Area-Efficient Reconfigurable Pooling Hardware

Jiho Park, Jeonghun Son, Kyoduk Ku, and Hoyoung Yoo
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Republic of Korea
jhpark.cas@gmail.com, jhsohn.cas@gmail.com, gdku.cas@gmail.com, hyyoo@cnu.ac.kr

Abstract— **Pooling operations enhance neural network efficiency by reducing data size, computational complexity, and overfitting, and recent research introduce diverse pooling methods to improve network performance. However, this diversification complicates hardware implementation of pooling operations. In this paper, a reconfigurable pooling unit architecture that can efficiently implement various pooling schemes on a single hardware is proposed. Reconfigurable pooling hardware supports max pooling, average pooling, and absolute average deviation pooling, sharing hardware resources such as addition, subtraction and comparison operations to minimize hardware complexity. Furthermore, a folding-based tree structure is used to effectively handle large input feature maps through repeated computation cycles. The proposed pooling hardware is implemented using Synopsys design compiler on 28 nm CMOS technology, reducing hardware complexity by 45% with only 8% increase in critical path delay compared to straight-forward pooling hardware of the same input size.**

Keywords; Pooling architecture, Max pooling, Average pooling, Absolute average deviation pooling, Resource sharing, Folding

I. INTRODUCTION

Pooling operations effectively reduce the size of data in AI networks, which reduces the computational complexity of neural networks and prevents them from overfitting, thereby improving the efficiency of feature extraction [1]. Previous neural networks mainly used a single method pooling operation, but as the network develops, various methods and kernel-sized pooling operations are introduced to improve the performance of the network or the accuracy of the network by using multiple pooling operations in one neural network [2]. In addition, changing the pooling method in the same network changes the performance [3]. This diversification improves difficulty of implementing pooling hardware.

In this paper, a unified pooling architecture is proposed that supports various methods and input size pooling operations. It supports max pooling [4], average pooling [5], and absolute average deviation(AAD) pooling [3] and minimizes the area by sharing the same resources used in each pooling. In addition, by using folding techniques, it is possible to compute input feature maps that are larger than hardware inputs. The proposed architecture can be flexibly applied to various network models and various pooling methods can be performed while maintaining the accuracy of the pooling operation with only a small hardware complexity.

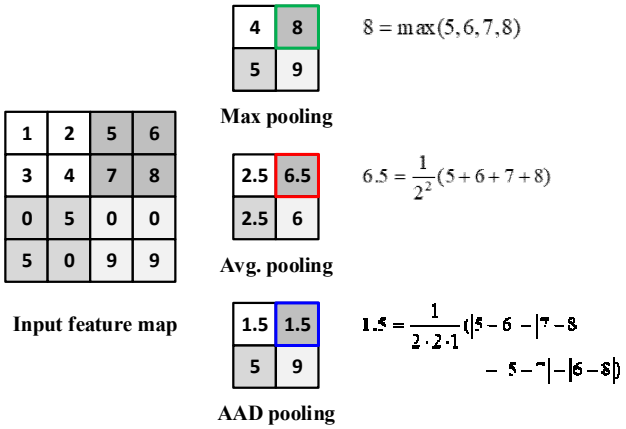

Figure 1. Pooling example of max pooling, average pooling and absolute average deviation pooling when K=2, S=2.

II. HARDWARE IMPLEMENTATION

Proposed pooling hardware can be operated through a single operation module by integrating pooling operations. Each pooling operation is shown in Fig. 1. Max Pooling selects the maximum value within a pooling window from the input feature map and can be expressed as follows

$$Y_{i,j} = \max_{0 \le m,n \le K} \left\{ X_{i \cdot s+m, j \cdot s+n} \right\}. \tag{1}$$

Average pooling calculates the average of all values within the same pooling window and can be expressed as follows

$$Y_{i,j} = \frac{1}{K^2} \sum_{m=0}^{K-1} \sum_{n=0}^{K-1} X_{i \cdot s+m, j \cdot s+n} \ . \tag{2}$$

AAD pooling calculates the average of the absolute differences between adjacent elements in both row and column directions within the pooling window, expressed as follows

$$Y_{i,j} = \frac{1}{2K(K-1)} \left[\sum_{m=0}^{K-1} \sum_{n=0}^{K-2} \left| X_{i \cdot s+m, j \cdot s+n} - X_{i \cdot s+m, j \cdot s+n+1} \right| + \sum_{m=0}^{K-1} \sum_{n=0}^{K-2} \left| X_{i \cdot s+m, j \cdot s+n} - X_{i \cdot s+m+1, j \cdot s+n} \right| \right], \tag{3}$$

where X and Y are input and output feature map, i and j are the positions in the output feature map, K is the window size, and S is the stride.

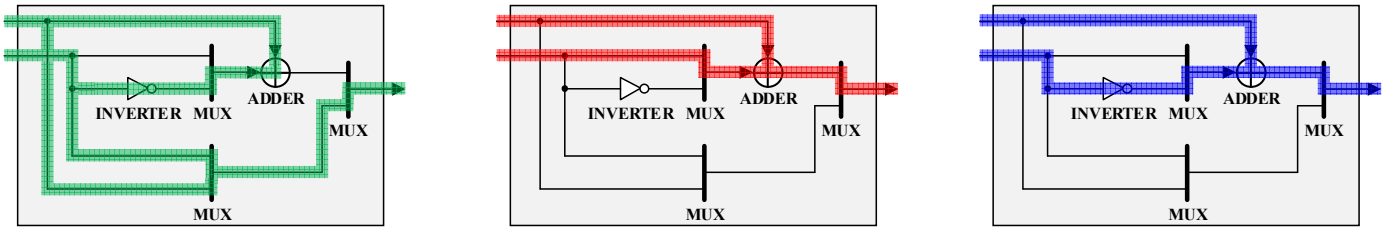

Figure 2. Pooling unit and each operation.

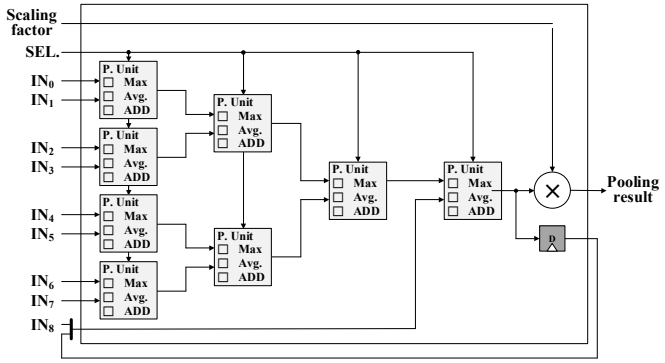

Figure 3. The Overall proposed pooling architecture.

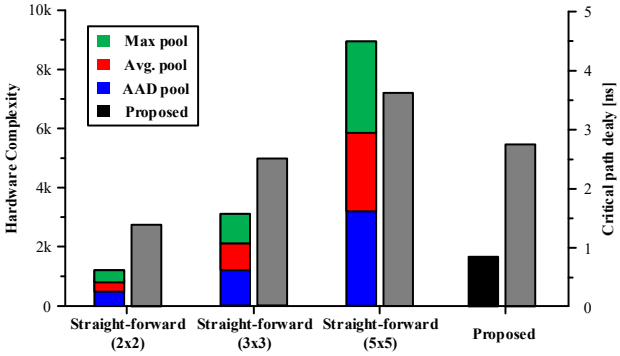

Figure 4. Hardware performance comparison

Each pooling method requires a unique arithmetic operation per pooling method. Max pooling performs a comparison operation to select the maximum value, average pooling performs an addition operation to sum the inputs to calculate the average, and AAD pooling performs a subtraction operation to find the absolute value of the difference between the inputs. Subtractors and comparators are organized based on addition operators, so these operations overlap hardware resources. In this paper, a reconfigurable pooling hardware architecture is proposed that utilizes these overlapping hardware resources by sharing them.

The pooling unit consists of one adder, one inverter, and three MUXs. The different outputs depending on the pooling method are shown in Fig. 2. In the case of Max pooling, the larger value is selected and output by subtraction operation, average pooling sums the inputs to get the average value, and AAD pooling outputs the result of subtraction operation to get the absolute value of the difference between two inputs. The hardware structure for reconfigurable pooling is shown in Fig. 3. The circuit consists of a folding-based tree structure composed of pooling units, enabling efficient pooling operations through repeated computation cycles for large input feature maps and the pooling result is exported by multiplying the scaling factor required for each pooling operation. For larger inputs, intermediate computation results are stored in D flip-flops and iteratively computed until final results are produced, providing flexibility for various feature map sizes.

III. Experimnetal Result

Various pooling hardware architectures are implemented using Synopsys Design Compiler S-2021.06 in 28nm CMOS technology, and hardware performance is evaluated. The proposed method is compared with the straight-forward architecture in terms of hardware complexity and critical path delay, as shown in Fig. 4. The results show that the proposed method reduce hardware complexity by 45% with only 8% increase in critical path delay compared to straight-forward 3x3 input size pooling hardware. The proposed reconfigurable pooling hardware supports three pooling operations with only lower hardware complexity than straight-forward pooling hardware. The results show that the reconfigurable pooling hardware is suitable for resource-constrained deep learning accelerator applications.

Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2022R1A5A8026986) and the EDA tool was supported by the IC Design Education Center (IDEC), Korea.

References

[1] Dhilleswararao, P., Boppu, S., Manikandan, M. S., & Cenkeramaddi, L. R. (2022). Efficient hardware architectures for accelerating deep neural networks: Survey. IEEE access, 10, 131788-131828.

[2] Khalil, K., Eldash, O., Kumar, A., & Bayoumi, M. (2022). Designing novel AAD pooling in hardware for a convolutional neural network accelerator. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 30(3), 303-314.

[3] Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., & Almotairi, S. (2022). A comparison of pooling methods for convolutional neural networks. Applied Sciences, 12(17), 8643.

[4] Zhao, B., Chong, Y. S., & Do, A. T. (2020, October). Area and energy efficient 2D max-pooling for convolutional neural network hardware accelerator. In IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society (pp. 423-427). IEEE.

[5] Wang, S. H., Phillips, P., Sui, Y., Liu, B., Yang, M., & Cheng, H. (2018). Classification of Alzheimer's disease based on eight-layer convolutional neural network with leaky rectified linear unit and max pooling. Journal of medical systems, 42, 1-11.