Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/51/20241212

A comparative study of FPGA and CGRA technologies in hardware acceleration for deep learning

Jiawei Luo

School of Electronics and Computer Science, University of Southampton, Southampton, S017 1BJ, UK

jl6a19@southamptonalumni.ac.uk

Abstract. With the rapid growth of deep neural networks, their increasing demand for computing resources poses significant challenges to resource-constrained edge devices. Field Programmable Gate Array (FPGA) and Coarse-Grained Reconfigurable Arrays (CGRA) have emerged as promising solutions due to their parallelism and energy efficiency advantages. This paper conducts a comprehensive comparative analysis of FPGA and CGRA for accelerating deep learning workloads. It begins by introducing convolutional neural network architecture and optimization techniques. Various dataflow strategies are discussed for memory access optimization. The paper then analyzes representative cases to clarify FPGA and CGRA architectural advantages, along with optimization techniques through different dataflow strategies. The comparison reveals that FPGA offers higher throughput with fine-grained programmability for flexible bit-level operations and dataflow control. In contrast, CGRA excels in energy efficiency and area consumption with coarse-grained custom processing units and optimized on-chip interconnection networks. Quantitative comparisons are provided for throughput, energy efficiency, and area occupation. These insights can serve as a reference for selecting the appropriate architecture when designing custom hardware accelerators for deep learning workloads.

Keywords: Deep Learning, Hardware Acceleration, FPGA, CGRA, Dataflow.

1. Introduction

In the field of science and technology, the widespread application of artificial intelligence is changing the world. In particular, deep learning, as an important branch of artificial intelligence, has shown significant advantages in image recognition, natural language processing and complex decision-making [1]. However, deep learning models' computationally intensive and complex nature imposes strict requirements on hardware platforms [2]. Traditional computing platforms, such as CPUs and GPUs, have begun to fall short in meeting the high performance and low power requirements of deep learning [2]. Therefore, the development of new hardware solutions has become a current research focus.

Among many hardware technologies, Field Programmable Gate Array (FPGA) and Coarse-grained Reconfigurable Array (CGRA) are gradually considered as ideal choices for hardware acceleration of deep learning due to their excellent computing power, flexible programmability and efficient energy consumption ratio [3]. This paper will deeply explore and discuss the application of FPGA and CGRA in deep learning from three aspects. Firstly, the basic structure and principle of neural networks are

DOI: 10.54254/2755-2721/51/20241212

discussed in detail, and the optimization method of convolution operation is introduced. Secondly, the basic principles of FPGA and CGRA and efficient memory access techniques are analyzed. Then, the implementation and optimization strategies of these two hardware technologies in deep learning are studied through case studies. Finally, FPGA and CGRA are compared in terms of throughput, energy efficiency and area, and their advantages and application scenarios are analyzed.
This paper aims to provide a reference for the design selection of deep learning hardware accelerator system by summarizing the previous results, analyzing and comparing the optimization methods of FPGA and CGRA.

2. Deep Neural Networks and Computational Paradigms

2.1. Definition

Significant advancements in the field of Artificial Intelligence (AI) are attributed to the development of Deep Neural Networks (DNNs), a progression facilitated by improvements in computational capacity, access to large-scale data, and algorithmic innovation [4]. The robustness of DNNs as a tool for AI is underscored by their provision of state-of-the-art results across a wide array of domains [5]. The success of DNNs hinges on hierarchical feature learning through multiple stacked layers. Stacking numerous hidden layers between the input and output layers allows DNNs to extract highly complex feature representations and function mappings from raw data [5].

Amongst various DNN models, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) find the most extensive application [6]. CNNs employ convolution operations to handle spatial correlations, making them especially adept at recognizing spatial hierarchies or patterns in data [6]. This attribute positions them as the preferred choice for tasks such as object detection, facial recognition, and even medical image analysis. On the other hand, RNNs and their variants, such as Long-Short Term Memory (LSTM), are designed to identify patterns in sequential data, making them suitable for handling tasks involving time-series data, including speech recognition, natural language processing, and sentiment analysis [7]. Unlike CNNs, RNNs possess "memory" in that the information utilized in the current step is derived from the previous one, thereby enabling the association of past information with present tasks.

2.2. Computational Paradigms

This section takes CNN as an example for analysis. The core components of a typical CNN consist of a convolutional layer, a pooling layer, a fully connected layer and a nonlinear activation layer.

2.2.1. Convolution Layer

In CNNs, the convolution operation is the core and foundation of computation [8]. These operations typically employ a convolution kernel, or filter, that slides across the input feature map [8]. With each slide, the dot product of the kernel and the local input feature map is computed, producing the output feature map. Despite its importance, the significant computational and parameter demands of traditional convolution operations can limit the model's efficiency and range of use. As a result, various optimized methods of convolution operations have been introduced to reduce computational and parameter needs, enhancing the model's efficiency. Examples of these methods include Separable Convolution, Fire Module, and Channel Shuffle.

Separable Convolution: The basic idea of Separable Convolution is to decompose traditional convolution operations into two independent steps: Depthwise Convolution (DWC) and Pointwise Convolution (PWC) [7].

Depthwise Convolution: DWC involves performing convolution operations on each channel of the input data independently. A unique convolution kernel (for example, 3x3) is used to process each input channel, resulting in an independent output feature map for each channel. This step primarily focuses on extracting spatial information and does not involve the interaction or integration of information across different channels.

DOI: 10.54254/2755-2721/51/20241212

Pointwise Convolution: PWC uses a 1x1 convolution kernel to process the output of the DWC. This 1x1 convolution kernel performs convolution operations across all input channels, integrating information from multiple channels into a single output channel. This step primarily deals with the integration of information across channels and does not involve further processing of spatial information.

The advantage of separable convolution lies in its decomposition of traditional convolution operations into two simpler steps, significantly reducing computational complexity and the number of parameters. For instance, MobileNet achieves a Top-1 accuracy of 70.6% on ImageNet, with only 1/30 of the computational complexity of VGG16, making it suitable for operation in resource-constrained environments such as mobile and embedded devices [9].

Fire Module: The Fire Module is a design consisting of one Squeeze layer and two Expand layers [7]. The Squeeze layer serves to reduce the number of channels, thereby effectively lowering the model's complexity and mitigating the risk of overfitting. The Expand layers utilize a combination of 1x1 and 3x3 convolutions to restore the spatial information and the number of channels in the features. The Fire Module significantly reduces the number of parameters and the computational load while maintaining model performance, making it of broad application value in scenarios such as model lightweight and mobile device deployment. For example, SqueezeNet achieves comparable accuracy to AlexNet, with only 1/50 of the number of parameters [10].

Channel Shuffle: Channel Shuffle is a strategy to improve the performance of group convolutions in DNNs. In group convolutions, the learning capacity may be limited since the information in each group is independent. Channel Shuffle enhances model performance by dividing the output feature map channels into multiple groups, performing cyclic shift operations within each group, and then concatenating all groups to form a new feature map [7]. This process facilitates communication of features between different groups. This strategy is widely applied in lightweight deep learning models such as ShuffleNet V2, effectively reducing model computational complexity and the number of parameters while preserving model performance [11].

2.2.2. Activation Function

Activation functions play a key role in introducing non-linearity into neural networks, enabling them to handle complex tasks. Common activation functions such as ReLU (Rectified Linear Unit) are efficient and can alleviate the vanishing gradient problem, but they also suffer from the "dead neuron" issue [12]. Leaky ReLU, an improved version of ReLU, mitigates the "dead neuron" problem, but the improvement is not significant. The Sigmoid function is characterized by its "on/off" property, but it also has the vanishing gradient problem. Although the Tanh function can achieve zero-centered output, it cannot avoid the vanishing gradient problem [12]. Despite the advantages and disadvantages of various activation functions, ReLU is generally considered the most suitable activation function for CNNs due to its computational efficiency and non-saturation.

2.2.3. Pooling Layer

The pooling layer plays a pivotal role within the network architecture. Its primary functions encompass diminishing the spatial dimensions of the features, thereby reducing the number of parameters and computational demands imposed on the network. Furthermore, it acts as a safeguard against overfitting while endowing the model with a measure of translation invariance [8].

2.2.4. Fully Connected Layer

The nodes within the fully connected layer establish connections with every node in the preceding layer, affording them the capacity to discern intricate patterns spanning from the input to the output layers [5]. In the context of Convolutional Neural Networks (CNNs), it's customary for the fully connected layer to occupy the final position within the network architecture, assuming the pivotal role of conducting high-level inferences by leveraging the features that have been extracted.

DOI: 10.54254/2755-2721/51/20241212

3. Hardware platforms for deep learning

The previous chapter delineated the fundamental structure of convolutional neural networks and outlined the challenges faced relating to memory access efficiency and computational performance. In an endeavour to address these issues, this chapter commences with an introduction to four principal data flow strategies aimed at optimizing memory access methods as shown in figure 1. Subsequently, an in-depth analysis is presented on how FPGA-based and CGRA-based accelerators realize efficient computation in neural network applications.

3.1. Dataflow scheme for accelerator

Weight Stationary (WS): This strategy involves fetching the weight from the Global Buffer and stores it in the Processing Element's (PE) register files [7]. The operation is performed by inputting ifmap and passing psum between PE. This approach enhances weight reusability, thereby reducing power consumption. A typical example of WS strategy is nn-X [13].

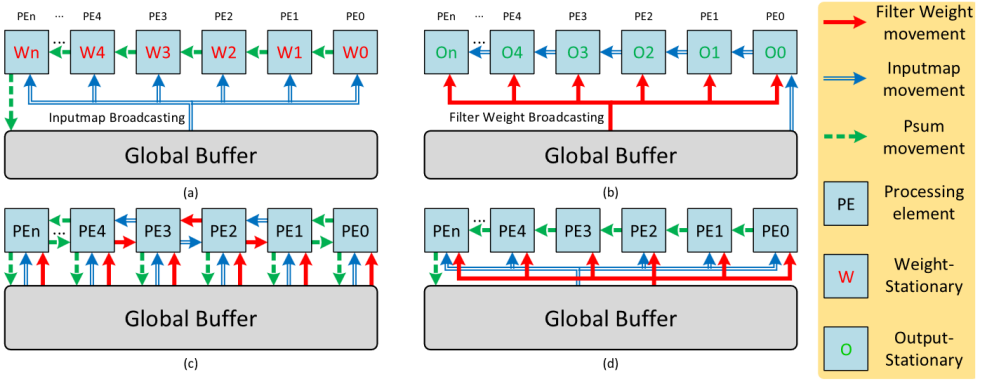
Output Stationary (OS): Contrary to the WS strategy, OS fixes the output psum in the register files, and then moves the ifmap and assigns weight to accumulate until the final sum is obtained [7]. Memory-centric accelerator is the use of OS strategy, by reducing the storage requirements of partial sum, successfully improving the memory utilization [14].

No Local Reuse (NLR): In this strategy, the data cannot be reused locally, resulting in a lot of data movement [7]. It can effectively reduce the area occupied by the register in the PE. Although it is less widely used in CNNs, there are some specific applications, such as graph rendering or some specific algorithms, where the NLR strategy may play its advantage.

Row Stationary (RS): This strategy allocates the ifmap and weight of the whole row to the register files of the PE unit, and calculates them to generate psums [7]. These psums will be further accumulated on the shortest moving path to obtain the ofmap finally. Eyeriss v2 uses RS data streams to improve data reuse and PE utilization [15].

It is important to note that stationary data can be stored inside the PE or in relevant storage elements outside the PE. The point is that this data remains stationary over a series of computations and is reused multiple times, reducing the need for data transfer, and improving performance and energy efficiency.

The dataflow movements for CONV is shown in figure 1.



(a) Weight-Stationary (WS) dataflow, (b) Output-Stationary (OS) dataflow, (c) No-Local-Reuse (NLR) dataflow, and (d) Row-Stationary (RS) dataflow

Figure 1. Dataflow movements for CONV [16].

3.2. FPGA-Based Accelerators for DNNs

3.2.1. Basic Principle

FPGA (Field-Programmable Gate Array) is a fine-grained Programmable Logic Blocks device based on Look-Up Tables (LUTs). It is composed of the adjacent logic block (LB), connect box (CB), and switch box (SB), as shown in figure 2 [17].

DOI: 10.54254/2755-2721/51/20241212

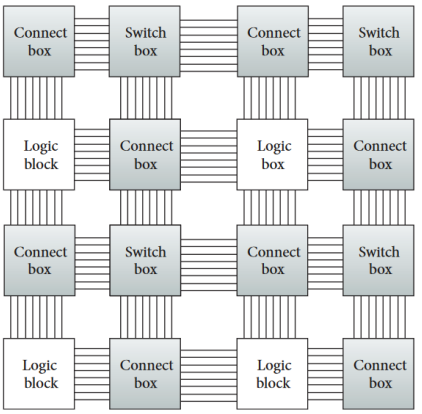


Figure 2. Fine-grained reconfigurable architecture [17].

That is, FPGA allows designers to directly manipulate and control data at the bit level, for example, FPGA uses LUTs in LB to perform basic Boolean logic operations and control data flow paths through SB that can be programmed to implement highly customized digital logic functions. This fine-grained programmability makes FPGA very advantageous when dealing with computationally intensive tasks such as neural networks, which often require massive parallel computation and flexible data flow control [7]. In addition, FPGA can also be reprogrammed at runtime, allowing them to dynamically adapt to different computational tasks, which is not possible with many other hardware devices. It is worth noting that in most FPGA-based architectures, multiple LUTs need to be combined into a higher-level abstraction known as the processing element (PE). IP cores or higher-level modules are often used to encapsulate some of the computational functionality. However, PE in CGRA-based architecture is composed of predefined actual hardware units. In the next section, several cases of FPGA-based combined with Dataflow schemes are introduced to analyze how to improve memory access efficiency and computing performance.

3.2.2. Case Analysis

nn-X adopts a parallel computing architecture with low power consumption based on the main processor and coprocessor, as shown in figure 3 [13].

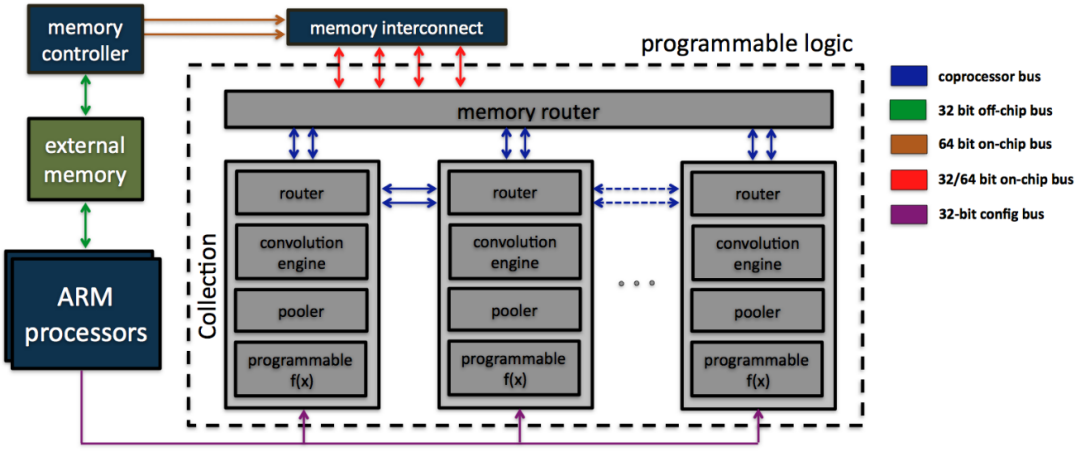


Figure 3. The Architecture of nn-X system [13].

The main processor parses the network structure, generates micro-code instructions, and coordinates data transfer and external storage. The coprocessor includes pipelined parallel-connected custom computing units, including convolutions, pooling, and non-linear programmable units. The core network

DOI: 10.54254/2755-2721/51/20241212

operations can be efficiently split into a large number of small-scale parallel operations. In addition, the fixed weight parameters are cached on the chip, while the activation data flows through each computing unit, realizing the weight stationary data flow architecture to avoid redundant memory accesses. nn-X is a FPGA-based design, and the experiment platform is a joint system of XC7Z045 chip and ARM processor, which makes full use of the parallel computing advantage of FPGA. Experimental results show that in a variety of vision tasks, nn-X has achieved more than 100 times speedup, and the computational efficiency is far ahead of other platforms.

A memory-centric accelerator adopts a flexible memory hierarchy, which includes a series of SIMD Multiply-accumulate (MACC) processing units and a programmable reuse buffer implemented on FPGA Block RAM, as shown in figure 4 [14].

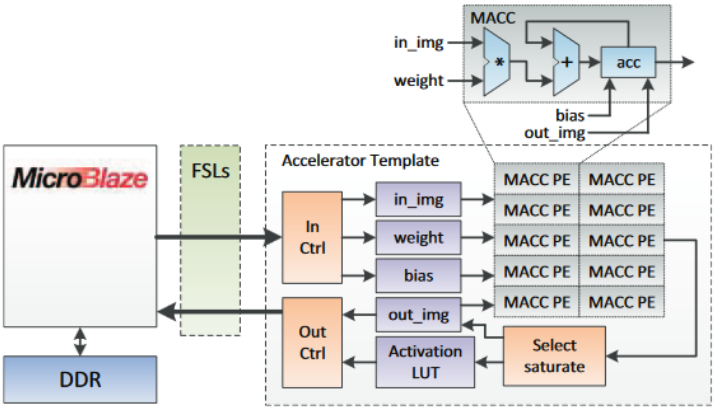


Figure 4. The Architecture of Memory-Centric Accelerator [14].

The architecture can effectively support different computing modes in CNN workloads and embodies the Output Stationary data flow architecture, that is, multiple PEs compute the results of different positions of the same output feature map in parallel, and the input feature map is shared between different output feature maps to improve the computational efficiency. A memory-oriented design flow is adopted to optimize the design, tiling and loop transformations are used to maximize buffer data reuse and reduce external bandwidth requirements. Finally, the accelerator template is implemented on the Virtex 6 FPGA development board. The results show that compared with the standard Scratchpad memory, the acceleration performance is improved by 11 times under the same FPGA resources. The FPGA resources are reduced by 13 times for the same performance.

Zhang et al. introduced the Caffeine framework, a software/hardware co-design approach aimed at optimizing CNN acceleration on FPGA, as depicted in Figure 5 [18].

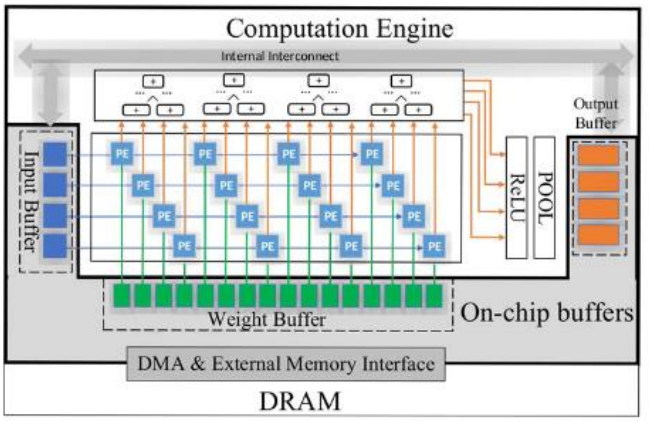


Figure 5. The Architecture of Accelerator Engine [18].

DOI: 10.54254/2755-2721/51/20241212

On the hardware side, they constructed the FPGA accelerator using systolic-like processing elements (PEs). To minimize memory access, the authors employed input-major and weight-major mappings to enhance data reuse, aligning with the data flow benefits of weight stationary designs. Furthermore, they devised a unified representation for convolutional matrix multiplication that applies to both convolutional and fully connected layers, thereby improving bandwidth utilization. The Caffeine framework was implemented on a Virtex7 690t FPGA, exhibiting an estimated speedup of 9.7 times over CPU performance and 2.2 times greater energy efficiency compared to GPUs.

3.3. CGRA-Based Accelerators for DNNs

3.3.1. Basic Principle

Coarse-Grained Reconfigurable Array (CGRA) is a coarse-grained reconfigurable computing architecture, which is connected by a programmable interconnection network to form a PEs matrix, as shown in figure 6 [19].

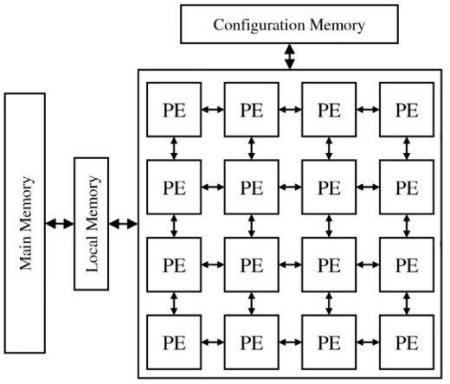


Figure 6. Coarse-grained reconfigurable architecture [19].

Unlike fine-grained, these PEs are capable of performing more complex operations than simple Boolean logic, such as integer addition or multiplication, and can be configured independently. The operation principle of CGRA is to map computing tasks to PEs by means of static scheduling, and exchange and communicate data through the interconnection network. A distinguishing feature of CGRA is its high parallelism and reconfigurability, which enables it to efficiently handle parallel computing tasks such as convolution operations in CNNs. Moreover, since CGRA can be optimized at the hardware level for specific applications, it can often outperform conventional general-purpose processors in terms of performance and energy efficiency.

3.3.2. Case Analysis

It is worth noting that many proposed hardware accelerator architectures claim to be ASIC-based, for example, Eyeriss v2, MAERI, DNPU, ISSCC19-7.4, etc [15, 20-22]. However, given the well-designed coarse-grained reconfigurability of these architectures, this category is more accurately classified as CGRA in this paper.

Lee et al. proposed Neural Processing CGRA (NP-CGRA), which was designed to better support depthwise separable convolution (DSC) in lightweight convolutional neural networks. DSC is composed of depthwise convolution (DWC) and pointwise convolution (PWC) layers, which can greatly reduce the model size and computational complexity compared with ordinary 3D convolution [23]. In order to deal with DSC efficiently, NP-CGRA adopts the output stationary (OS) mechanism. When mapping PWC, multiple rows of the input feature map flow into the PE through the horizontal bus, and the weights are broadcast through the vertical bus, so that the output feature map is generated on the PE array. In addition, NP-CGRA introduces the multiply-accumulate unit, which can complete the multiply-add operation in one cycle, and the crossbar-style memory bus to provide high-bandwidth local memory

DOI: 10.54254/2755-2721/51/20241212

access. Experimental results show that compared with the traditional CGRA, it brings 8 times and 18 times improvement of area-delay product performance for DWC and PWC on the MobileNet model, respectively.

Fan et al. proposed a stream dual-track CGRA (SDT-CGRA), which mainly includes two parts: global memory and a computing array, as shown in figure 7 [24].

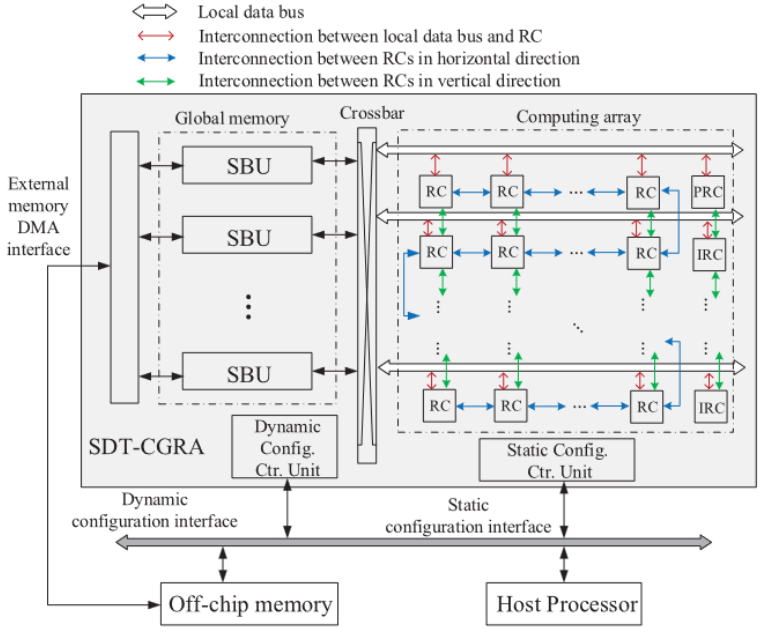


Figure 7. The Architecture of SDT-CGRA [24].

Among them, the main acceleration part contains the reconfigurable cell (RC) matrix and stream buffer units (SBU) matrix. The RC matrix consists of RC units interconnected in reverse-S fashion, each RC integrates a fixed number of multipliers and ALUs, which can efficiently perform convolution and matrix operations in parallel, and the SBU matrix is responsible for caching the input-output data streams. In addition, in order to realize special operations, small-scale interpolation RC (IRC) and power RC (PRC) are designed. This architecture takes full advantage of the flexibility of CGRA, adopts a dual-track programming model, reconfigures static configuration to build different RC operation functions according to computing requirements, and dynamically configures and schedules data flow to improve operation efficiency. Experimental results show that compared with CPU and GPU, the efficiency of the proposed architecture on multiple object inferencing algorithms is improved by tens to hundreds of times. Compared with the FPGA implementation of AlexNet acceleration, the energy efficiency is improved by 1.78 times. Therefore, the architecture design verifies that the method based on CGRA can efficiently implement object reasoning algorithms, and has the advantages of flexibility and computation intensive.

MAERI is a DNN accelerator architecture based on reconfigurable interconnected networks. It consists of configurable multiplier switches, adder switches, and reconfigurable interconnection networks for data allocation and reduction. MAERI can support various DNN topologies and mappings by appropriately configuring the switches. Its adder switch uses a newly proposed Enhanced Reduction Tree (ART), which can efficiently map reduction operations of any size. In the convolutional layer computation, each multiplier switch implements weight stationary, the row map on the multiplier switch implements row stationary, and each configurable ART implements output stationary, the combination of these three optimizes throughput and energy efficiency. MAERI can also achieve efficient mapping by building custom-sized virtual neurons in recurrent and fully connected layers as well as sparse networks. Compared to accelerators with fixed interconnection networks, MAERI demonstrates 8-459%

DOI: 10.54254/2755-2721/51/20241212

higher cell utilization across multiple instances. The comprehensive results show that compared with Eyeriss and Systolic Array, MAERI can provide better area utilization under the same computing resources. It can accommodate 2-7 times more computing units under the same chip area constraint.

4. Comparative analysis

In the previous section, some representative FPGA-based and CGRA-based accelerators have been introduced and analysed in detail. These cases cover various dataflow strategies such as Weight Stationary (WS), Output Stationary (OS), Row Stationary (RS), etc. as well as different hardware platforms and deep learning models. Based on these cases, this chapter presents a detailed comparative analysis between FPGA and CGRA in terms of throughput, Energy Efficiency, and Area, as shown in table 1.

Table 1. Comparison among accelerators implemented on different hardware platforms.

Architecture, Year	Platform	Edge Device	Dataflow	Clock (MHz)	Area (mm^2)	Throughout (OPS)	Power (W)	Energy Efficiency (OPS/W)
nn-X [13] 2014	FPGA-Based	Xilinx ZC706	WS	142	N/A	227 G	8 W	25 G
Memory-Centric Accelerator [14] 2013	FPGA-Based	Xilinx ML-605 Virtex 6	OS	150	N/A	17 G	N/A	N/A
Caffeine [18] 2016	FPGA-Based	Virtex7 690t	WS	150	N/A	354 G	26	13.6 G
Angle Eye [25] 2018	FPGA-Based	Zynq XC7Z045	WS, IS, OS	150	N/A	137 G	9.63	14.2 G
FP-DNN [26] 2018	FPGA-Based	Stratix-V GSMD5	N/A	150	N/A	364.4 G	25	14.57 G
NP-CGRA [23] 2021	CGRA-Based	TSMC 28 nm CMOS	OS	500	2.14	64 G	N/A	N/A
SDT-CGRA [24] 2018	CGRA-Based	SMIC 55nm CMOS	N/A	450	5.19	77.4 G	1.526	50.78 G
MAERI [20] 2018	CGRA-Based	TSMC 28 nm CMOS	RS, OS WS	200	6	N/A	0.3	N/A
Eyeriss v2 [15] 2019	CGRA-Based	TSMC 65 nm CMOS	RS	200	32	153.6 G	1.65	253.2 G
DNPU [21] 2018	CGRA-Based	65nm IP8M CMOS	N/A	200	16	1.2 T	0.279	3.9 T
ISSCC19-7.4 [22] 2019	CGRA-Based	65nm CMOS	WS, IS	200	16	206 G	0.0024-0.196	2.16 T

OS = output stationary, IS = input stationary, RS = row stationary, WS = weight stationary

4.1. Throughout

Throughput is a key metric to evaluate the performance of accelerators, which reflects the amount of operations that can be processed per unit time. As shown in figure 9, it can be seen that FPGA-based accelerators achieve high throughput, such as 227GOPS for nn-X, 364.4GOPS for FP-DNN, and 354GOPS for Caffeine.

DOI: 10.54254/2755-2721/51/20241212

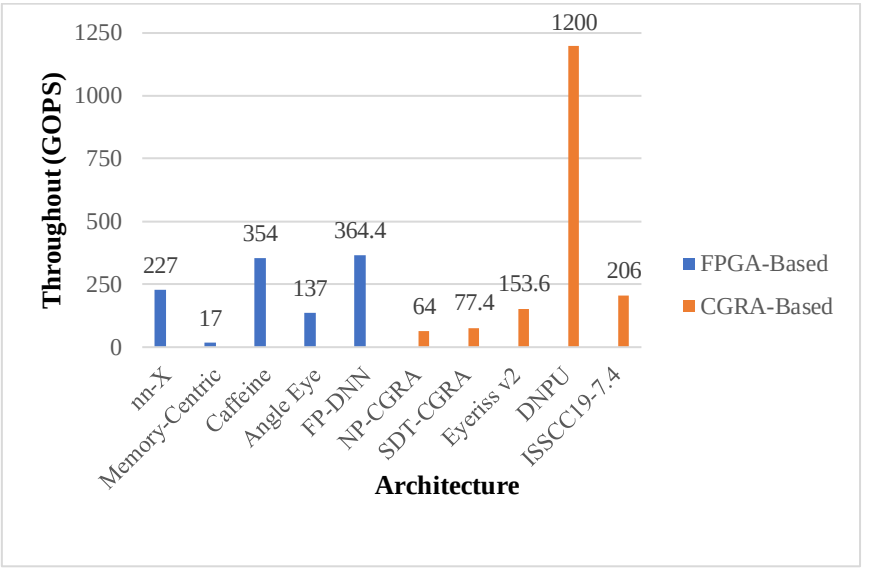


Figure 9. Throughout (Photo/Picture credit: Original).

This is mainly due to the fact that FPGAs can perform large-scale bit-level parallel processing. In comparison, CGRA-based accelerators have a low throughput, such as 64GOPS for NP-CGRA, 77.4GOPS for SDT-CGRA, and 153.6GOPS for Eyeriss v2. This does not mean that CGRA performs poorly, but rather because its coarse-grained property makes it difficult for bit-level operations and flexible data-flow control, which limits its throughput performance on certain tasks that require higher flexibility.

4.2. Energy Efficiency

As shown in figure 10, in terms of power consumption and energy efficiency, CGRA outperforms FPGA as a whole.

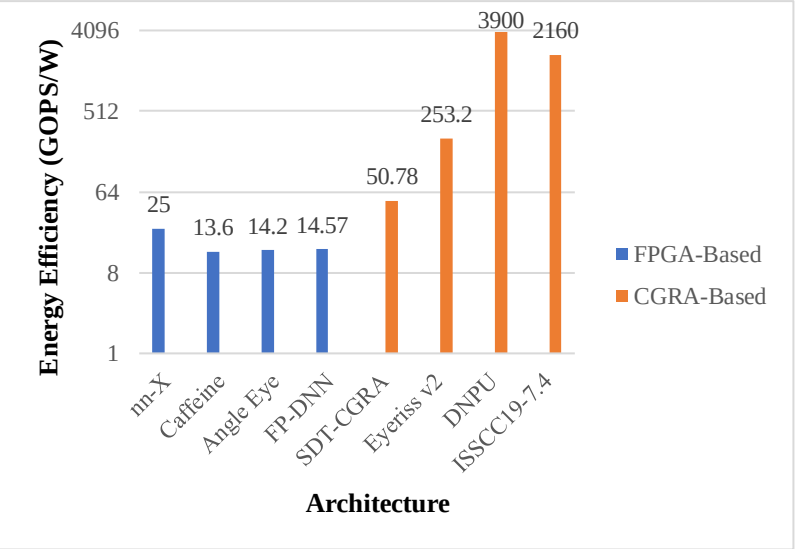


Figure 10. Energy Efficiency (Photo/Picture credit: Original).

For example, CGRA-Based SDT-CGRA, Eyeriss v2, and DNPU have the energy efficiency of 50.78 GOPS/W, 253.2 GOPS/W, and 3.9 TOPS/W, respectively, which are significantly higher than those of FPGA-Based accelerators such as nn-X, Caffeine, and DNPU. Angle Eye is 25 GOPS/W, 13.6GOPS/W

DOI: 10.54254/2755-2721/51/20241212

and 14.2 GOPS/W, respectively. The reason is that CGRA reduces the static and dynamic power consumption by using coarse-grained custom design and interconnection network optimization to reduce switching activities and redundant resources. In addition, CGRA can be easily customized to use a variety of data flow strategies to utilize data reuse and reduce data movement, thus reducing energy consumption more effectively. For example, MAERI improved resource utilization by 8-459%, which reduces redundant operations [20]. However, FPGA is more versatile, has more redundancy, and is difficult to optimize for specific applications, so its energy efficiency is usually lower than that of CGRA.

4.3. Area

The area is an important indicator to measure the hardware resource occupation of accelerators. Since the construction of an FPGA consists of programmable logic blocks and programmable interconnects, and these elements can be dynamically configured according to the requirements of a specific task, the area depends on the specific hardware configuration and task, which makes the area of an FPGA difficult to calculate directly. In contrast, the structure of CGRA is relatively fixed and each processing element and interconnect is predefined, so the number of required processing elements and interconnects, and hence area, can be calculated more directly based on the requirements of a specific task. In these cases, CGRA-based accelerators such as NP-CGRA and SDT-CGRA have a smaller area of 2.14 mm^2 and 5.19 mm^2, respectively, which indicates that they are more efficient in hardware resource usage.

5. Conclusion

This article analyzes and compares the application of FPGA and CGRA in the field of deep learning acceleration in detail. First, we discuss the structure of convolutional neural networks and common optimization techniques for convolutional layers. Then starting from optimizing data flow, different storage access modes are discussed to reduce the amount of memory access. Then it introduces the architectural advantages of FPGA and CGRA, analyzes relevant typical cases, and discusses their technical means of using various data flow modes and custom architectures to optimize DNN. Finally, we compare the throughput, energy efficiency, and area of the two types of accelerators. The conclusion is that FPGAs generally exhibit higher throughput when handling computationally intensive tasks that require massive parallel computing and flexible data flow control. However, for applications that have strict requirements on energy consumption and hardware resource usage, CGRA may be a better choice due to its advantages in energy efficiency and area.

Overall, FPGA and CGRA, as emerging deep-learning acceleration platforms, have their own advantages and can play a huge role in different application scenarios. The current challenge lies in continuing to improve their ability to support larger and more complex networks. Future research can consider the joint optimization of FPGA and CGRA, and cooperate with algorithm innovation to obtain greater acceleration effects.

References

[1] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436-444.
[2] Nurvitadhi E, Sheffield D, Jaewoong Sim, et al. Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC. 2016 International Conference on Field-Programmable Technology (FPT). IEEE, 2016.
[3] Peccerillo B, Mannino M, Mondelli A, et al. A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives. Journal of Systems Architecture, 2022, 129: 102561.
[4] Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks, 2015, 61: 85-117.
[5] Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press, 2016.
[6] Gu J, Wang Z, Kuen J, Et Al. Recent advances in convolutional neural networks. Pattern Recognition, 2018, 77: 354-377.

DOI: 10.54254/2755-2721/51/20241212

[7] Shuvo Md M H, Islam S K, Cheng J, et al. Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review. Proceedings of the IEEE, 2023, 111(1): 42-91.
[8] Dumoulin, V., Visin, F. (2016). A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285.
[9] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. ArXiv. /abs/1704.04861
[10] Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360, 2016.
[11] Ma N, Zhang X, Zheng H T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design. Proceedings of the European conference on computer vision (ECCV). 2018: 116-131.
[12] Xu B, Wang N, Chen T, et al. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853, 2015.
[13] Gokhale V, Jin J, Dundar A, et al. A 240 G-ops/s Mobile Coprocessor for Deep Neural Networks. 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops. IEEE, 2014.
[14] Peemen M, Setio A A A, Mesman B, et al. Memory-centric accelerator design for Convolutional Neural Networks. 2013 IEEE 31st International Conference on Computer Design (ICCD). IEEE, 2013.
[15] Chen Y H, Yang T J, Emer J S, et al. Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2019, 9(2): 292-308.
[16] Zhang B, Gu H, Wang K, et al. A Novel CONV Acceleration Strategy Based on Logical PE Set Segmentation for Row Stationary Dataflow. IEEE Transactions on Computers, 2021: 1-1.
[17] Chattopadhyay A. Ingredients of Adaptability: A Survey of Reconfigurable Processors. VLSI Design, 2013, 2013: 1-18.
[18] Zhang C, Sun G, Fang Z, et al. Caffeine: Toward Uniformed Representation and Acceleration for Deep Convolutional Neural Networks. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 38(11): 2072-2085.
[19] Aliagha E, Gohringer D. Energy Efficient Design of Coarse-Grained Reconfigurable Architectures: Insights, Trends and Challenges. 2022 International Conference on Field-Programmable Technology (ICFPT). IEEE, 2022.
[20] Kwon H, Samajdar A, Krishna T. Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects. ACM SIGPLAN Notices, 2018, 53(2): 461-475.
[21] Shin D, Lee J, Lee J, et al. 14.2 DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks. 2017 IEEE International Solid-State Circuits Conference (ISSCC). IEEE, 2017.
[22] Kim C, Kang S, Shin D, et al. A 2.1TFLOPS/W Mobile Deep RL Accelerator with Transposable PE Array and Experience Compression. 2019 IEEE International Solid- State Circuits Conference - (ISSCC). IEEE, 2019.
[23] Lee J, Lee J. NP-CGRA: Extending CGRAs for Efficient Processing of Light-weight Deep Neural Networks. 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021.
[24] Fan X, Wu D, Cao W, et al. Stream Processing Dual-Track CGRA for Object Inference. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2018, 26(6): 1098-1111.
[25] 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, 2018, 37(1): 35-47.
[26] Guan Y, Liang H, Xu N, et al. FP-DNN: An Automated Framework for Mapping Deep Neural Networks onto FPGAs with RTL-HLS Hybrid Templates. 2017 IEEE 25th Annual

DOI: 10.54254/2755-2721/51/20241212

International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, 2017.