FPGA Implementation of Cycle-Reduced Diagonal Data Flow Systolic Array for Edge Device AI

Gyubin Seong, Jong Kang Park and Jong Tae Kim
Department of Electrical and Computer Engineering
Sungkyunkwan University
Suwon, Republic of Korea
{top9641, jkpark1, jtkim}@skku.edu

2023 20th International SoC Design Conference (ISOCC) | 979-8-3503-2703-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISOCC59558.2023.10396567

Abstract— **As deep-learning has been widely adopted, utilization of resource-constrained edge devices becomes important to relax communication costs and user data privacy between server and edge device. The systolic array is generally used in deep-learning accelerators. Recent studies of systolic arrays are trying to reduce the total computation times of deep-learning applications inference. However, computation cycles for one matrix multiplication are not reduced. This paper proposes the systolic array for edge device with reduced computation cycles. The proposed systolic array reduces computation cycles by changing data transmitting direction between processing elements (PEs). We implement our design on FPGA in 200MHz frequency. The simulated cycle reduction in the 8×8 PE array is 31.82% without additional resources.**

Keywords; systolic array; on-device AI; edge device; FPGA;

I. INTRODUCTION

As artificial intelligence (AI) applications using deep-learning models are widespread in the real world, the use of AI applications on edge device is also getting attention. Typically, AI inference of edge device is computed in a cloud server. However, AI inference through a cloud server causes significant communication latency and user data security problems. Therefore, On-device AI [1] has been conducted, which computes AI inference on edge devices. However, for AI application services, high throughput and real-time response with constrained resources must be achieved [2].

The systolic array is a widely used architecture to implement efficient deep-learning accelerators. Recent studies on systolic arrays exhibited flexible architectures that support variable input dimension and dynamic dataflow. The Configurable Multi-directional Systolic Array (CMSA) [3] shows a flexible systolic array to effectively support small-scale convolution. It contains a split processing element (PE) array to provide two simultaneous data paths for small-scale matrix multiplication. This increases PE utilization, and the total computation cycles for small-scale matrix multiplication can be reduced. In [4], a dynamic dataflow was realized to choose one of the data stationary techniques appropriate to input dimensions. According to the input matrix dimension, it determines the data stationary dataflow that requires the least computation time. However, the above studies did not consider a reduction in computation cycles of original matrix multiplication.

This paper proposes a new systolic array architecture requiring fewer computation cycles than the conventional systolic array. Generally, the systolic array in a deep-learning accelerator is an $N\times N$ square array, and each PE is connected vertically and horizontally. The proposed architecture replaces the vertical path with a diagonal path to reduce the required cycles to enter data into an array. Consequently, computation cycles are reduced regardless of input dimensions and data stationary dataflow. Cycle reduction in the 8×8 processing element array is 31.82%.

II. ARCHITECTURE DETAILS

A. Conventional Systolic Array

As shown in Fig. 1, a general systolic array in deep-learning inference is arranged in a 2-D square. The rows of the input matrix are delayed to be multiplied sequentially with another path's input. In $N\times N$ size matrix multiplication, one element is multiplied N times. It means that the element must pass N PEs and take N cycles during the entire matrix multiplication. Due to the input matrix delay, the input matrix is stretched from N to $2N–1$, and the stretched input matrix passes N PEs. The total computation cycles of matrix multiplication in $N\times N$ size conventional systolic array are defined as

$$\text{Cycles} = 3N–2 \tag{1}$$

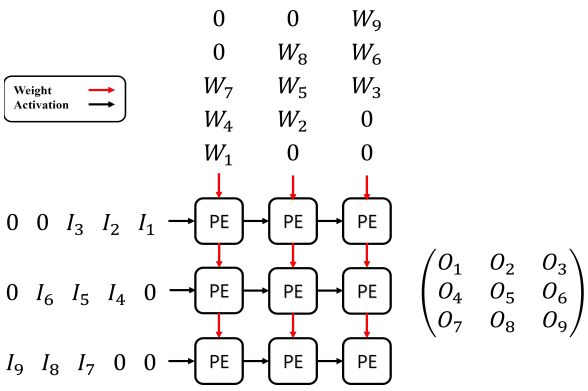

Figure 1. Conventional systolic array example

B. Proposed Systolic Array Using Diagonal Data Flow

Due to the delayed input matrix, the required cycles to enter the input matrix increase, leading to an increase in the total computation cycles. However, if the delay for the input matrix is removed to decrease computation cycles, elements for matrix multiplication are not matched properly in PEs. To assign

This work was supported by the BK21 FOUR Project

elements in PEs correctly, the connections between PEs must be changed. The weight matrix cannot enter vertically if the activation matrix is entered horizontally in the PE array. Because when the weight element entered to destination PE, the activation element that needs to be multiplied with is already left that PE.

As shown in Fig. 2, the weight element must pass to the lower right (i.e., diagonal) PE to chase the activation element that needs to be multiplied. However, except for the main diagonal PEs, the rest of the weight matrix paths have less than N PEs. Furthermore, the left downside of main diagonal PEs is not used to multiply because the weight matrix is not entered. To solve this problem, we connect these unused PEs to uncompleted weight matrix paths to make them the same as the main diagonal PEs path. However, the order of multiplication is changed because of connected unused PEs. Therefore, locations of output matrix partial sums in PEs are different from conventional systolic arrays. As the weight matrix paths become diagonal, the outputs are shifted according to the order of rows. Fig. 3 represents the completed PE array and location of matrix multiplication output. Because each matrix row has no input delay, the input matrix is not stretched like a conventional systolic array. Therefore, the total computation cycles in the $N\times N$ size array are defined as

$$\text{Cycles} = 2N-1 \tag{2}$$

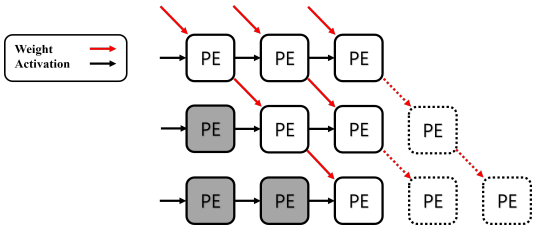

Figure 2. Uncompleted PEs Path

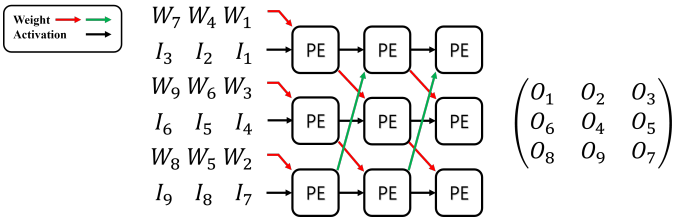

Figure 3. Completed PEs Path

However, one can experience a critical path issue with increased PE array size. As shown in Fig. 3, green weight-paths take long path delays than red weight-paths. In a small-size PE array, the difference in path delay is negligible. However, the difference becomes more significant as the PE array increases because the green path connects the lowest PE to the highest PE. To solve this problem, we removed the green path and added the N–1 weight matrix path to the highest row PEs. Fig. 4 represents the proposed systolic array. Weight matrix entered to additional weight paths with input delay because the path's prior PEs are removed. Because the delayed weight matrix passes a reduced PEs path, the input delay does not affect the total computation cycles. Therefore, computation cycle reduction can be fully achieved in large-size PE arrays without significant path delay differences.

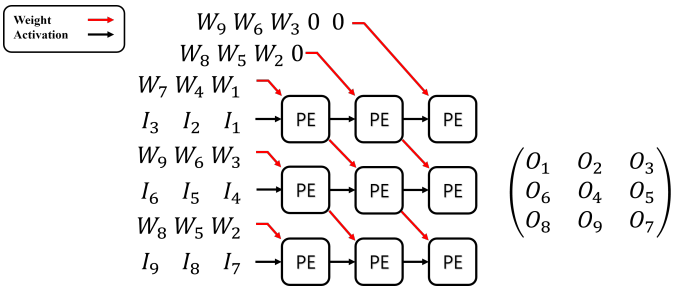

Figure 4. Proposed Systolic Array example

III. EXPERIMENTS

We implemented the register transfer level of the proposed systolic array in Verilog Hardware Description Language and synthesized it on FPGA with Xilinx Vivado. Edge device AI applications generally use lightweight deep-learning models. Because 9×9 to 15×15 arrays can effectively perform lightweight neural network models [3], we validated the proposed architecture in 8×8 size PE array.

Table I shows the implementation results of the conventional and proposed systolic array. Both simulated 8×8 size matrix multiplication in 200MHz frequency. Despite the additional data paths, the proposed systolic array uses flip-flops less than the conventional systolic array. The critical path delay between PEs of the proposed systolic array is longer but negligible. The total computation cycles are reduced by 31.82%. The estimated total computation time with setting the clock period as critical path delay is reduced by 30.22%

TABLE I. EXPERIMENT RESULTS

	Conventional Systolic Array	**Proposed Systolic Array**
Computation cycles	22	15
Critical path delay	3.84ns	3.93ns
Slice Flop-Flops	2270	2130
Slice LUTs	2019	2019

IV. CONCLUSION

Real-time inference on resource constraint edge devices is required for on-device AI. This paper proposed a cycle-reduced systolic array by changing the vertical data path to the diagonal data path. Computation cycle reduction is achieved without additional resources and negligible critical path delay increase.

REFERENCES

[1] MIT Technical Review. (accessed 1 May 2023). On-Device AI. https://www.technologyreview.com/hub/ubiquitous-on-device-ai/.

[2] M. Ham et al., "NNStreamer: Efficient and Agile Development of On-Device AI Systems," 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2021, pp. 198-207.

[3] R. Xu, S. Ma, Y. Wang and Y. Guo, "CMSA: Configurable Multi-directional Systolic Array for Convolutional Neural Networks," 2020 IEEE 38th International Conference on Computer Design (ICCD), 2020, pp. 494-497.

[4] Wang, Bo, et al. "A novel systolic array processor with dynamic dataflows." Integration 85, 2022, 42-47.