International Journal of Computer Applications Technology and Research
Volume 12–Issue 04, 12-15, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1204.1004

A Convolutional Neural Network Accelerator Based on FPGA

Jincheng Zou
School of Communication Engineering
Chengdu University of Information Technology
Chengdu, China

Qing Tang
School of Communication Engineering
Chengdu University of Information Technology
Chengdu, China

Congcong He
School of Communication Engineering
Chengdu University of Information Technology
Chengdu, China

Abstract: This paper analyzes and studies the hardware programmable logic resources on small-scale FPGA chips, providing reasonable hardware resource support for subsequent neural network accelerator designs. A flexible 32-bit instruction set is designed for control by the Processing System (PS) on the Programmable Logic (PL) side, making motion state detection flexible and controllable. When designing the hardware side, this paper uses a resource-sharing strategy, and most of the calculation modules are designed using on-chip DSP resources to reduce the resource consumption of the calculation module. An innovative strategy of partially not caching the data between layers of the neural network is applied to reduce the demand for on-chip cache. To optimize on-chip storage space, this article partitions the limited BRAM space on the chip in a reasonable manner and improves the efficiency of on-chip data reading and writing through parallel processing, thereby improving the real-time performance of the neural network.

Keywords: FPGA; Accelerator; Neural Network; Real-time; Instruction

1. INTRODUCTION

Currently, the hardware platforms for computing or accelerating convolutional neural network algorithms mainly consist of four types: Central Processing Units (CPUs), Graphics Processing Units (GPUs), Application-Specific Integrated Circuits (ASICs), and Field-Programmable Gate Arrays (FPGAs). CPUs are general-purpose processors that have strong single-threaded performance and a large amount of cache, which is very effective for processing control flow and serialization tasks in neural network models. However, for compute-intensive tasks, CPU performance may not meet the requirements. GPUs are highly parallel hardware computing platforms designed to accelerate graphics rendering, but due to their highly parallel nature, they are also widely used to accelerate the training and inference computation of neural networks. Compared to CPUs, GPUs have stronger computing capabilities, which can greatly reduce the training and inference time of neural networks. Although GPUs have an advantage in computing power, their power consumption is relatively high. Therefore, energy consumption and heat dissipation may need to be considered when performing large-scale neural network computations, and it may be difficult to apply GPUs on embedded platforms. ASICs are customized integrated circuits specifically designed for specific applications, and their advantages lie in their high performance and low power consumption. In some neural network computations that have high requirements for performance and energy efficiency, ASICs can provide very outstanding performance. However, the design and production cost of ASICs is high, and they typically need to be designed and manufactured for specific applications. FPGAs are programmable logic devices that can implement specific logic functions through programming. They can be designed and optimized according to specific applications, and therefore, they can adapt to new neural network algorithms, models, and tasks faster than ASICs. Compared to GPUs and CPUs, FPGAs have lower latency and higher computing performance, and their power consumption is lower, making it easier to apply them on embedded platforms. Currently, accelerating neural network computations through FPGAs still faces some challenges, such as the high development cost of accelerators for specific convolutional neural network models, weak portability of accelerators designed for different FPGA models, and relatively few open-source materials related to accelerating neural network computations using FPGAs. These problems to some extent hinder the development of FPGAs as accelerators for neural network computations.

This article studies and designs a hardware accelerator for a convolutional neural network model that is used for computing motion state detection in a software-hardware collaborative manner. The accelerator has the characteristics of flexibility, high scalability, and low power consumption, which makes it easy to apply to FPGAs with limited hardware resources. A set of instruction sets based on this hardware computing platform is designed so that users can flexibly implement various convolution calculations through different instructions. This article optimizes the hardware design of the accelerator for a specific convolutional neural network model used for motion state detection, using on-chip DSP resources to build multiplier arrays and accumulators to complete convolution and pooling calculations, and designs an activation function calculation module to enable it to run smoothly on FPGA platforms with limited resources. Special hardware design is also carried out for the reading of image data by the hardware computing platform, with limited on-chip BRAM resources allocated as on-chip memory, and the on-chip cache is divided into multiple subspaces.

2. THE DEVELOPMENT OF FPGA-BASED NEURAL NETWORK ACCELERATORS

In 1996, Cloutier et al. first used FPGA to perform calculations for convolutional neural networks. However, due to the limited amount and variety of resources on the FPGA at that time, the speed of computing convolutional neural networks was very slow. In 2015, Microsoft researchers used Intel's Stratix 10 FPGA to accelerate the convolutional neural network inference of the deep learning platform Caffe. They

International Journal of Computer Applications Technology and Research
Volume 12–Issue 04, 12-15, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1204.1004

used a system called Project Brainwave, which uses a large number of FPGAs to achieve low-latency neural network inference. On the ImageNet dataset, the system achieved higher inference performance than GPUs and has been widely used in fields such as speech and image recognition. In 2022, the National Space Science Center of the Chinese Academy of Sciences proposed a convolutional parallel acceleration scheme based on FPGA to improve the speed and energy efficiency of convolutional neural networks running on resource- and power-limited embedded platforms. They used the fusion of convolutional layers and batch normalization layers to reduce the complexity of computation, and achieved a peak computing performance of 52.56 GFLOPS on the ZCU104 platform. The performance is 4.1 times that of CPUs, and the energy consumption is only 9.9% of GPUs..

3. ACCELERATOR BASED ON FPGA

3.1 Convolution Calculation Module

This article presents three parallel multiplier arrays designed for simultaneously computing convolution operations on three channels in a convolutional neural network. Each parallel multiplier array consists of 25 8-bit width multipliers generated by DSPs, with a one-clock cycle delay in the output of the multiplier. To prevent overflow in the output results, the multiplier's output width is set to 16 bits. With this design, the parallel computing module can output the results of 75 multiplication operations in two clock cycles at the highest speed.

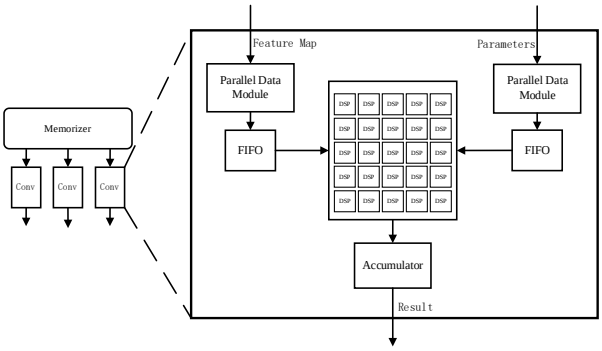

Figure. 1 Convolution computing module

In the convolution operation of a neural network, after completing the multiplication operation of a convolution kernel, the output results of the multiplier need to be accumulated to obtain a complete output result of the convolution kernel. This article designs a unified accumulator structure using DSPs to construct a stack of adders. The inputs of the first-layer adder consist of two data, while the inputs of the remaining adders come from the calculation results of the previous-layer adder and a new data. After the accumulation is completed, the result is output through a register to ensure stability, thereby saving the on-chip buffer space required for the accumulation process.

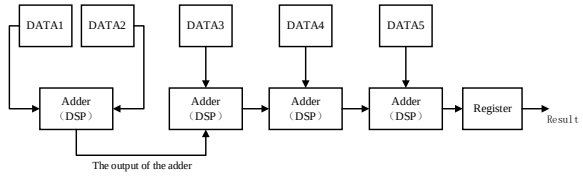

Figure. 2 Accumulator

In order to efficiently perform convolution operations, the feature maps and weight parameters need to be input into parallel multipliers. Since the data is output from BRAM in a serial manner, a separate parallel data generation module is designed in this paper for data parallelization. This module converts the data of the feature maps output from BRAM and the data of the convolutional neural network model parameters into parallel data by using shift registers. Specifically, when the data output from BRAM is valid, the control logic outputs a data valid signal. Upon receiving the data valid signal, this module stores the data and performs shifting operations. When this module has stored all the data to be calculated for a convolution kernel, it puts the parallel data into the corresponding FIFO.

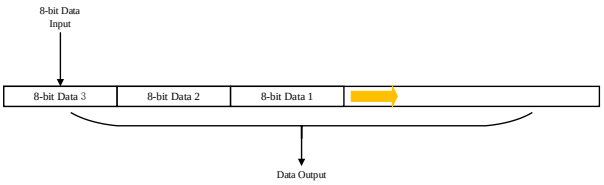

Figure. 3 Parallel data module

3.2 Activation Function Calculation Module

This article uses Hard-Sigmoid and Hard-Swish as activation functions. These activation functions are different piecewise linear functions. Compared with traditional activation functions, piecewise linear activation functions can not only retain the information of the original signal but also introduce non-linear characteristics, enabling the network to better adapt to various complex input data. Moreover, they do not involve complex exponential calculations that are unfriendly to FPGA. Therefore, the structure of the activation function calculation module in this article can be designed to be relatively lightweight.

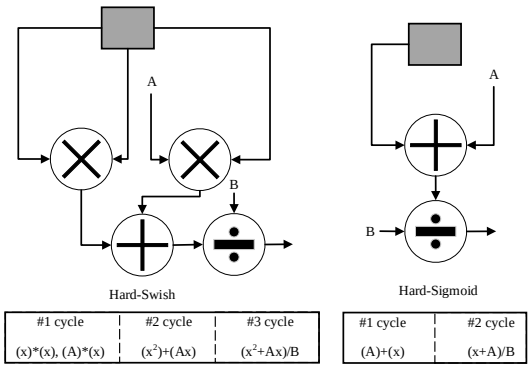

Figure. 4 Activation function calculation

International Journal of Computer Applications Technology and Research
Volume 12–Issue 04, 12-15, 2023, ISSN:-2319–8656
DOI:10.7753/IJCATR1204.1004

3.3 Instruction

The instruction set designed in this article adopts a 32-bit width, which is the same as the register width of the ARM Cortex-A9 processor. Using one register can store all instruction information, thus reducing the hardware resource consumption on the programmable logic side. This instruction set covers the basic information required to perform one convolutional neural network calculation, including the calculation operation type, input feature map size, input channel number, output channel number, convolution kernel size, padding size, stride size, and so on. Among them, the bit width used for the calculation operation type is 4 bits.

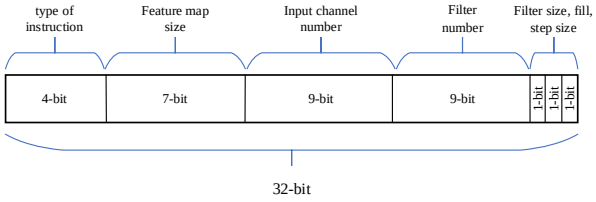

Figure. 5 Instruction composition

3.4 Memory on Chips

There is programmable BRAM memory in FPGA, which is a dual-port memory based on on-chip resources of FPGA. It can achieve high-bandwidth and low-latency storage access.

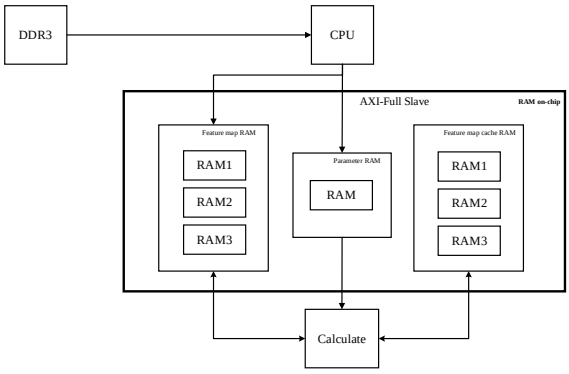

Figure. 5 RAM on-chips

The data structure of the on-chip RAM storage space is shown in the following table.

Table 1. The data structure for storing data in RAM

RAM	Width(bit)	Depth	Capacity(Mb)
FM-RAM1	8	102400	0.78125
FM-RAM2	8	102400	0.78125
FM-RAM3	8	102400	0.78125
P-RAM	8	20480	0.15625
FMC-RAM1	8	53248	0.40625
FMC-RAM2	8	53248	0.40625
FMC-RAM3	8	53248	0.40625

4. ANALYSIS OF RESULTS

The development environment for the convolutional neural network accelerator designed for motion state detection in this article is Xilinx's Vivado 2018.3. The experimental platform uses the XC7Z020-CLG400-2 chip from the ZYNQ-7000 series.

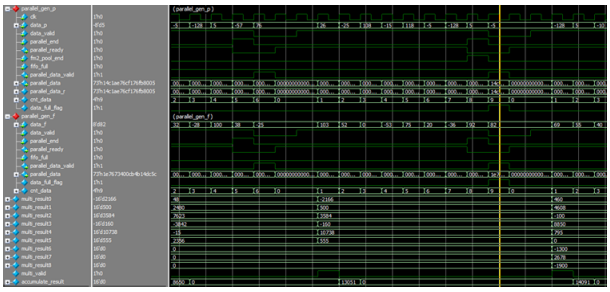

Figure. 6 Simulation waveform of the convolution calculation

In order to evaluate the error caused by quantized 8-bit parameters in this article, the following figure shows the error in each interval when performing operations on quantized 8-bit data for Hard-Swish and Hard-Sigmoid.

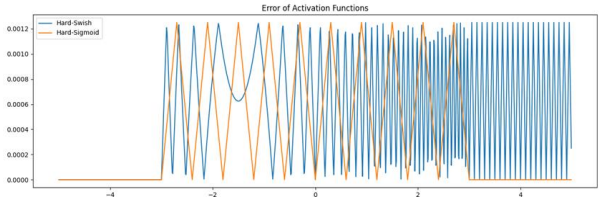

Figure. 7 Error of activation function

The Hard-Swish and Hard-Sigmoid activation functions have fixed value ranges, resulting in relatively small errors outside the range of [-3, 3]. Based on statistics, the average error caused by the Hard-Sigmoid within the range of [-3, 3] is 6.197×10^{-4}, while the average error caused by the Hard-Swish is 8.135×10^{-4}.

The improved LeNet-5 was tested on the MNIST dataset, and the inference results were validated using both hardware computation and simulation.

Table 2. The data structure for storing data in RAM

	LeNet-5
Original model	96.5%
Accelerator	96.1%

To ensure the operational efficiency of the convolutional neural network accelerator and improve the real-time performance of motion detection, it is necessary to increase the operating frequency of the accelerator within a reasonable range. In this article, the entire convolutional neural network accelerator was tested using clock frequencies of 150MHz, 180MHz, and 200MHz, respectively.

Table 3. The performance of accelerator

Clock frequency(MHz)	Performance	Power（W）
150	10.60GOPS	3.293
180	12.67GOPS	3.354
200	14.12GOPS	3.401

5. CONCLUSION

This paper investigates the acceleration of convolutional neural networks based on FPGAs, and proposes a motion state detection platform with high flexibility and lower power consumption, and runs on top of a chip of limited size, highly utilizing the on-chip BRAM and DSP resources, making it advantageous for embedded devices.

6. REFERENCES

[1] Krogh A. What are artificial neural networks?[J]. Nature biotechnology, 2008, 26(2): 195-197.

[2] Collobert R, Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning[C]//Proceedings of the 25th international conference on Machine learning. 2008: 160-167.

[3] Erhan D, Szegedy C, Toshev A, et al. Scalable object detection using deep neural networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 2147-2154.

[4] Li D, Chen X, Becchi M, et al. Evaluating the energy efficiency of deep convolutional neural networks on CPUs and GPUs[C]//2016 IEEE international conferences on big data and cloud computing (BDCloud), social computing and networking (SocialCom), sustainable computing and communications (SustainCom)(BDCloud-SocialCom-SustainCom). IEEE, 2016: 477-484.

[5] Qiu J, Wang J, Yao S, et al. Going deeper with embedded fpga platform for convolutional neural network[C]//Proceedings of the 2016 ACM/SIGDA international symposium on field-programmable gate arrays. 2016: 26-35.

[6] Choquette J, Giroux O, Foley D. Volta: Performance and programmability[J]. Ieee Micro, 2018, 38(2): 42-52.

[7] DiCecco R, Lacey G, Vasiljevic J, et al. Caffeinated FPGAs: FPGA framework for convolutional neural networks[C]//2016 International Conference on Field-Programmable Technology (FPT). IEEE, 2016: 265-268.

[8] Hu J, Shen L, Sun G. Squeeze-and-excitation networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7132-7141.

[9] Han S, Mao H, Dally W J. Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv 2015[J]. arXiv preprint arXiv:1510.00149, 2015, 305.

[10] Zhang C, Li P, Sun G, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks[C]//Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays. 2015: 161-170.

[11] Ma Y, Suda N, Cao Y, et al. Scalable and modularized RTL compilation of convolutional neural networks onto FPGA[C]//2016 26th international conference on field programmable logic and applications (FPL). IEEE, 2016: 1-8.