2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW) | 979-8-3503-0208-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCGridW59191.2023.00065

A flexible dataflow CNN accelerator on FPGA

Haoran Li, Lei Gong, Chao Wang, Xuehai Zhou
School of Computer Science and Technology
University of Science and Technology of China, Hefei, China
lhr18@mail.ustc.edu.cn, {leigong0203, cswang, xhzhou}@ustc.edu.cn

***Abstract*—With the increasing diversity of neural networks, accelerators supporting multiple dataflows are more efficient than those only supporting a specific dataflow. However, due to different storage structures and transmission patterns in different dataflows, deploying an accelerator with multiple heterogeneous cores on FPGA that lacks resources is challenging. In this paper, we present an instruction-based CNN accelerator supporting flexible dataflow. Towards reducing the resource occupation of FPGA, we propose a novel architecture with a delicate data path reusing the storage module and the computing module. In addition, a special instruction mechanism is designed to control the data transmission. As a case study, we implement a prototype and test it on ResNet18 and MobileNetv1. Results show, compared to the multi-core accelerator that supports the flexible dataflow, we achieve 1.3-2.7x speedup.**

***Index Terms*—FPGA, CNN, Hardware accelerator**

I. Introduction

Previous FPGA-based CNN accelerators used fixed dataflows for customization of simple models [1]–[6]. However, they do not work well in models with multiple computing tasks because the parallel factors and parallel types in fixed dataflows do not match well with multiple computing tasks at the same time. For computing tasks with insufficient parallelism in certain dimensions, the utilization of computing resources will decrease significantly. Operators with different parallel types, such as depthwise convolution, will result in inefficient task mapping. Therefore, designing an accelerator that supports flexible dataflow can perform these diverse tasks more efficiently.

To solve this problem, the previous work adopted the multi-core structure, which deploys multiple computing cores supporting different dataflows for different computing tasks [7] [8]. However, there are two problems with multi-core architecture solutions. Firstly, multiple heterogeneous computing cores cause significant resource consumption. Secondly, complex task scheduling is required between multiple cores to avoid idle.

In this article, we propose a new accelerator design, which maps multiple dataflows to a single computing core, avoiding the excessive consumption of resources and scheduling difficulties of multi-core structures. There are two main challenges in designing such an accelerator. The first is constructing a data path to enable multiple data transmission patterns. The second is designing the instruction mechanism to achieve accurate and flexible fine-grained control.

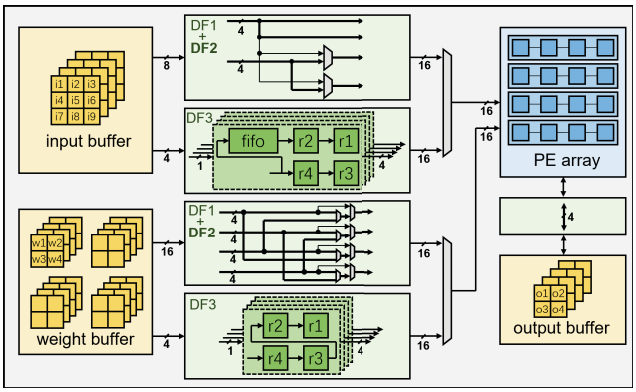

Fig. 1. Block diagram of the accelerator with three dataflows (DF1-DF3).

II. Accelerator Architecture

A. Design Overview

The overall architecture of our accelerator is shown in Figure 1, which consists of the computational module, the storage module, and the data path three parts. The computational module consists of a 2-D processing element(PE) array, and each row uses an adder tree structure to form a vector inner product unit. The storage module is divided into the input buffer, weight buffer, and output buffer. The data path connects the storage module and the computational module and supports multiple data transmission patterns of different dataflows.

In fixed dataflow accelerators, efficient data reuse was achieved by constructing specialized broadcast paths. This results in data dependencies between different PEs, limiting the flexibility of task mapping and data transmission. To solve this problem, we provide multiple data paths for different dataflows. Taking the input buffer as an example, you can choose to broadcast the same data to all vector inner product units, or you can broadcast different data to different groups, which ensures flexible task mapping for different dataflows.

Depthwise convolution usually maps computations within the same kernel to the same vector inner product unit to improve resource utilization. To save resources, we reuse storage structures of other dataflows. However, it can only provide parallel data transmission of channel dimension, and cannot provide parallel data transmission of feature map dimension required by depthwise convolution. To solve this problem, we add the line buffer to construct a new data path,

979-8-3503-0208-0/23/$31.00 ©2023 IEEE
DOI 10.1109/CCGridW59191.2023.00065

which improves the parallel data transmission capability by increasing data reuse.

Figure 1 shows an example of an accelerator that supports three dataflows DF1, DF2, and DF3. DF1 supports the parallelism of input and output channel dimensions. DF2 additionally supports the parallelism of feature map dimensions to perform tasks with insufficient output channels. DF3 supports the parallelism of kernel dimensions for depthwise convolution. For DF2, the corresponding packet broadcast structure is constructed for the input data and weight data. For DF3, the line buffer is added to the data path for input data and weight data reducing the repeated transmission of the same data by caching it in the line buffer. The data path of the output buffer can be reused directly because the transmission pattern of the three dataflows is the same.

B. Instruction

To control the complex data path, we design a fine-grained instruction, which can not only accurately control the data transmission behavior in the micro-architecture, but also support efficient mapping of various tasks. The Instruction consists of four control signals, including input address, weight address, output address, and data path selection. Thanks to the delicate storage structure, the above information is enough to accurately control the data transmission per cycle. It should be noted that the meaning of the control signal changes with the hardware customization, but the instruction format remains unchanged.

By combining fine-grained instructions, we can easily construct instruction sequences for different tasks. As shown in Figure 2, two instruction sequences perform conventional convolution and depthwise convolution respectively. In the instruction sequence of depthwise convolution, the filling of input data and the reuse of weight data in the line buffer are completed by setting the addresses of three buffers in an orderly manner.

CONV:
1: DF2, i1, w1, o1
2: DF2, i2, w2, o1
3: DF2, i4, w3, o1
4: DF2, i5, w4, o1
5: DF2, i2, w1, o2
6: DF2, i3, w2, o2
7: DF2, i5, w3, o2
8: DF2, i6, w4, o2

DW-CONV:
1: DF3, i1, w1, ---
2: DF3, i2, w2, ---
3: DF3, i3, w3, ---
4: DF3, i4, w4, --- //**filling line buffers**
5: DF3, i5, ---, o1
6: DF3, i6, ---, o2
7: DF3, i7, ---, --- //**filling line buffers**
8: DF3, i8, ---, o3
9: DF3, i9, ---, o4

Fig. 2. Conventional convolution and depthwise convolution code examples.

III. Evaluation

To evaluate our design, we implement a prototype and tested it on ResNet18 and MobileNetv1, using a cycle accuracy simulator to calculate the computing resource utilization at runtime. For comparison, we implemented two fixed dataflow accelerators and one multi-core accelerator.

When testing on ResNet18, the two fixed dataflow accelerators support DF1 (X: Tn=4, Y: Tm=512) and DF2 (X: Tn=4, Y: Tm=64, Tr=8) respectively, while our design supports both of them. The multi-core accelerator is constructed with the same number of PE and supports DF3 (X: Tn=4, Y: Tm=256) and DF4 (X: Tn=4, Y: Tm=64, Tr= 4). Figure 3 shows the resource utilization of each layer, where similar layers are hidden. Our design can achieve high utilization in different layers, due to the advantages of two dataflows. The utilization of the multi-core accelerator is insufficient because the parallelism between cores cannot be fully utilized.

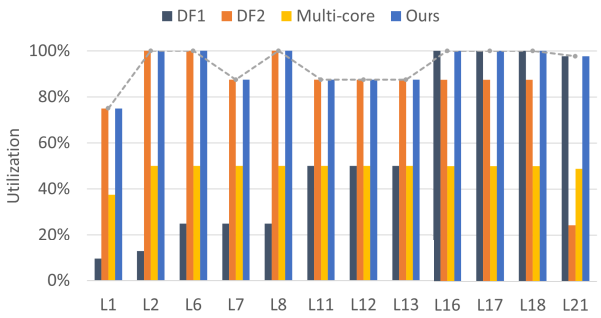

Fig. 3. Comparison with fixed dataflow and multi-core accelerators.

TABLE I
AVERAGE UTILIZATION ON RESNET18 AND MOBILENETV1

	Multi-core		**Ours**
	batch=1	batch=10	
ResNet18	48.9%	72.6%	93.6%
MobileNetv1	27.6%	54.1%	73.9%

To further compare our design with the multi-core accelerator, we continue to test the average resource utilization on ResNet18 and MobileNetv1. Both accelerators add a new dataflow to support depthwise convolution. As shown in Table I, when the batch is large enough, the multi-core accelerator can fully optimize task scheduling, and the resource utilization is significantly improved. But it is still lower than our single-core design because it is difficult to achieve perfect inter-core parallelism in real tasks.

Acknowledgment

This work was supported in part by the National Key R&D Program of China under Grants 2017YFA0700900 and 2017YFA0700903, in part by the National Natural Science Foundation of China under Grants 62102383, 61976200, and 62172380, in part by Jiangsu Provincial Natural Science Foundation under Grant BK20210123, in part by Youth Innovation Promotion Association CAS under Grant Y2021121, and in part by the USTC Research Funds of the Double First-Class Initiative under Grant YD2150002005.

References

[1] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie and X. Zhou, "DLAU: A Scalable Deep Learning Accelerator Unit on FPGA," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 3, pp. 513-517, March 2017.

[2] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. 2015. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. In Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '15). Association for Computing Machinery, New York, NY, USA, 161–170.
[3] L. Gong, C. Wang, X. Li, H. Chen and X. Zhou, "MALOC: A Fully Pipelined FPGA Accelerator for Convolutional Neural Networks With All Layers Mapped on Chip," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 11, pp. 2601-2612, Nov. 2018.
[4] W. Lou, C. Wang, L. Gong and X. Zhou, "OctCNN: An Energy-Efficient FPGA Accelerator for CNNs using Octave Convolution Algorithm," 2020 IEEE International Conference on Cluster Computing (CLUSTER), Kobe, Japan, 2020, pp. 410-411.
[5] L. Gong, C. Wang, X. Li and X. Zhou, "Improving HW/SW Adaptability for Accelerating CNNs on FPGAs Through A Dynamic/Static Co-Reconfiguration Approach," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 7, pp. 1854-1865, 1 July 2021.
[6] T. Wang et al., "ViA: A Novel Vision-Transformer Accelerator Based on FPGA," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 11, pp. 4088-4099, Nov. 2022.
[7] H. Kwon, L. Lai, M. Pellauer, T. Krishna, Y. -H. Chen and V. Chandra, "Heterogeneous Dataflow Accelerators for Multi-DNN Workloads," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Korea (South), 2021, pp. 71-83.
[8] E. Baek, D. Kwon and J. Kim, "A Multi-Neural Network Acceleration Architecture," 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), Valencia, Spain, 2020, pp. 940-953.