Systematic analysis of FPGA-based hardware accelerators for convolutional neural networks

Fangrong Zhang

School of Electrical Engineering and Information, Southwest Petroleum University, Chengdu, 610500, China

202131073127@stu.swpu.edu.cn

Abstract. In the modern era, machine learning stands as a pivotal component of artificial intelligence, exerting a profound impact on various domains. This article delineates a methodology for designing and applying Field Programmable Gate Array (FPGA) based hardware accelerators for convolutional neural networks (CNNs). Initially, this paper introduces CNNs, a subset of deep learning techniques, and underscore their pivotal role in artificial intelligence, spanning domains such as image recognition, speech processing, and natural language understanding. Subsequently, we delve into the intricacies of FPGA, an adaptable logic device characterized by high integration and versatility, elucidating our approach to creating a hardware accelerator tailored for CNNs on the FPGA platform. To enhance computational efficiency, we employ technical strategies like dual cache structures, loop unrolling, and loop tiling for accelerating the convolutional layers. Finally, through empirical experiments employing YOLOv2, and validate the efficacy and superiority of our designed hardware accelerator model. This paper anticipates that in the forthcoming years, the methodology and research into FPGA-based CNN hardware accelerators will yield even more substantial contributions, propelling the advancement and widespread adoption of deep learning technology.

Keywords: Hardware Accelerator, CNN, FPGA, YOLOv2.

1. Introduction

In this age of data inundation, deep learning, a significant branch of artificial intelligence, has produced remarkable outcomes in a variety of areas, such as image recognition, speech processing, natural language processing, and more. The past few years have seen a great deal of attention drawn to the convolutional neural network (CNN) from both industry and academia, due to its remarkable accomplishments in many domains, including computer vision and natural language processing [1]. This network is one of the most influential in the realm of deep learning. The Field Programmable Gate Array (FPGA) has been widely studied and utilized by researchers and engineers due to its versatility, high degree of dependability, rapid performance, and capacity for reconfiguration.

This paper aims to review the design methods and applications of FPGA-based hardware accelerators for convolutional neural networks. Introducing the fundamentals and evolution of convolutional neural networks, as well as the features and uses of FPGA, this paper shall proceed. At the same time, it will also focus on a FPGA-based convolutional neural network hardware accelerator design method, through the dual cache structure, loop unrolling and loop tiling strategy and other technical means to achieve the hardware acceleration of the convolutional layer and improve the computing performance. Finally, the impact of different quantization precision on hardware resource consumption is analysed, and the process of hardware design and verification is explored. It is expected that this review will provide an outlook on the future of FPGA hardware accelerator design methods for convolutional neural networks, and will be helpful and inspirational to related disciplines.

2. Theoretical basis and development history

2.1. CNN Theory Analysis

2.1.1. Basic definition of CNN algorithm

The Convolutional Neural Network (CNN) is a prime example of deep learning, which utilizes a deep feedforward neural network to carry out convolutional computations [2-3]. Neural networks are a set of algorithms that are loosely designed to mimic the structure of the human brain and are used to recognize patterns [4]. Machine perception systems enable neural networks to interpret sensor data and perform operations like labelling or clustering raw inputs. Converting real-world data, such as images, sounds, texts, and time series, into numerical form is necessary for neural networks to recognize patterns [5].

A convolutional neural network is composed of three distinct layers: a convolutional layer, a pooling layer, and a fully connected layer. Figure 1 displays the convolutional neural network.

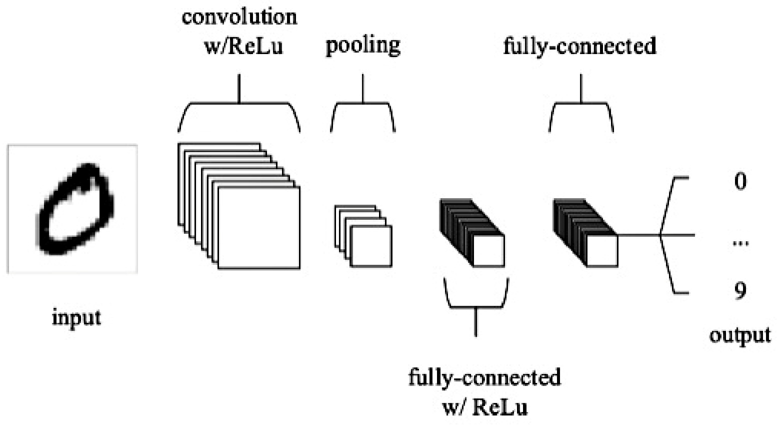

Figure 1. Convolutional neural network (Photo/Picture credit: Original).

2.1.2. Introduction of each Layer

(1) Convolution layer: This layer mainly uses convolution kernels to convert input data into output data through convolution calculation. The complexity of the convolutional layer, with its multiplication and addition calculations, makes it a highly demanding task for computing power. Moreover, taking the convolution processing of images as an example, each pixel in the feature map and between different feature maps are independent, that is, the calculation of the feature map and different pixels in the feature map can be parallelized, so the calculation in the convolution layer is usually parallelizable [6].

(2) Pooling layer: Following the convolutional layer's feature extraction, the output feature map is given to the pooling layer for feature selection and data filtering. The preset pooling function in the pooling layer's code replaces a single point's feature map result with the statistics from that point's surrounding regions. Similar to the convolution kernel scanning feature map, which is controlled by the pooling size, step size, and fill, the pooling layer chooses the pooling region [1].

(3) Conventional feedforward neural networks have hidden layers that are analogous to the fully connected layers in convolutional neural networks. The convolutional neural network's fully connected layer is the last part of the hidden layer and only dispatches signals to other fully connected layers. In fully connected layers, feature maps are expanded into vectors and subjected to activation functions, losing their spatial topology in the process [1].

2.2. Definition and advantages of FPGA

A programmable logic device, the FPGA (Field Programmable Gate Array), allows users to modify the way hardware functions are carried out as needed. It is made up of a number of programmable interconnect resources and programmable logic blocks that can be used to create different digital circuits and systems. FPGA has a number of benefits over application specific integrated circuits (ASIC). After the design is finished, FPGA can be reconfigured to include new features.

FPGA is a kind of programmable signal processing device, which has been concerned by people from all walks of life. In addition, FPGA has the following features:

(1) The unique flexibility of FPGA programming allows it to execute any logical function that an ASIC can, and its ability to be altered at any moment can reduce the cost and risk of the product.

(2) High degree of integration: FPGA internal circuit size is very small, the interconnect line is short, the distributed capacitance is small, the power consumption required by the drive circuit is greatly reduced. The FPGA chip's very brief wiring can drastically reduce the delay time and not be affected by outside interference, which is highly advantageous for accelerating, reducing the size and weight of electronic items, cutting down on power usage, and enhancing the functioning speed of products [7].

(3) Good reliability: FPGA usually have built-in fault tolerance functions that can detect and correct errors. Error correcting codes can be employed to identify and rectify bit mistakes during transmission, for instance. During the production process, the FPGA chip undergoes rigorous reliability testing to ensure its proper operation. These tests can detect and rule out potential manufacturing defects, thus improving the reliability of the FPGA.

(4) Fast operation: The essentially instruction-free, shared-memory architecture of FPGA computing speed is responsible for its speed. The registers and on-chip memory (BRAM) in the FPGA are controlled by their own control logic, without any unnecessary caches, to save state.

2.3. Development History and Current Situation

Invention period: 1984-1992: The first FPGA was created by Xilinx in 1984, but the term "FPGA" wasn't coined until Actel did so in 1988. In the following 30 years, the devices, also known as FPGA, increased their capacity by more than 10,000 times, their speed by 100 times, and their cost and energy consumption by more than 10,000 times per unit of function.

Expansion period: 1992-1999: At that time, IC foundries realized that FPGAs were an ideal enabler for process development, and FPGA became a powerful tool to eliminate barriers to process development. As long as transistors and wires are produced using the new process, SRAM-based FPGA can be made by founders. As the process advances, the amount of transistors rises, the cost per function diminishes, and the magnitude of the most expansive FPGA grows [8].

Accumulation period: 2000-2007: The 2000s saw a change in design features. Exceptionally large designs (complete subsystems) can be accommodated by large FPGAs. FPGA users are now required to make the FPGA design compliant with system standard requirements, not just implementing logic. Communication standards in terms of signals and protocols are the focus of these standards, which can be utilized to connect external components or implement internal module communication. Processing standards have allowed FPGA to play an increasingly important role in computation-intensive applications. At the end of the accumulation era, FPGA is not only a gate array, but also a complex function set integrated with programmable logic [9]. The FPGA becomes a system.

System period: After 2008: FPGA is increasingly integrating system modules, including high-speed transceivers, memories, DSP processing units, and complete processors, to solve system design problems. Integration of important control functions includes bitstream encryption and verification, mixed signal processing, power and temperature monitoring, and power management. The Zynq All-Programmable device reflects all of these features. At the same time, devices also drive the development of tools. System FPGA requires efficient system programming language, now OpenCL and C language can be used to program in a software-like process [10].

To sum up, the timeline of FPGA evolution history is shown in figure 2.

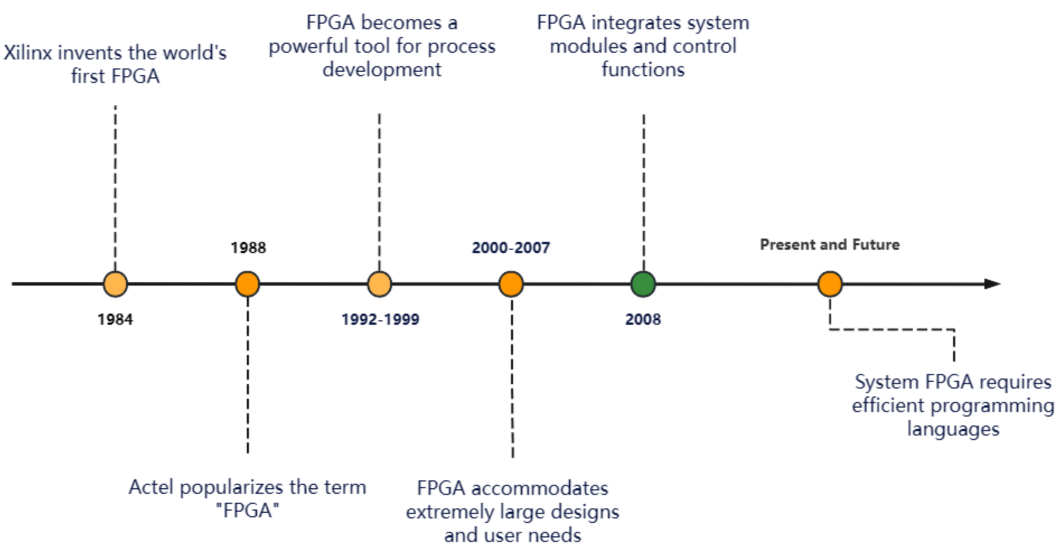
Xilinx invents the world's first FPGA
FPGA becomes a powerful tool for process development
FPGA integrates system modules and control functions
1988
2000-2007
Present and Future
1984
1992-1999
2008
System FPGA requires efficient programming languages
Actel popularizes the term "FPGA"
FPGA accommodates extremely large designs and user needs

Figure 2. Timeline of FPGA evolution history (Photo/Picture credit: Original).

3. Technical analysis

3.1. Design and Implementation

As an example, the FPGA chip's weight buffering and input-output feature mapping consume too much resources, making it unsuitable for deployment on small and medium-sized FPGAs. To improve the inference speed of convolutional neural networks, a hardware acceleration circuit based on FPGA is designed, following the experimental design mentioned in the "FPGA-Based Convolutional Neural Network Hardware Accelerator Design" [11]. In the example case, the researchers used YOLOv2 (You Only Look Once version 2) network model. YOLOv2 is a deep learning network model for object detection. It is an improvement and extension of the YOLO (You only Look once) model. However, the improvement of YOLOv2 is that it can ensure the accuracy of object detection while maintaining high speed, so it is very in line with the goals and needs of researchers. The YOLOv2 is shown in figure 3.

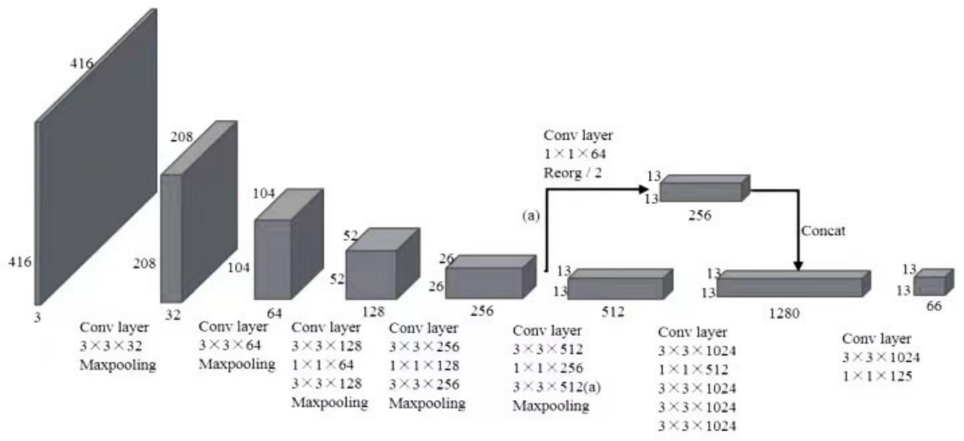

Figure 3. YOLOv2 [11].

3.2. Case Scenario Analysis

3.2.1. Overall architecture design of hardware accelerator

In the inference stage of convolutional neural network, there is a serial structure between the layers of the network. This design employs a double cache structure, as depicted in Figure 4 to minimize the transmission delay of data reading or writing. The hardware accelerator of this design capitalizes on the inter-layer pipeline and intra-layer parallel computing features of convolutional neural networks, with each network layer's output being the input of the following layer, resulting in numerous parallel computations.

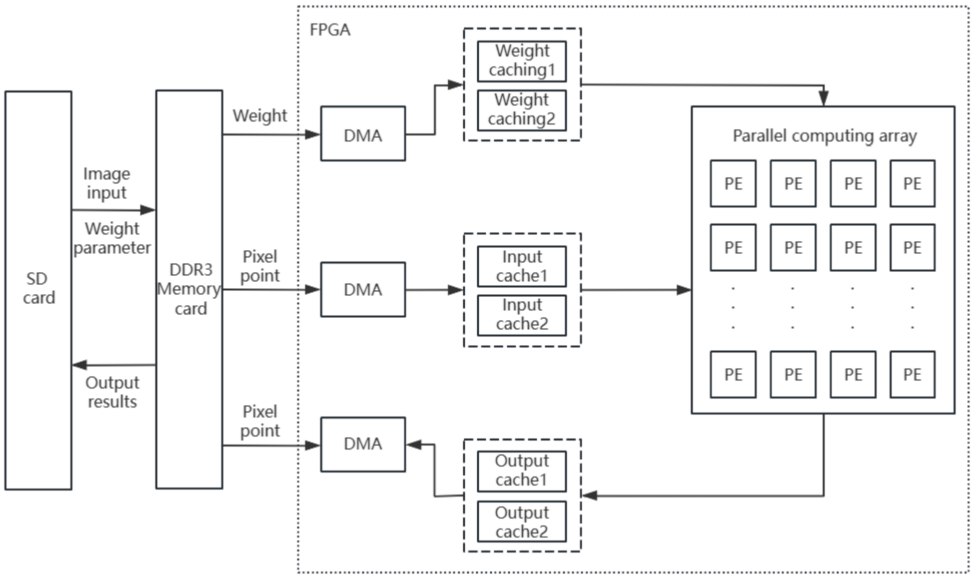

Figure 4. Hardware Accelerator Architecture [11].

3.2.2. Convolutional layer hardware acceleration unit design

The YOLOv2 network model's convolution layer is composed of three components: calculation of convolution, Batch Normalization (BN), and Leaky ReLU activation. For each pixel in the convolution operation's output feature map, batch normalization is a linear procedure. By fusing batch normalization with weight and bias parameters prior to the convolution calculation, this design is advantageous in reducing the amount of computation on the FPGA chip and improving the accelerator's performance.

Because the convolution operation occupies more than 90% of the computation in most CNN networks, the current design of CNN hardware accelerator mainly focuses on the hardware design of the convolution layer. This design adopts the strategy of loop unrolling and loop blocking to expand the input and output feature maps in two dimensions. In the case of reasonable consideration of hardware resources and YOLOv2 network structure, this design uses 128 parallel multipliers to realize the hardware acceleration of the convolution layer, as shown in Figure 5.

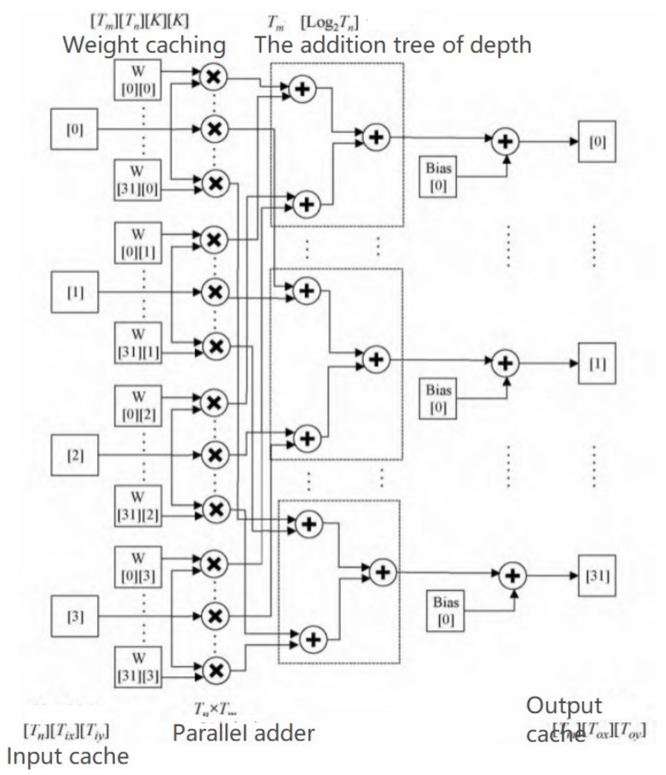

Figure 5. Diagram of convolutional layer hardware acceleration architecture [11].

The input and output ports of convolution calculation unit include three parts: feature map input port, weight parameter input port and feature map output port. Because the number of bias parameters of convolution calculation is small, they are stored in the FPGA chip in advance. During the convolution calculation, the input and output ports use ping-pong operation, which makes the data reading and writing be carried out at the same time under the same clock, which can effectively reduce the transmission delay of reading and writing data and improve the calculation speed of each PE unit, thereby improving the calculation performance of the whole hardware accelerator.

3.2.3. Double cache design

This design employs a dual cache structure is shown in Figure 6, and to significantly reduce the transmission delay of reading and writing data, thus enhancing the accelerator's performance and significantly diminishing the delay caused by data transmission. In this design, the zyng-7020 chip is used to complete the design of the accelerator. Firstly, the data set to be tested, 16-bit fixed-point quantized weight parameters and bias parameters are stored in the SD card, and then read from the SD card into the off-chip DDR3 memory, and then between the DDR3 memory and the on-chip cache, the data set is stored in the SD card. The data in the off-chip memory are read into the on-chip cache via DMA(Direct memory Access) through the HP ports of the four AXI buses. Among them, the HP port of the AXI bus is designed with double cache. The Double cache structure is shown in figure 6.

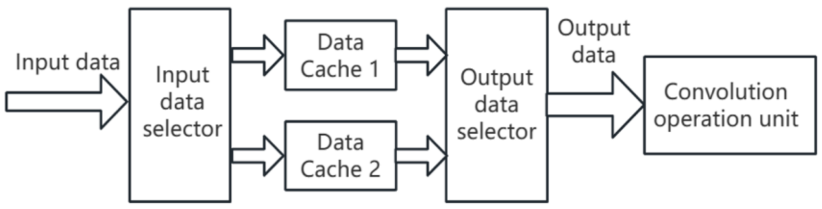

Figure 6. Double cache structure [11].

3.2.4. Fixed-point quantization and data format

In the YOLOv2 model, 32-bit floating point numbers are used for the pixel values, weight parameters and bias parameters of the feature map in each layer of the network. To simplify the YOLOv2 network model, accelerate its inference speed, and conserve the FPGA's hardware resources, this design employs a 16-bit fixed-point quantization model.

Table 1. Resource consumption of different precision [11].

Different precision operation types	DSP48E	LUT
Adder (32-bit floating point number)	2	215
Multiplier (32-bit floating point number)	3	134
Adder (32-bit floating point number)	1	48
Adder (32-bit floating point number)	1	102

As can be seen from the table 1, using 16-bit fixed-point numbers saves more than half of the hardware resources compared to using 32-bit floating-point numbers. At the same time, the actual test finds that the accuracy of the 16-bit fixed-point quantization YOLOv2 final model has almost no change.

3.3. Comparison of Technical Advantages

3.3.1. Results verification experiment

This design uses the pynq-z1 heterogeneous platform of Xilinx company, and the main control chip is zynq-7020. This design uses HLS software to complete the design of YOLOv2 network model hardware accelerator, and then imports the accelerator IP into Vivado software to complete integration and implementation. The hardware resource consumption is shown in the table 2.

Table 2. Hardware resource consumption on zynq-7020 [11].

Hardware Resource Type	Total Resource	Actual consumption	percentage
DSP	220	147	66.8%
BRAM_36K	140	87	62.5%
FF	53200	30284	28.5%
LUT	106400	38922	73.2%

In this design, when verifying the correctness of the results of YOLOv2 network accelerator, 100 images of transportation vehicles are selected from the test set of COCO dataset for verification. Images of vehicles mainly include people, bicycles, cars, motorcycles, airplanes, buses, trains, trucks, traffic lights and traffic The test results show the time statistics of reading the prediction results of the input images, including the total time, the average prediction time of 0.035338 seconds and 1.024473 seconds for each image respectively, and the input images of cars and motorcycles. Finally, denotes the proportion in the accuracy test where the motorcycle accuracy is 81% and the car accuracy is 69%.

3.3.2. Performance test experiment

In this part, the hardware accelerator designed in this paper is compared with the general CPU Core i5-4440 processor.

Through the test results, it is found that the hardware accelerator designed in this paper ensures that the accuracy is almost unchanged, the CPU computing time for a single picture with resolution of 416 x 416 is 5.89s, while the computing time of this design is shortened to 1.02s, about 5.77 times of the CPU, which greatly improves the network inference speed.

4. Conclusion

This paper reviews the design methods and application research progress of FPGA-based hardware accelerators for convolutional neural networks. At the outset, a convolutional neural network's fundamental definition and growth procedure is presented, as well as the features and uses of FPGA. A detailed description is given of the FPGA-based hardware accelerator design method for convolutional neural networks, including the specific implementation process and advantages of technical means such as dual cache structure, loop unrolling and loop tiling strategy. At the same time, the impact of different quantization precision on hardware resource consumption is analyzed, and the specific steps and skills of hardware design and verification are discussed.

Although FPGA-based design methods for convolutional neural network hardware accelerators have achieved remarkable results in improving computational performance and storage efficiency, there are still some challenges. These include how to further improve the performance of hardware accelerators to meet the needs of larger scale and more complex neural network models; And how to optimize the logical resource allocation and utilization of FPGA to reduce power consumption and cost.

Looking into the future, with the continuous development and innovation of FPGA technology, the design method of FPGA-based convolutional neural network hardware accelerator is expected to welcome more breakthroughs and progress. For example, the introduction of new FPGA architectures will provide higher computing power and storage capacity; The integration of heterogeneous computing environments will achieve more efficient data transmission and computing cooperation. The development of adaptive computing and parallel computing algorithms will further improve the flexibility and intelligence of hardware accelerators. Therefore, it is expected to achieve more important results in the design method and application research of FPGA-based convolutional neural network hardware accelerators in the future, and make greater contributions to the development and application of deep learning technology.

References

[1] Li Z, Liu F, Yang W, et al. A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems, 2021.
[2] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT press, 2016.
[3] Gu J, Wang Z, Kuen J, et al. Recent advances in convolutional neural networks. Pattern recognition, 2018, 77: 354-377.
[4] Farabet C, Couprie C, Najman L, et al. Learning hierarchical features for scene labeling. IEEE transactions on pattern analysis and machine intelligence, 2012, 35(8): 1915-1929.
[5] Bao Jun, Dong Yachao, Liu Hongzhe. A Survey on the Development of Convolutional Neural Networks. Network Application Branch of China Computer Users Association. China computer users association network application branch 2020 24th session of new technology of network and application essays, 2020: 16-21.
[6] Wu Yanxia, Liang Kai, Liu Ying et al. Progress and trend of FPGA Accelerators for Deep Learning . Chinese Journal of Computers, 2019, 42(11): 2461-2480.
[7] Zhao Min. Research on Controller Based on FPGA. Northwestern Polytechnical University, 2004.
[8] Wang C, Luo Z. A Review of the Optimal Design of Neural Networks Based on FPGA. Applied Sciences, 2022, 12(21): 10771.

[9] Parnell K, Bryner R. Comparing and contrasting FPGA and microprocessor system design and development. WP213, 2004, 1(1): 1-32.
[10] Mencer O, Allison D, Blatt E, et al. The History, Status, and Future of FPGAs: Hitting a nerve with field-programmable gate arrays. Queue, 2020, 18(3): 71-82.
[11] Huang Peiyu, Zhao Qiang, Li Yulong. Design of Hardware Accelerator for Convolutional Neural Network based on FPGA. Computer Application & Software, 2023, 40(03): 38-44.