A Comparison of Single-Buffer and Double-Buffer Design in a Systolic Array Generator

Dongchu Su¹, Yong Li¹*

¹College of Computer Science, National University of Defense Technology, Changsha 410073, China
*Corresponding author’s e-mail: yongli@nudt.edu.cn

Abstract. In application fields such as face recognition and image recognition using deep learning, more convolution operations are required for the increasing amount of data. Therefore, the use of systolic array acceleration is becoming a key technology trend to accelerate the development of deep learning applications. In previous designs, most of the systolic array used single-buffer or double-buffer structures, but most of them did not compare the difference between the two in detail. This work designs and implements a three-level systolic array generator, which can be configured in single-buffer or double-buffer modes. We also program the generated systolic array accelerator on Nexys-Vedio FPGA to explore the performance and overhead of single-buffer and double-buffer structure modes. The results show that the throughput of the double-buffer structure is increased by nearly 3× compared to the single-buffer structure while only brings an additional 28% power consumption and 25% area overhead under the SIMC 130nm technology. And compared with the previous work, the proposed systolic array generator reduces power consumption by 75% and area overhead by 34% with almost no loss in performance.

1. Introduction

In recent years, with the improvement of chip technology, there have been more and more deep learning applications using systolic array to accelerate face recognition and image recognition, especially the state-of-the-art work Google’s TPU [1], showing the advantages of the systolic array structure. In the training reset-net50 [2] scenario, 1 full TPU V2 is 27 times faster than 8 V100 GPUs, and the power consumption is 38% lower, with 33% lower cost [3], which makes the systolic array more eye-catching. There are sorts of computing structure of systolic array, including OS (Output Station) [4], WS (Weight Station) [5], single-buffer structure, double-buffer structure and other types.

Inspired by rather excellent systolic array work such as TPU, a trend on the implementation and research of systolic array have appeared in recent years. For example, Berkeley's Gemmini [8] implemented a highly configurable systolic array generator using chisel [9] language. User can freely configure Gemmini with WS, OS mode, and different array size, but it can also be found that the poor multiplier in PE (Process Element) causes serious timing problems when programming the FPGA. There is also a design for low power consumption, FlexSA [10] proposes a systolic array architecture that
supports sub-systolic mode, which can operate GEMM in part of systolic array. However, this mode can actually be implemented using multiple tiles design. Based on the above problems, we design and implement a three-level systolic array generator, which can freely configure the size of the systolic array, the number of buffers in the PE, and can perform different GEMM operations on different tiles at the same time, which can also be efficiently programmed onto FPGA.

2. Systolic Array Miro-Architecture

Fig.1 is an example showing how data flows in the systolic array to implement GEMM within six clocks.

- The control signal is converted to propagation, and the weights are preloaded into the systolic array in the first two clocks.
- The control signal is converted to calculation so that the weights no longer flow and the data of the matrix A and B are sent into the systolic array horizontally, and the partial sum is also transmitted in the subsequent time in the vertical direction.
- The calculation result will be output from the bottom of systolic array after the 5th cycle.

![Fig.1 Data Flow of Systolic Array](image_url)

2.1. Single-Buffer Systolic Array Structure

It is assumed that all examples are WS type, the single-buffer structure means that there is only one register in the PE for storing weights, as shown in Fig.2 (a). Under this structure, every time the MAC calculation is performed, the weight data needs to be loaded in advance, and other convolution kernel’s weight data can only wait for the stall during the calculation. Therefore, the calculation and the data waiting occupy half of the working time respectively.
2.2. **Double-Buffer Systolic Array Structure**

In the double-buffer structure, as shown in Fig.2 (b), there are two registers in each PE for weight storage. When one register is used for calculation, the other register will be used to load the next weight for next calculation in advance. In this structure, the MAC unit is utilized in each clock cycle. In fact, it traded area overhead for performance.

3. **Systolic Array Generator**

The systolic array generator is written in Verilog-2001 and adopts a three-layer structure of PE, Tile, and Mesh, and arranges storage banks outside the array to store the matrix data to be calculated.

3.1. **Processing Element**

As shown in Fig. 3, the PE unit designed in this work adopts the weight station mode, and the weight is fixedly stored in the register, and is multiplied and added with the subsequent activation. The propagated signal prop and the two registers W1, W2 in the PE design can implement the behaviour of single-buffer and double-buffer with different configurations.

In single-buffer mode, PE only needs to control prop to activate one register, and after loading the weights, PE should stall for activation coming in.

In double-buffer mode, control prop makes the two registers switch between calculation and propagation states, so that the MAC unit has always corresponding weight and activation input for calculation in clock cycle.
3.2. **Tile**

Fig. 4 shows the structure of Tile which is the smallest unit for operating a complete GEMM, consisting of multiple PEs in a certain shape. For GEMM operation, the elements of one of the matrices need to be pre-loaded into the systolic array by pipeline tapping, and then the other matrix is input in a certain way, after the pulsation calculation, the result can be obtained from the bottom of the systolic array. The proposed Tile can be freely adjusted in width and height to suit different scale computing needs.

3.3. **Mesh**

Mesh is composed of multiple Tiles whose structure can be seen from Fig. 5. Since each Tile can perform a GEMM operation independently, a Mesh composed of multiple Tiles can perform different GEMM operations at the same time. This structure is very suitable for a convolution operation case that different weight matrices while the input activation is unchanged. Because the data enters from the left side of the array and advances to the right as the clock pulsates, it is possible to update the later Tile’s weights when other Tiles are calculating, which makes the calculation continuous without pause.

At the same time, we add storage banks to each input and output section to increase spatial locality and effectively reduce the delay caused by data transferring.
4. Experimental Results

In order to compare with the previous work, we used the proposed systolic array generator and Gemmini generator to generate the same scale Mesh whose size is 16-by-16. Both Meshes use WS mode, and then we compare single-buffer and double-buffer design on the two Meshes about the performance and overhead. The summary of results is shown in Table 1.

The experiment uses Design Compiler and SIMC 130nm process to obtain area and power consumption, and get computational throughput on Nexys-Vedio FPGA with certain frequency constraints. The results of the experiment are shown in Table 2, in which GMACOPS means Giga MAC Operations Per Second.

![Fig.5 Structure of Mesh and Banks.](image)

**Table 1. Synthesis Summary in TSMC 130nm Technology**

| NO. Structure       | Bitwidth  | Area (mm²) | Power (mW) | Freq(Mhz) |
|---------------------|-----------|------------|------------|-----------|
| Proposed Single-Buffer | 32 bit input 64 bit result | 14.3 | 395.04 | 100 |
| Proposed Double-Buffer | 32 bit input 64 bit result | 17.9 | 508.26 | 100 |
| Gemmini Single-Buffer | 32 bit input 32 bit result | 13.2 | 1021.71 | 100 |
| Gemmini Double-Buffer | 32 bit input 32 bit result | 27.1 | 2060.33 | 100 |

**Table 2. FPGA Performance Summary**

| NO. Structure       | Freq(Mhz) | Throughput(GMACOPS) |
|---------------------|-----------|---------------------|
| Proposed Single-Buffer | 100 | 8.89 |
| Proposed Double-Buffer | 100 | 25.59 |
| Gemmini Single-Buffer | 32.5 | 2.62 |
| Gemmini Double-Buffer | 32.5 | 8.27 |

Both Gemmini and out proposed generator have similar characteristics on the difference of single-buffer and double-buffer structure. Because there is one more register inside the PE, the double-buffer...
structure is almost 25% larger than the single-buffer structure in area, the power consumption increases by 28%, and the throughput is increased by nearly 3 times.

Compared with Gemmini, the Mesh generated by the proposed systolic array generator increases the power consumption by 75%, the area overhead is reduced by 34% and it can be easily programmed onto the FPGA with a constraint frequency of 100Mhz, while the Mesh generated by Gemmini can only work normally at less than 32.5Mhz.

5. Conclusion
This work designs and implements a systolic array generator with a three-layer structure, which can freely configure the size of the systolic array and the number of buffers in the PE, and can be used for simultaneous multiple GEMM operations. On the proposed systolic array generator, we compare the performance and cost of two structure systolic array. The experimental results show that the throughput of the double-buffer structure is nearly increased by nearly 3 x than the single-buffer structure with an additional 28% power consumption and 25% area overhead. Compared with the previous work, the proposed systolic array generator reduces power consumption by 75% and area overhead by 34% while the performance remains almost unchanged. Furthermore, it can be easily programmed on FPGA. In future work, we will start to optimize the GEMM matrix mapping algorithm to make the calculation of systolic array more efficient and flexible.

Acknowledgments
This work was supported by National Key Research and Development Program of China (No. 2018YFB0204301)

References
[1] N. P. Jouppi et al. (2017) In-datacenter performance analysis of a tensor processing unit. ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), Toronto, ON, 2017, pp. 1-12.
[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016) Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778.
[3] Google, TPU. (2020). Accessed on: September 17, 2020. [Online]. Available: https://cloud.google.com/tpu.
[4] V. Sze, Y. Chen, T. Yang and J. S. Emer. (2017) Efficient Processing of Deep Neural Networks: A Tutorial and Survey. in Proceedings of the IEEE, vol. 105, no. 12, pp. 2295-2329.
[5] Y. Chen, T. Yang, J. Emer and V. Sze. (2019) Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 2, pp. 292-308.
[6] Kung, H. T. (1982). Why systolic architectures. IEEE computer, 15(1), pp.37-46.
[7] Warden, P. (2015). Why GEMM is at the heart of deep learning. Peter Warden’s Blog. Accessed on: September 17, 2020. [Online]. Available: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning
[8] Genc, H., Haj-Ali, A., Iyer, V., Amid, A., Mao, H., Wright, J., ... & Shao, Y. S. (2019). Gemmini: An agile systolic array generator enabling systematic evaluations of deep-learning architectures. arXiv preprint arXiv:1911.09925.
[9] Bachrach, J., Vo, H., Richards, B., Lee, Y., Waterman, A., Avižienis, R., ... & Asanović, K. (2012, June). Chisel: constructing hardware in a scala embedded language. In DAC Design Automation Conference 2012. pp. 1212-1221.
[10] Lym, S., & Erez, M. (2020). FlexSA: Flexible Systolic Array Architecture for Efficient Pruned DNN Model Training. arXiv preprint arXiv:2004.13027.