Systolic Array Based Accelerator and Algorithm Mapping for Deep Learning Algorithms

Zhijie Yang, Lei Wang(✉), Dong Ding, Xiangyu Zhang, Yu Deng, Shiming Li, and Qiang Dou

College of Computer, National University of Defense Technology, Changsha, China
Leiwang@nudt.edu.cn

Abstract. As the depth of DNN increases, the need for DNN calculations for the storage and computing power of the underlying computing platform is increasing. In this work, we implement an accelerator on FPGA for deep learning algorithms (CNN and RNN). The core computing module of the accelerator is a 32 * 32 systolic array of PEs. A mapping method for variable size of CNN and RNN algorithms is proposed. The experiment result shows that the maximum power consumption of the accelerator is 7.5W@100Mhz, the peak performance is 0.2Tops/s, and the real performance is 7.6Mops@100Mhz when running the 1st layer of LeNet-5.

Keywords: Accelerator · Systolic array · DNN · Data mapping

1 Introduction

At present, almost all large companies are developing their own artificial intelligence chips. Facebook's hardware is optimized for its Caffe2 framework [1]. Amazon is building an ecosystem of cloud infrastructure by AWS [2]. The most notable example of deep learning algorithm accelerators is Google's TPU [3]. Google Data Center has been using TPU to accelerate AI services such as image recognition and language translation.

Compared with CPUs and GPUs, TPU can provide high performance and high energy efficiency. For example, TPU1 can provide 92Top/s with 8-bit integer [3]. Google's TPU brings systolic design, which is an old hardware architecture [4], back to the face of architecture designers. The core computing unit of the TPU is a 256 * 256 systolic array of MACs. The systolic array structure can effectively support the memory intensive and computing intensive features of deep learning algorithms. Google introduced some details of TPU1 in a paper published at ISCA 2017 [3]. However, Google did not disclose much detail about TPU2 and TPU3 until now.

Therefore, in order to design deep learning accelerators based on the systolic array, we need to solve the following problems: How to implement a deep learning accelerator using a systolic array and map different deep learning algorithms to it? The main contributions of this paper are:

(1) An RTL design of the accelerator architecture whose core computing unit is a $32 * 32$ systolic array and necessary peripheral modules.
(2) We propose a method for mapping arbitrary size convolution and matrix multiplication operations to the fixed-sized systolic array to accelerate CNN and RNN computation.

We synthesis the accelerator to Xilinx's FPGA V7 690T, and perform detailed functional verification and performance, power and area analysis. The experiment results show that the inference process of the CNN model and RNN model can be run correctly on this accelerator and be accelerated. The real performance is 7.6Mops@100Mhz when running the first layer of LeNet-5.

2 Related Work

In addition to google [3] and microsoft work [6], the Cambricon designed a new generation of edge intelligent processor Cambricon 1M. Farabet et al. proposed an extensible data flow hardware architecture for running the generic vision algorithm neuFlow on Xilinx V6 FPGA platform [5]. Chen et al. made a custom multi-chip machine-learning architecture [7]. Alwani et al. constructed a fused-layer CNN accelerator for the first five convolutional layers of the VGGNet-E network [8]. Li et al. designed an 8-Bit fixed-point CNN hardware inference engine [9].

3 Systolic Array Design

Figure 1 shows the overall architecture of the proposed accelerator design. It contains the systolic array, input memory, output memory, weight memory, controller, and an AXI interface for data exchange with the host computer.

3.1 Processing Element Design

Figure 2 shows the module diagram of the processing element (PE). The PE contains input registers, part sum registers, weight registers, and the counter.

The input register is used for receiving an incoming data at each cycle from the upper computing unit or input memory for MAC and transferring input data to lower PE. The weight register is used for storing weights and forward weights to the right PE. The part sum register is responsible for keeping the temporary result and continuously accumulate with the new result. The counter is used for counting the cycles, because, for some algorithm, it will take several cycles to propagate the result to the adjacent PEs. When the counter reaches the specified threshold, the final result is transmitted to the upper unit or the output memory.

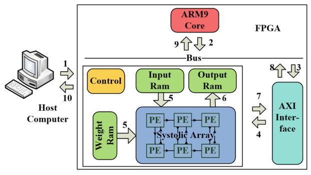


Fig. 1. Overall design of the accelerator.

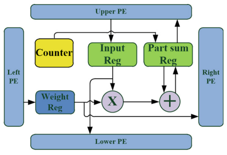


Fig. 2. Systolic array PE.

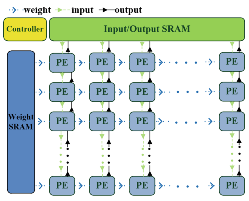


Fig. 3. Two-dimensional systolic array's data flow. (Color figure online)

3.2 Array Structure and Data Flow

As shown in Fig. 3, the structure of a two-dimensional systolic array has the following data flow. In the systolic array, the input data flow, represented by the green dotted solid arrow, is input into the systolic array from the top and propagates to the bottom until the bottommost PE discards it after use. The weight data flow, represented by the blue dot-shaped hollow arrow, is similar to the input data flow but it propagates from the left to the right. The output data flow, represented by the black solid black arrow, is output from each PE and it propagates from the bottom to the top.

3.3 Controller Design

The controller module has the following functions. It controls the data to be read into the memory from the buffer, starts the computation, controls the data to flow into the array, and controls the results to flow out of the array and to write back to memory. Finally, it controls the results to be returned to the buffer to be taken by the host computer.

4 Data Mapping Method

4.1 Mapping of CNN Algorithm

We only discussed the computation of convolution layer at this section. FC layers mapping to the systolic array is similar to RNN algorithm's mapping.

The convolutional layer receives N feature maps as input. Each input feature map is convolved by a $K*K$ size kernel to generate one pixel in one output feature map. The stride is S. M output feature maps will be the set of input feature maps for the next conv layer. Figure 4 shows the mapping to an ideal systolic array, assuming that the size of the systolic array is $M \times (C \times R)$ and the stride is 1. The horizontal axis input is the kernel, the first row is $weight[0][t_i][*][*]$, t_i is from 0 to $N-1$. The second row is $weight[1][t_i][*][*]$ and so on. The data in the second row is provided one cycle later than the element of the same position in the first row, and so on. The vertical axis input is the data block corresponding to the input feature map, as shown in Fig. 4. The input feature map's data flows as similar as the kernel's data flow.

The last point of the first row of the systolic array is completed in $K \times N + R \times C - 1$ cycles. Each PE in the first row saves the output feature map of output channel 0. Assuming that the 1st PE of the 1st row can read the first $point[0,0]$ of the output channel 0 of output feature map and in $K \times N$ and at $K \times N + 1$, it can read the second $point[0,1]$. In this way, the last $point[C-1, R-1]$ is read at $K \times N + R \times C - 1$ cycles. And so forth, the point in time at which each PE completes the operation is represented as an anti-diagonal line as shown in the Fig. 4. After completing the operation, all results are transmitted vertically.

Assume that the size of the systolic array is $A \times B$. The ideal systolic size of the algorithm is $M \times C \times R$. It can be seen as a box with the size of $A \times B$, overlaid on the ideal systolic array layout, and filled with zeros where there is no data at the boundary.

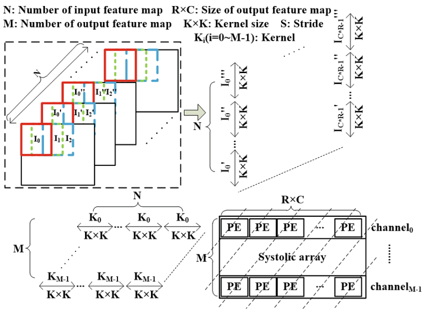

N: Number of input feature map R×C: Size of output feature map
M: Number of output feature map K×K: Kernel size S: Stride
K_i(i=0~M-1): Kernel
N
R×C
K₀
K×K
K_{M-1}
M
PE
Systolic array
channel₀
channel_{M-1}

Fig. 4. A CNN ideal mapping algorithm.

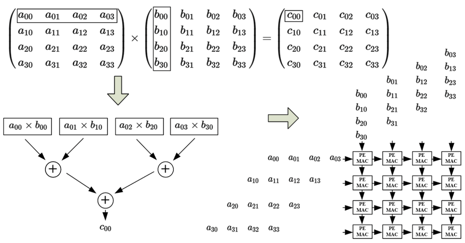


Fig. 5. A RNN mapping algorithm.

4.2 Mapping of RNN Algorithm

The core operation of RNN is matrix multiplication used by forwarding propagation. And matrix multiplication's operation method is very suitable for the systolic array. Both of them use a row to multiple a column.

Thus, as shown in Fig. 5, the left matrix is input in rows of the systolic array and the right matrix in the columns of the systolic array. Then the matrix product from the corresponding position of the systolic array can be got. Consider a matrix multiplication $A \times B = C$ and a systolic array of size $M \times N$. The size of A is $r \times s$, the size of B is $s \times t$ and size of C is $r \times t$. When $r <= M$, $t <= N$, the matrix multiplication can be operated by the systolic array in one pass. While if $r > M$ or $t > N$, it is necessary to split r or t. Separate the rows of matrix A into parts and divide the columns of matrix B into parts. When the result is not an integer, PEs not used in the array are filled with zeros.

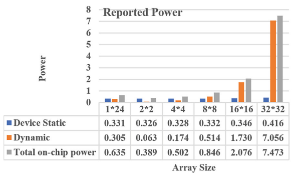
5 Experimental Result

5.1 Implementation

The accelerator is implemented in Verilog RTL. Vivado 2017.04 is used for synthesis and implementation. The implementation platform is Xilinx's FPGA V7 690T. This paper implements a one-dimensional $1 * 24$ systolic array, $2 * 2$, $4 * 4$, $8 * 8$, $16 * 16$ and $32 * 32$ systolic arrays, and compares the results. The clock frequency is set to 100 MHz. Its power consumption is about 7.5 W. The comparison of power consumption of different size systolic array is shown in Fig. 6(a). As shown in the Fig. 6(b), the on-chip resource consumption ratios of the various size of systolic arrays are compared.

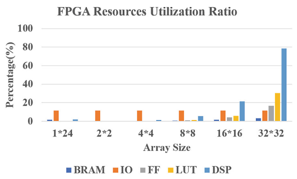
5.2 Performance Evaluation

In the design, the operand of each PE unit is 32-bit fixed-point, and the clock frequency is set to be 100MHz. Because the performance metrics are limited by the size of the specific problem, our metrics in subsequent performance evaluations are subject to the above conditions. ***Peak Performance:*** The peak performance is 0.2Tops@100Mhz when the data is completely filled with the array. ***Throughput:*** The peak throughput of the $32 * 32$ systolic array is 1600 Mresults/s. The real throughput rate calculated is 60.3 Mresults/s when running the 1st layer of LeNet-5. ***Performance/Area Ratio:*** As shown in Fig. 6(c), as the array grows, the performance/area ratio decreases. This also shows that as the number of computing units increases, the performance (throughput) revenue per PE tends to decrease.

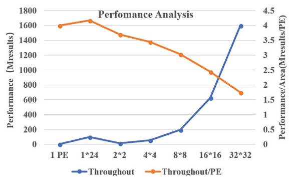


	1*24	2*2	4*4	8*8	16*16	32*32
■ Device Static	0.331	0.326	0.328	0.332	0.346	0.416
■ Dynamic	0.305	0.063	0.174	0.514	1.730	7.056
■ Total on-chip power	0.635	0.389	0.502	0.846	2.076	7.473

(a) Power consumption evaluation



(b) Resources utilization



(c) Throughout and Throughout/PE

Fig. 6. Comparison of high level synthesis results of the systolic array of different sizes.

6 Conclusion and Future Work

This work proposes the implementation of a DNN accelerator based on a $32 * 32$ systolic array. Then the data mapping method for mapping variable sizes of CNNs and RNNs to the systolic array is proposed. In the future, we will enlarge the systolic array and design an instruction set in order to satisfy the requirements of the controller so that the accelerator can achieve more complex functions.

Acknowledgment. This project is supported by HGJ2017ZX01028103.

References

1. Hazelwood, K., Kalro, A., Law, J., Lee, K., Lu, J., Noordhuis, P., et al.: Applied machine learning at Facebook: a datacenter infrastructure perspective. In: IEEE International Symposium on High Performance Computer Architecture, pp. 620–629. IEEE Computer Society (2018)
2. Tang, B., S.O. Information, Y.N. University: Case study of the application of field programmable gate array FPGA in the smart skill. Application of IC (2018)
3. Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., et al.: In-datacenter performance analysis of a tensor processing unit, pp. 1–12 (2017)
4. Kung, H.T., Leiserson, C.E.: Systolic arrays (for VLSI). In: Proceedings of Sparse Matrix Conference, pp. 256–282 (1978)
5. Farabet, C., Martini, B., Corda, B., Akselrod, P.: NeuFlow: a runtime reconfigurable dataflow processor for vision. In: Computer Vision and Pattern Recognition Workshops, vol. 9, pp. 109–116. IEEE (2011)
6. Chung, E., Fowers, J., Ovtcharov, K., Papamichael, M., Caulfield, A., Massengill, T., et al.: Serving DNNs in real time at datacenter scale with project brainwave. IEEE Micro **38**(2), 8–20 (2018)
7. Chen, Y., Sun, N., Temam, O., Luo, T., Liu, S., Zhang, S., et al.: DaDianNao: a machine-learning supercomputer. In: IEEE/ACM International Symposium on Microarchitecture, vol. 5, pp. 609–622. IEEE (2014)
8. Alwani, M., Chen, H., Ferdman, M., Milder, P.: Fused-layer CNN accelerators. In: IEEE/ACM International Symposium on Microarchitecture, pp. 1–12. IEEE (2016)
9. Li, Z., et al.: Laius: an 8-bit fixed-point CNN hardware inference engine. In: 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC). IEEE (2017)