Efficient Convolutional Neural Network Accelerator Based on Systolic Array

Yeong-Kang Lai and Yu-Jen Tsai
Department of Electrical Engineering
National Chung Hsing University
Taichung, Taiwan
7107064076@smail.nchu.edu.tw, yklai@dragon.nchu.edu.tw

***Abstract*—This paper uses 72 PE as the basis for convolution operations, which can handle 3 x 3 and 1 x 1 filter sizes. Moreover, using the Systolic Array design architecture, the data reuse of this architecture is better than general PE architecture. Systolic Array architecture only needs to access once. This paper integrates Convolution and Max Pooling. This hardware verifies on Xilinx ZCU102 FPGA board. The hardware uses quantized weight parameters, and the hardware arithmetic precision is UINT8. The operation frequency sets at 100 MHz, throughput can reach 14.4 GOPs. The efficiency is 98.90%, the bandwidth is 150.82 MB, and Convolution integrates Max-Pooling to save 31.75% of DRAM access. In the future, the Operation Frequency can increase to more than 200 MHZ. The increase in the number of PEs can enhance the efficiency of parallel operations, which can effectively improve the throughput of the hardware.**

***Keywords*—CNN Accelerator, Systolic array**

I. INTRODUCTION

CNN (Convolution Neural Network) is composed of a large number of parallel operations. Because of the particularity of its operations, the hardware design also optimizes for high parallel operations. The overall architecture goal is to provide high throughput and high parallelism, but the memory locates on the hardware. The power consumption of data access and transmission is higher than that of the data operation itself. Therefore, if we want to reduce the power consumption of data access, we need to think about how to reuse data efficiently. It requires optimizing the design of dataflow. In line with the adjustment of input data and filters to different dimensions between different layers of CNN, the above is the key to consider in hardware architecture design. The previous literature on CNN hardware accelerators like [1]-[4] all proposed their accelerators. Some optimize for throughput, some aim at low power consumption, and there are many hardware accelerators according to different needs. This paper will design a set of hardware accelerators, a systolic array accelerator based on Weight stationery, and a scalable modular design.

II. PROPOSED TRANSFORM ARCHITECTURE

A. System Architecture

Most of the deep neural network operations are convolution, multiplication, and addition. Although different neural networks have different layer designs, they mainly base on convolutional layers. Using PE's high parallel computing can speed up the calculation speed of the convolutional network.

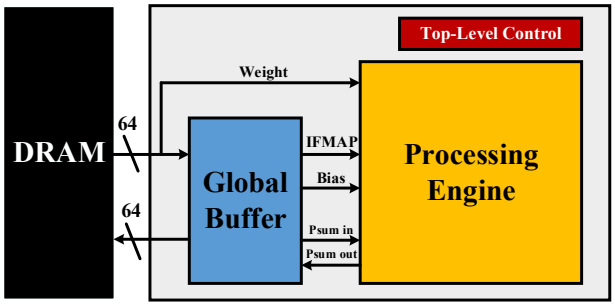

Fig. 1 System Architecture

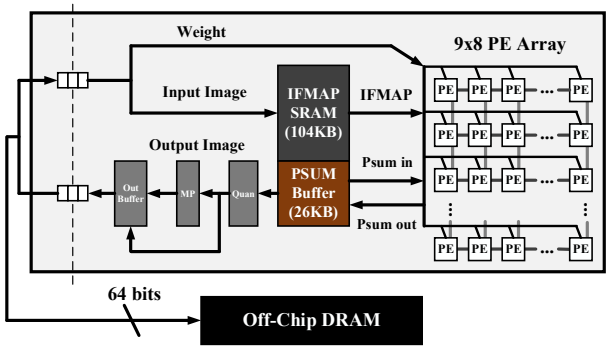

Fig. 2 Accelerator Architecture

As shown in Fig. 1, the input images required for neural network operations also have the weight coefficients of each layer. They are first stored in DRAM and transferred from AXI to Global Buffer for storage. The width of AXI from Dram to Global Buffer is 64bit, and the data from Global Buffer pass in. Do calculations in Processing Engine. After the operation of each module, the result obtained is sent back to DRAM for storage, waiting for the call of the next operation. After the required number of operations is complete, the final result output can obtain.

B. Accelerator Design

Fig. 2 is the architecture designed for this paper. Input Feature Map and Weight are stored in DRAM and stored in their respective Buffers via FIFO. IFMAP will save four rows of data in the IFMAP Buffer. Then save the Weight in the correct Weight Buffer according to the position of each Pixel. When ready, IFMAP enters the PE Group in sequence and performs the Convolution operation on the Weight. The result of the operation stores in PSUM for accumulation. After getting the OFMAP, you can enter Max Pooling to find the

2022 IEEE International Conference on Consumer Electronics (ICCE) | 978-1-6654-4154-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCE53296.2022.9730180

maximum characteristic value, and finally enter the Output

Table I Comparison Table

	EyerissV2[1]	EIE[2]	Ardakani'[3]	Sim'[4]	NullHop[5]	Proposed
Technology	65nm	45nm	65nm	65nm	28nm	-
FPGA	-	-	-	Zynq 7000	Zynq 7100	ZCU102
Frequency	200MHz	800MHz	200MHz	60Mhz	60MHz	100MHz
CNN model	AlexNet	AlexNet	VGG-16	AlexNet	VGG-19	AgileV3
Precision	8-bit FXP	4-bit FXP	16-bit FXP	16-bit FXP	16-bit FXP	UINT8
Throughput	153.6GOPS	102GOPS	76GOPS	61.44GOPS	15GOPS	14.4GOPS

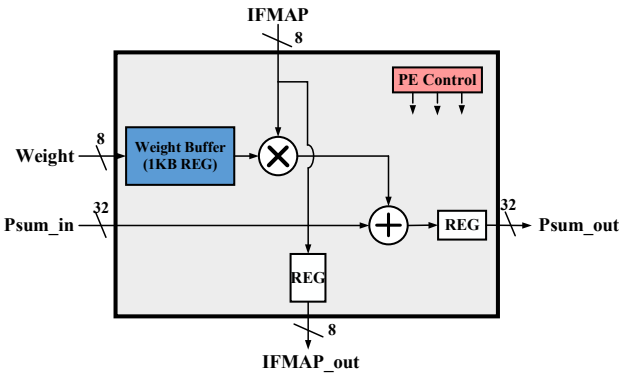

Fig. 3 Processing Element Architecture

Buffer and wait to be transferred back to DRAM for storage.

C. *Processing Element Design*

As shown in Fig. 3, this architecture design uses Weight Stationary Dataflow, so each PE has a Weight Buffer designed to provide Weight storage. One word of the Weight Buffer is 8-bit, and the total size is 1 KB Register Buffer. The PE design is mainly multiplication and addition operations. After IFMAP enters the PE, it divides into two lines, one saves directly in the register, and the next clock output prepare. The other multiplies by the Weight stored in the Weight Buffer in advance. Then add to the Psum calculation by the previous PE. It saves in the register to wait for the output of the next clock. The architecture of the Systolic Array uses in many papers. After analysis, this paper also adopts the architecture of Systolic Array and the dataflow calculation method of Weight stationary. Because Weight Stationary use, Weight will fix in each PE, so the flowing data is Input Feature Map and PSUM. After the IFMAP enters the PE, it will wait for a clock, and the second clock will transfer the IFMAP stored in the register to the next PE to prepare for the operation. IFMAP adopts the skew arrangement input method and divides it into three levels. The calculation that required nine clocks to complete can be compressed to three clocks. PSUM is the result of multiplying IFMAP and Weight, which is then added to the previous PSUM, and transferred to the right to the next PE for accumulation.

III. Energy-Efficient Features

In the hardware design, the memory setting is related to the overall data reuse. If SRAM or Global Buffer is not set, it means that all data needs to be accessed from DRAM. However, the energy consumption of accessing DRAM is much higher than that of the internal SRAM or Global Buffer of the hardware. Reading or storing data from the outside also causes the data access time to be out of alignment due to different pulses, causing problems such as hardware idle. Therefore, the design of SRAM or Global Buffer is necessary, which can save much hardware energy consumption and ensure the correct data access during hardware operations. The power-saving of this architecture design is mainly the access of IFMAP. If the hardware design does not use Systolic Array, the number of repeated accesses of IFMAP will be eight times that of the current, which will generate a lot of power consumption. Systolic Array is an ordinary design, which means that the hardware has better scalability, and the appropriate expansion of the PE structure can save more access and energy consumption.

IV. Experiental Results & Conclusions

Table I is the performance comparison table of this paper and other journal papers. The hardware design specification of this paper uses the Xilinx Zynq series ZCU102 board, the neural network uses Agilev3, the clock rate sets at 100 MHz, and the Throughput can reach 14.4 GOPs. The precision used in this paper is UINT8, the other papers use 16bit, 10bit, 8bit, or 4bit fixed-point, and the number of PEs is 72. This paper uses the Systolic Array design architecture. Compared with general PE operations, this architecture is better in data reuse. It can reduce the energy consumption of SRAM access. This paper uses the dataflow method of Weight Stationary and stores the weights on the PE can effectively reduce the energy consumption of the weights. It can reduce the number of DRAM accesses effectively. The hardware design bases on 72 PE for convolution processing.

References

[1] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," IEEE J. Emerging Sel. Topics Circuits Syst., vol. 9, no. 2, pp. 292–308, Jun. 2019.

[2] S. Han et al., "EIE: Efficient Inference Engine on Compressed Deep Neural Network," 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), 2016, pp. 243-254, doi: 10.1109/ISCA.2016.30.

[3] A. Ardakani, C. Condo, M. Ahmadi and W. J. Gross, "An Architecture to Accelerate Convolution in Deep Neural Networks," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 4, pp. 1349-1362, April 2018, doi: 10.1109/TCSI.2017.2757036.

[4] J. Sim, S. Lee and L. Kim, "An Energy-Efficient Deep Convolutional Neural Network Inference Processor With Enhanced Output Stationary Dataflow in 65-nm CMOS," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 1, pp. 87-100, Jan. 2020, doi: 10.1109/TVLSI.2019.2935251.