolin Li, Haiyong Peng, Changfeng Cao and Dong Wang [17] Proposed efficient FPGA solution for data transfer, using pipelined MAC and optimized dataflow for parallelism and reduced data access. However, high resource utilization leads to high power consumption (27.2W) making it unsuitable for low-resource, battery-powered IoT platforms. ZHANG, Zhichao, MAHMUD, MA Parvez, and KOUZANI, Abbas Z. [18] Present object Detection on Low-Resource IoT Devices using FPGA-based YOLOv2 Implementation. Improves data transfer with efficient dataflow and multi-level buffers, leading to low memory utilization and power consumption of 4.8 W. Limitations include conflict between YOLOv2 size and low-resource requirement and support for only 8-bit precision. Future work focuses on improving speed, precision flexibility, and CNN optimization algorithms.

In this paper, section 2 gives an overview of the architectures used in the implementation of CNN in the FPGA and the different framwork existing. We provide a description of the methodology used in our survey and research on FPGA in Section 3. The document ends with a summary of the results.

2 Background

This research is part of a larger project focused on investigating the use of FPGAs for CNN implementation. Recently, FPGAs have become a highly attractive platform for implementing CNNs in real-time, with low latency and power consumption, for various applications including image classification, segmentation and object detection.

In this context, two main approaches have been proposed: streaming architectures and single unit computing.

Streaming Architecture refers to a type of hardware design that processes data in a continuous stream. This architecture is particularly well-suited for tasks that require real-time processing of large amounts of data, such as video and audio processing, image processing, and data compression. The key characteristic of streaming architecture is that it processes data as soon as it arrives, allowing it to handle large data sets with low latency. FPGAs are an ideal platform for implementing streaming architecture because they are highly configurable, allowing designers to create custom hardware solutions that match the specific requirements of their applications.

Fig. 1. Overall architecture of streaming approach.

Single computation unit (SCU) accelerators are hardware components that are designed to perform specific tasks with high performance and low power consumption. In the context of FPGAs, SCUs are programmable logic blocks that can be configured to perform specific computations, such as those needed for artificial intelligence. These accelerators are particularly useful for implementing deep neural networks and other computationally intensive ML algorithms, as they provide a high degree of parallelism and can perform computations much faster than traditional processors.

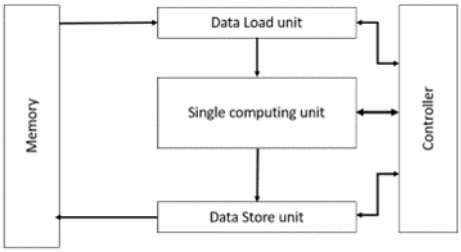

Fig. 2. Overall architecture of SCU approach.

CNN acceleration FPGA is a rapidly growing area of research and development, with numerous accelerator frameworks available to assist in the implementation of CNNs on these platforms:

• DNNWeaver [19] is a software framework designed for deploying deep neural network (DNN) applications FPGA. It provides a high-level programming interface for developing DNNs, allowing developers to quickly and easily implement their DNN models on FPGAs without needing to know low-level hardware details. With its support for various DNN models and layers.

• HLS4ML [20] is an open-source high-level synthesis (HLS) toolkit for converting machine learning models into optimized, hardware-friendly RTL implementations that can be run on FPGA platforms. It provides a way to take trained machine learning models, such as those developed in popular frameworks like TensorFlow, and translate them into optimized RTL designs that can run on FPGA hardware.

• CHaiDNN is an open-source HLS framework for the implementation of deep neural networks (DNNs) on FPGA hardware. It is designed to provide a high-level abstraction for the implementation of DNNs, making it easier for developers to implement DNNs on FPGAs without the need for low-level hardware design.

• FpgaConvNet [21] is a FPGA-based implementation of CNN (ConvNets or CNNs), which is a type of deep learning algorithm commonly used in computer vision and image recognition tasks. The implementation is designed to run efficiently on FPGAs, allowing for real-time processing of large amounts of data and efficient use of available resources.

• DeepBurning [22] is a framework for running deep neural networks on FPGA-based hardware platforms. It aims to provide an efficient and flexible solution for running deep learning algorithms on these platforms. DeepBurning uses high-level synthesis to convert deep learning algorithms into hardware circuits, making it easier for developers to implement their models on FPGAs.

• HADDOC2 is an acronym for Hardware Accelerated Deep Deep On-Chip CNN. It is a hardware accelerator design for CNN that aims to provide high-performance and low-power solutions for CNN-based applications. HADDOC2 utilizes the features of FPGA platforms, including reconfigurable hardware, parallel computing capability, and low-power consumption, to optimize the performance of CNNs.

• The Xilinx Deep Learning Processor Unit (DPU) is an accelerator designed for deep learning inference on Xilinx FPGAs. The DPU is optimized for CNN and provides a high-performance solution for running deep learning models on edge devices with low power consumption.

3 Methodology

In this section, we describe the methodology used in our survey and research on the implementation of CNN on FPGA. We outline the criteria used for comparison between different implementation methods, including:

• Block Random Access Memory (BRAM) utilization: refers to the amount of on-chip memory resources used by an implementation of a CNN on FPGA.

• DSP utilization refers to the utilization of digital signal processing units in an FPGA for implementing a CNN.

• LUT utilization refers to the number of Look-Up Tables (LUTs) used in the implementation of a CNN on FPGA.

• Precision refers to the number of bits used to represent the values of data and operations in digital circuits

• Frame size refers to the dimension of a single image or frame used in a CNN implementation.

• Power consumption is a measure of the amount of electrical energy used by a device over a certain period of time.

• Throughput refers to the number of operations or tasks performed by a system per second. In the context of an FPGA implementation of CNNs, the throughput can be measured in GOP/s (Giga Operations Per Second), which represents the number of CNN operations executed by the FPGA per second.

• Frequency refers to the number of clock cycles per second in a digital circuit.

• Frames per second (FPS) refers to the number of individual frames or images that are processed per second

4 Results and discussion

In this comparison, we examine two different studies that implemented LeNet-5 on FPGA using two different approaches: The Single Unit Computing method and the Streaming method. The first study [10] was conducted on the Zedboard, while the second was conducted on the PYNQ-Z1 [3]. The results were compared in terms of execution time, power consumption and resources used to evaluate the advantages and disadvantages of each approach.

Table 1. Comparison between SCU and streaming architecture.

	SCU [10]	**Streaming [3]**
Times（ms）	6.8	2.024
Powers（W）	4.35	1.895

The performance comparison of the Single Unit Computing method and the Streaming method shows that the Single Unit Computing method offers slower execution times than the Streaming method, with an execution time of 6.8 ms versus 2.024 ms for the Streaming

method. However, the Streaming method has a lower power consumption, with a power consumption of 1.895 W compared to 4.35 W for the Single Unit Computing method

Occupation of the Xilinx Zedboard and PYNQ-Z1 card modules during the implementation of LeNet-5 with the Single Unit Computing method:

Table 2. Comparison between SCU and streaming architecture.

	SCU [10]	**Streaming [3]**
BRAM	257	139
DSP	188	130
LUT	25436	38304

The two tables show the results of implementing LeNet-5 on two different FPGA boards with different computing methods. The first method is Single Unit Computing with 18-bit quantization and the second is Streaming with 8-bit quantization.

The comparison of the two tables shows differences in the use of the modules (BRAM, DSP and LUT) between the Single Unit Computing method quantized in 18 bits on the Xilinx Zedboard and the Streaming method quantized in 8 bits on the PYNQ-Z1 board. While the Single Unit Computing method has high BRAM and DSP usage, but relatively low LUT usage, the Streaming method has maximum BRAM usage, moderate DSP usage and relatively high LUT usage.

In conclusion, the comparison between the two methods of unitary computation and streaming shows notable differences in the execution rate, power, the number of modules used and their occupancy rate. Table 1 shows that the streaming method is faster, while Table 2 shows that the streaming method used more BRAMs, DSPs and LUTs on the PYNQ-Z1 board, while it shows that the single unit computing method used fewer modules on the Xilinx Zedboard.

For complex models such as YOLO, it is better to use the single unit computing method for optimal performance.

The objective of this comparison is to examine the different results obtained in scientific papers on the use of the Single Unit Computing method on FPGA boards. This comparison will focus on aspects such as performance in terms of execution time and power consumption, as well as the complexity of the models used, and the advantages and disadvantages of this method compared to other signal processing methods on FPGAs. The selected scientific papers have worked on single unit computing and same model detection approach to understand the use of this method on FPGA boards with different architecture and provide an overview of the latest trends and developments in this field.

Several researchers have used this single unit computing method to implement the YOLOV2 model, we have selected 4 papers that have used the same model and size of image on different platforms. The following table presents the comparison.

In comparing the specifications of these four FPGAs for their use in single unit computing in the implementation of CNN on FPGA, it can be noted that the Zynq UltraScale MPSoC ZU7EV has the largest number of BRAMs at 8298 Kb and the most LUTs at 91.17 K. However, it has the lowest power consumption at 4.8 W. On the other hand, the Arria-10 GX1150 has the most DSPs at 1092 and the highest throughput at 1817.5 GOP/s, as well as operating at the highest frequency at 204 MHz. All FPGAs have a frame size of 416x416 and 8-bit precision, except for the Arria-10 which has a precision of 8-16 bits. The FPS values for the Zynq UltraScale MPSoC ZU7EV are not available, but among the other FPGAs, the Arria-10 has the highest FPS at 61.9.

In conclusion, the table compares the specifications of 4 FPGAs for their use in single unit computing in the implementation of YOLOV2 on FPGA. The Zynq UltraScale MPSoC ZU7EV has the largest number of BRAMs and LUTs, but also has the lowest power consumption.

[16] conducted a test of the same model on the GPU card, it shows a power consumption of 219.1 W and a fps of 100. The previous chart on the Arria-10 GX1150 and Zynq UltraScale MPSoC ZU7EV FPGAs showed a power consumption of 26 W and 18.17 W respectively, which is significantly lower than the power consumption of the GPU. However, in terms of fps, the GPU only reaches 100 fps, while the Arria-10 GX1150 and Zynq UltraScale MPSoC ZU7EV FPGAs achieve 61.9 fps. In conclusion, in terms of power consumption, FPGAs are more efficient.

Table 3. Comparison between different algorithms using SCU

	[18]	**[11]**	**[17]**	**[16]**
Year	2022	2020	2020	2020
FPGA	Zynq UltraScale MPSoC ZU7EV	Arria-10 GX1150	Arria-10 GX1150	Arria-10
BRAMs (Kb)	8298	52090	-	27320
DSPs	387	91	1092	410
LUTs (K)	91.17	517.5	-	-
Precision (bit)	8	8	8	8-16
Frame size	416*416	416*416	416*416	416*416
Power (W)	4.8	26	26	27.2
Throughput (GOP/s)	100.33	1817.5	566	740
Frequency (mhz)	200	204	190	200
FPS	-	61.9	18.86	25

5 Conclusion

In this article, we provide an overview of various approaches for implementing CNN on FPGA boards. We compare and analyze the performance, power consumption, and resource utilization of different FPGA-based CNN implementations. The results show that the design choices single unit computing have a significant impact on the performance and power efficiency of the system. FPGA-based CNN accelerators can achieve real-time performance and high-power efficiency, making them suitable for various applications, especially in embedded systems with stringent resource constraints. Further research is needed to explore new optimization techniques and to balance the trade-off between performance and resource utilization.

References

1. D. T. Nguyen, T. N. Nguyen, H. Kim, H.-J. Lee, A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection. *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*. 27, 1861–1873 (2019)
2. J. Redmon, A. Farhadi, Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR) (2017)
3. E. WANG, PYNQ Classification-Python on Zynq FPGA for Neural Networks, Imperial College London, *Final Year Project Report*, (2017)
4. Z. Zou, K. Chen, Z. Shi, Y. Guo, J. Ye, Object detection in 20 years: A survey. *Proceedings of the IEEE*. 111, 257–276 (2023)
5. A. Sharma, V. Singh, A. Rani, Implementation of CNN on Zynq based FPGA for real-time object detection. *2019 10th International Conference on Computing, Communication and Networking Technologies* (ICCCNT) (2019)
6. S. Kalapothas, G. Flamis, P. Kitsos, Efficient edge-AI application deployment for fpgas. *Information*. 13, 279 (2022)
7. R. O. Hassan, H. Mostafa, Implementation of Deep Neural Networks on FPGA-CPU platform using Xilinx SDSOC. *Analog Integrated Circuits and Signal Processing*. **106**, 399–408 (2020).
8. D. T. Nguyen, H. Kim, H.-J. Lee, Layer-specific optimization for mixed data flow with mixed precision in FPGA design for CNN-based Object Detectors. *IEEE Transactions on Circuits and Systems for Video Technology*. **31**, 2450–2464 (2021)
9. G. Dinelli, G. Meoni, E. Rapuano, L. Fanucci, Advantages and limitations of fully on-chip CNN FPGA-based hardware accelerator. *2020 IEEE International Symposium on Circuits and Systems (ISCAS)* (2020)
10. W. Lin, L. Zhang, Design of convolutional neural network SOC system based on FPGA. *2020 International Conference on Communications, Information System and Computer Engineering (CISCE)* (2020)
11. Z. Wang, K. Xu, S. Wu, L. Liu, L. Liu, D. Wang, Sparse-YOLO: Hardware/software co-design of an FPGA accelerator for yolov2. *IEEE Access*. **8**, 116569–116585 (2020)
12. A. Ahmad, M. A. Pasha, G. J. Raza, Accelerating tiny yolov3 using FPGA-based hardware/software co-design. *2020 IEEE International Symposium on Circuits and Systems (ISCAS)* (2020)
13. J. Redmon, A. Farhadi, Yolov3: An incremental improvement, In: *Computer vision and pattern recognition*, Berlin/Heidelberg, Germany 2018, (2018)
14. C. Ding, S. Wang, N. Liu, K. Xu, Y. Wang, Y. Liang, Req-yolo. *Proceedings of the 2019* ACM/SIGDA *International Symposium on Field-Programmable Gate Arrays* (2019)
15. S. Bouguezzi, H. B. Fredj, T. Belabed, C. Valderrama, H. Faiedh, C. Souani, An efficient FPGA-based convolutional neural network for classification: AD-mobilenet. *Electronics*. **10**, 2272 (2021)
16. S. Li, Y. Luo, K. Sun, N. Yadav, and K. K. Choi, A novel FPGA accelerator design for real-time and ultra-low power deep convolutional neural networks compared with Titan X GPU. *IEEE Access*. 8, 105455–105471 (2020)
17. K. Xu, X. Wang, X. Liu, C. Cao, H. Li, H. Peng, and D. Wang, A dedicated hardware accelerator for real-time acceleration of yolov2. *Journal of Real-Time Image Processing*. 18, 481–492 (2020)

18. Z. Zhang, M. A. Mahmud, A. Z. Kouzani, Resource-constrained FPGA implementation of yolov2. *Neural Computing and Applications*. **34**, 16989–17006 (2022)
19. H. Sharma, J. Park, E. Amaro, B. Thwaites, P. Kotha, A. Gupta, K. Kim Joon, A. Mishra, H. Esmaeilzadeh, Dnnweaver: From high-level deep network models to fpga acceleration, *In: The Workshop on Cognitive Architectures*, 2016, (2016)
20. F. Fahim, B. Hawks, C. Herwig, J. Hirschauer, S. Jindariani, N. Tran, L.P. Carloni, G. Di Guglielmo, P. Harris, J. Krupa, D. Rankin, hls4ml: An Open-Source Codesign Workflow to Empower Scientific Low-Power Machine Learning Devices, *arXiv preprint arXiv:2103.05579*, (2021)
21. S. I. Venieris, C.-S. Bouganis, FpgaConvNet: Mapping regular and irregular convolutional neural networks on fpgas. *IEEE Transactions on Neural Networks and Learning Systems*. **30**, 326–342 (2019)
22. Y. Wang, J. Xu, Y. Han, H. Li, X. Li, Deepburning. *Proceedings of the 53rd Annual Design Automation Conference* (2016)