Bit-serial systolic accelerator design for convolution operations in convolutional neural networks

Lin Li¹,², Jianhao Hu¹, a), Qiu Huang¹, and Wanting Zhou¹

Abstract The accuracy of Convolutional Neural Networks (CNNs) has exceeded the human level in many fields, but the high computation complexity is one of the main challenges for CNNs applied in the mobile or embedded devices. In this paper, we provide a hardware accelerator scheme for the convolution operations in CNNs, which adopts the bit-serial systolic architecture. Implementation results show that the proposed scheme can reduce the area by about 64%, increase the maximum frequency by about 4.4 times and increase the hardware efficiency by about 1.2 times compared with the state-of-the-art Eyeriss architecture.

Keywords: accelerator, bit-serial, convolution operations, systolic architecture

Classification: Integrated circuits (memory, logic, analog, RF, sensor)

1. Introduction

With the massive growth of information data, the significant improvement of computing power and the continuous optimization of algorithm models, artificial intelligence (AI) has drawn dramatic attention in research and industry [1, 2]. As the mainstream AI algorithm [3], Deep learning (DL) is widely applied in computer vision [4], speech processing [5], etc. Thereinto, the Convolutional Neural Network (CNN) has been one of the most representative algorithms due to the excellent learning performance [6]. However, the CNN performance relies on the deep, wide and complex network with huge computation complexity and storage requirements, which limits the applications in mobile or embedded devices [7, 8, 9]. Therefore, it is of great significance to study the CNN accelerator with high hardware efficiency.

According to the research results [10], more than 90% of the CNN computation load is concentrated on convolution layers, the core of which is the convolution operation. Thus, the key to the design of CNN accelerators is how to optimize the convolution operation. Eyeriss [11], developed by MIT, is the most high-efficiency architecture for accelerating the convolution operation in the publications. With the Row Stationary (RS) data-flow structure, Eyeriss scheme can maximize the reuse rate of all data involved in the convolution layer [12]. However, the multiplier and accumulator (MAC) architecture used for the convolution operation in Eyeriss consumes a lot of combinational logic resources, and the complex connection between the computation units also increases the hardware area [13]. Besides, the long critical path is another issue of Eyeriss [14].

In this paper, we provide a high efficient hardware architecture, which is the first (to the best of our knowledge) to combine the bit-serial processing and systolic array together to accelerate the convolution operation in CNNs. The bit-serial scheme is used to reduce the logic complexity of computation units and the connection complexity of wires. The systolic array is adopted to reduce the data access requirements between computation units and memory units. In the proposed scheme, the processing elements (PEs) in the systolic array use novel pipelined bit-serial multipliers, and combine the signed bit-serial accumulators (ACCs) to realize the MACs. With the simple and pipelined architecture, the proposed scheme can greatly improve the clock frequency and silicon utilization, reduce wire resources, balance the throughput, latency and area. We analyze the computation performance of the proposed scheme, and find that it can achieve the same calculation results with the traditional structure. We also implement the proposed scheme with IC compiler (ICC) of Synopsys under 65nm CMOS technology. Compared with Eyeriss, our work can increase the maximum frequency by about 4.4 times @1.25GHz, reduce the area by about 64%, and increase the hardware efficiency by about 1.2 times.

The rest of this paper is organized as follows. Section II presents the analysis of CNNs. We design the systolic array in Section III. Section IV provides the proposed scheme and details the novel bit-serial multiplier. We analyze the performance of the proposed scheme in Section V. Section VI concludes this paper.

2. Analysis of CNNs

The research on CNNs began in 1980s, when Kunihiko Fukushima proposed the predecessor of CNNs, i.e. Neocognitron [15]. LeNet-5 [16], which was proposed by Yann Lecun in 1998 to solve the handwritten digit recognition on MNIST data set, is the earliest CNN model. In 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), Alex Krizhevsky proposed the AlexNet [17], which won the champion of the image classification task. Its classification accuracy is far higher than the results achieved by traditional methods, so as to start the research upsurge of CNNs. With the improvement of the network structure and the increase of the network depth, various CNNs, including VGGnet [18],
GoogLeNet [19] and ResNet [20], have become the winning algorithms of the ImageNet competition.

2.1 CNN structure

CNN is a deep feedforward neural network with the local connection, weight sharing and pooling characteristics, which use the back propagation algorithm to train weights [21]. It is generally composed of convolutional layers, rectified linear units, pooling layers and full connection layers [22]. Fig. 1 shows the typical CNN structure.

2.2 Computational complexity of CNNs

Based on the LeNet-5 model, we analyze the calculation load of the convolution layers, pooling layers and full connection layers respectively. The calculation load can be measured by floating-point operations (FLOPs). Assume the height and width of the convolution kernel be \( K_h \) and \( K_w \), the height and width of the input image be \( I_h \) and \( I_w \) with the channel number being \( C_{in} \), and the height and width of the feature map be \( H \) and \( W \) with the channel number being \( C_{out} \).

2.2.1 Convolution layer

There are two convolution layers C1 and C3 in Lenet-5, and the calculation load can be defined as:

\[
C_{\text{FLOPs}} = (K_h \times K_w \times C_{in} + 1) \times C_{out} \times H \times W
\]  

(1)

2.2.2 Pooling layer

Generally, a convolution layer is followed by a pooling layer, thus the calculation load of the two pooling layers S2 and S4 can be defined as:

\[
P_{\text{FLOPs}} = (P_h \times P_w + 1) \times C_{out} \times H \times W
\]  

(2)

2.2.3 Full connection layer

Lenet-5 has three full connection layers, and the calculation load can be defined as:

\[
F_{\text{FLOPs}} = (I_h \times I_w \times C_{in} + 1) \times C_{out} \times H \times W
\]  

(3)

Based on the above analysis, Fig. 2 shows the calculation distribution of different layers in LeNet-5. We can conclude that convolutional layers account for about 80.3% of the total calculation load.

We also summarize the classical CNNs during recent years, as shown in Table I. We can find that the calculation load is mainly concentrated on the convolution layers for all these CNN models. With the improvement of accuracy, the depth and complexity of the network is increasing, as well as the higher computation load ratio of convolution layers.

2.3 Convolution principle in CNNs

The convolution layers function is to extract different features of the input image, in which the core computation is the convolution operation. Because of the high time and space complexity, the convolution operation is the optimization target in this paper.

If the size of the input image is \( n \times n \), the size of the convolution kernel is \( f \times f \), and \( s \) is the feature map, the convolution operation in CNNs can be defined as below:

\[
s[i,j] = \sum_{p=0}^{f-1} \sum_{q=0}^{f-1} x[i+p, j+q] \omega[p, q] \\
0 \leq i, j < n - f + 1
\]  

(4)

Fig. 3 shows the convolution operation process of a \( 5 \times 5 \) gray image and a \( 3 \times 3 \) convolution kernel, which can output a \( 3 \times 3 \) feature map with an iterative step of 1 and no filling. The kernel can be treated as a sliding window, which scans the input image from top to bottom and from left to right. The feature map can be obtained from the inner products of the kernel value and the value of its corresponding position on the input image.

3. Systolic array design for convolution

Systolic array is a PE network that pumps data into and out of PEs for rhythmic computations [23]. Because the speed of data access is often much lower than that of data processing, the processing ability of a system is largely limited by the ability of the memory access. Systolic array allows data to flow as much as possible in PEs, essentially reusing the input data to balance the time consumption between I/O and computations. It works well for the computer-bound
We take the convolution operation shown in Fig. 3 as an example to analyze the treatment method of the input data. For the process of a row, if the length of the input data is $n$ and the length of the kernel is $f$, we can conclude that the length of data $A$ after the arrangement is $f \times (n - f + 1)$. And $A$ can be expressed as:

$$\text{for } j = 1 \text{ to } f$$

$$\text{for } k = 1 \text{ to } (n - f + 1)$$

$$A(i) = X(j + k - 1), 1 \leq i \leq f \times (n - f + 1)$$

We take the convolution operation shown in Fig. 3 as an example to analyze the treatment method of the input data in detail. Table II shows the complete data arrangement for the systolic array.

### 3.1 Systolic array mapping

As a computer-bound model, the convolution operation in CNNs consists of a large number of multiplications and additions with strong regularity. Therefore, it can be realized simply and efficiently with the systolic array. Referring to the convolution principle in CNNs, it is necessary to rearrange the input data of the systolic array to ensure the accuracy and efficiency of the convolution operation. For the process of a row, if the length of the input data $X$ is $n$ and the length of the kernel is $f$, we can conclude that the length of data $A$ after the arrangement is $f \times (n - f + 1)$. And $A$ can be expressed as:

$$\text{for } j = 1 \text{ to } f$$

$$\text{for } k = 1 \text{ to } (n - f + 1)$$

$$A(i) = X(j + k - 1), 1 \leq i \leq f \times (n - f + 1)$$

We take the convolution operation shown in Fig. 3 as an example to analyze the treatment method of the input data in detail. Table II shows the complete data arrangement for the systolic array.

### 4. Bit-serial systolic architecture

Systolic array is characterized by modularization and regularization. Typically, all PEs in the array are the same MACs based on bit-parallel computing [25]. However, a large number of MACs in the systolic array become the key issue for the acceleration performance of the convolution operation in view of the critical path, areas and power consumption [26]. Assuming that the bit-width of the convolution operation is $w$-bit, different from the traditional bit-parallel MACs to complete the calculation of all $w$-bit data in one clock, the bit-serial computing method serializes each input data and disperse the calculation in one clock. Therefore, the bit-serial scheme has the advantages of less occupied area and low power consumption due to its simple structure, short critical path and less wiring resources [27]. In addition, the connection wires in the bit-serial scheme is much less than that in the bit-parallel scheme.

Therefore, we design the bit-serial MACs to achieve the bit-level supper pipeline for the systolic array, which can simplify the hardware circuit by the bit-serial computing and improve the throughput greatly by the high parallelism. This section first describes the accelerator architecture based on the bit-serial scheme, and then details the design and optimization of the pipelined bit-serial multiplier.

#### 4.1 Proposed accelerator architecture

In this paper, $3 \times 3$ PE array is taken as an example to illustrate the proposed scheme as shown in Fig. 5. The input and output are processed into bit-serial data by global buffers. Different from the typical systolic array, we have made some improvements. MACs are decomposed into the multiplier and accumulator respectively, while PEs perform multiplication, and the results are transferred to accumulators in the corresponding column for addition and accumulation. There are two benefits with this scheme. First, It can reduce the numbers of accumulators. Second, it is convenient for optimizing the multiplier and the accumulator respectively, which makes the design more simple and flexible. Another point to emphasize is that we apply novel pipelined bit-serial multipliers and novel signed bit-serial accumulators [28] in the proposed architecture.
4.2 Pipelined bit-serial multiplier

All bit-serial data is represented by the fixed-point two’s complement in this paper. A of $w$-bit can be expressed as:

$$A = a_{w-1}a_{w-2} \cdots a_0$$  \hspace{1cm} (6)

where $a_i$ is 0 or 1, $0 \leq i \leq w-1$. $a_{w-1}$ is a sign bit, 0 is for positive, and 1 for negative. Without losing generality, the numerical interval of $A$ is $[-1,1-2^{-w+1}]$, and can be given by the following equation:

$$A = -a_{w-1} + \sum_{i=0}^{w-1} a_{w-1-i} + 1$$  \hspace{1cm} (7)

Since there is a sign bit, two’s complement multiplication can be realized by means of the shifting plus form, only in the case of the sufficient sign bit expansion. It is worth noting that if the multiplier extends 1-bit sign bit, the overflow cannot be completely eliminated in the numerical interval $[-1,1-2^{-w+1}]$. In order to get the correct product, the multiplier must be in the interval $[-1/2,1/2]$ [29].

The multiplication vertical formula of $4$-bit two’s complement data $a_3a_2a_1a_0$ and $b_3b_2b_1b_0$ is shown in Fig. 6, where $pp$ represents the partial product. If one of the inputs has to be positive, it can be used as a multiplier to remove the last sign bit added in the multiplication vertical formula.

To realize the pipelined processing, we know that if two $w$-bit data are multiplied, the result should be $2w - 1$ bits. Therefore, it is necessary to truncate $2w - 1$ bits to $w$ bits to ensure the pipelining of the bit-serial computing. That is to say, the computing number of each clock entering the multiplier is the same as that of the multiplier output. As shown in Fig. 6, it is necessary to discard the data $pp$ in the box when the pipelined computing is implemented, which may result in the truncation error of the finite word length.

According to the systolic mapping principle and Horner’s law, a $4$-bit pipelined multiplier design, corresponding to Fig. 6, based on bit-serial computing is shown in Fig. 7. The multiplier is composed of AND gates, registers, data shifters and full adders. It inputs data and outputs results from the least significant bit (LSB) serially. The data shifter is marked by the red dotted box in Fig. 7, and consists of a single bit register and a single bit multiplexer (MUX), by which the data with different delay is selected at the corresponding time. The shifter can expand the sign bit of the two’s complement data and truncate the end bit of the result. Using the pipeline method, the combinational logic is divided into several small combinational logic, and a register is inserted in the middle to store the data. The registers and full adders constitute the shift plus structure. If the AND gate and MUX are ignored, the critical path of the multiplier is $w$ full adders, which is proportional to the word length of the processed data.

In this paper, a new multiplication vertical formula and its corresponding pipelined multiplier are obtained by using the retiming operation as shown in Fig. 8 and Fig. 9 respectively, of which the full adders in red dotted boxes are corresponding. The critical path of this new multiplier is fixed as two full adders. The multiplier can achieve a higher system frequency. However, the actual performance will be slightly lost due to the loss of the carry caused by the sum of $pp_1$ and $a_0b_2$.

In the same way, we implement the convolution operation with a word length of 10-bit to verify the performance of the proposed scheme in the next section. The 10-bit pipelined bit-serial multiplier applied is shown in Fig. 10. The longest critical path reduces from 1 AND Gate + 2 full adders + 8
MUXs to 1 AND Gate + 8 full adders + 8 MUXs.

5. Implementation

The convolution layers in CNNs are independent and repetitive. There are also a large number of independent and repetitive convolution operations in each convolution layer that can be performed in parallel. Therefore, as long as the minimal functional module which can complete the convolution operation is implemented, the proposed architecture can be compared with the existing accelerators objectively and accurately.

We implement the bit-serial systolic architecture with 9 PEs to accelerate the convolution operation shown in Fig. 3. Verilog HDL is used for its behavior level description, and we complete the functional verification with Vivado tool. Then, based on the SMIC 65nm CMOS technology, the ICC of Synopsys is used for logic synthesis and implementation. We analyze and compare the performance of the proposed accelerator architecture with Eyeriss [30] under the same implementation conditions. The experimental results are shown in Table III.

The area, energy consumption and hardware efficiency are three important indicators to evaluate the performance of hardware architecture. From Table III, we can see that the bit-serial systolic architecture designed in this paper occupies only 36% of Eyeriss in area, reduces the power consumption by about 12%, and increases the maximum frequency by about 4.4 times.

Because the proposed scheme is different from Eyeriss, in order to compare them fairly, the peak throughput per gate (MOPS/Gate) is used as the hardware efficiency to measure the chip’s computing performance, which is equal to the operation throughput divided by the hardware area. For fairness, this indicator of our scheme has been parallelized, i.e. divided by 10. Compared with Eyeriss, our architecture improves the hardware efficiency by about 1.2 times.

6. Conclusion

To solve the challenge of applying CNNs with high computation complexity in the mobile and embedded devices, this paper proposes bit-serial systolic architecture to accelerate the convolution operations. We design a novel pipelined bit-serial multiplier used in PEs, which realizes the bit-level super pipeline of the whole architecture. The proposed bit-serial systolic architecture can reduce the logic and connection complexity as well as the critical path, and improve the computing throughput with high parallelism. Thus, the proposed scheme can be deployed in CNN applications with high hardware efficiency.

References

[1] M.L. Ginsberg, *Essentials of Artificial Intelligence* (Morgan Kaufmann Publishers, San Francisco, 1993) 1.
[2] D. Acemoglu and P. Restrepo: “Artificial intelligence, automation and work,” National Bureau of Economic Research (2018) (DOI: 10.2139/ssrn.3098384).
[3] I. Arel, et al.: “Research frontier: deep machine learning—a new frontier in artificial intelligence research,” IEEE Comput. Intell. Mag. 5 (2010) 13 (DOI: 10.1109/MCI.2010.938364).
[4] A. Voulodimos, et al.: “Deep learning for computer vision: a brief review,” Computational Intelligence and Neuroscience (2018) 2018 (DOI: 10.1155/2018/7068349).
[5] D. Wang and J. Chen: “Supervised speech separation based on deep learning: an overview,” IEEE/ACM Trans. Audio, Speech, Language Process. 26 (2018) 1702 (DOI: 10.1109/TASLP.2018.2842159).
[6] J. Gu, et al.: “Recent advances in convolutional neural networks,” Pattern Recognition 77 (2018) 354 (DOI: 10.1016/j.patcog.2017.10.013).
[7] Y.D. Kim, et al.: “Compression of deep convolutional neural networks for fast and low power mobile applications,” Computer Science 71 (2015) 576 (DOI: 10.1109/ICC.2015.73).
[8] B. Zhao, et al.: “An energy-efficient coarse grained spatial architecture for convolutional neural networks AlexNet,” IEICE Electron. Express 14 (2017) 20170595 (DOI: 10.1587/elex.14.20170595).
[9] M. Capra, et al.: “An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks,” Future Internet 12 (2020) 113 (DOI: 10.3390/fi12070113).
[10] V. Sze, et al.: “Efficient processing of deep neural networks: a tutorial
and survey,” Proc. IEEE 105 (2017) 2295 (DOI: 10.1109/JPROC.2017.2761740).

[11] Y.-H. Chen, et al.: “Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks,” ACM SIGARCH Computer Architecture News 44 (2016) 367 (DOI: 10.1145/3007787.3001177).

[12] Y.-H. Chen, et al.: “Using dataflow to optimize energy efficiency of deep neural network accelerators,” IEEE Micro 37 (2017) 12 (DOI: 10.1109/MM.2017.54).

[13] C. Xin, et al.: “COSY: an energy-efficient hardware architecture for deep convolutional neural networks based on systolic array,” IEEE 23rd ICPADS (2017) 180 (DOI: 10.1109/ICPADS.2017.00034).

[14] H. Sharma, et al.: “Bit fusion: bit-level dynamically composable architecture for accelerating deep neural network,” 2018 ACM/IEEE 45th ISCA (2018) 764 (DOI: 10.1109/ISCA.2018.00069).

[15] K. Fukushima: “Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” Biological Cybernetics 36 (1980) 193 (DOI: 10.1007/BF00344251).

[16] Y. Lecun, et al.: “Gradient-based learning applied to document recognition,” Proc. IEEE 86 (1998) 2278 (DOI: 10.1109/5.726791).

[17] A. Krizhevsky, et al.: “ImageNet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems 60 (2017) 84 (DOI: 10.1145/3065386).

[18] C. Szegedy, et al.: “Going deeper with convolutions,” IEEE Conf. Comput. Vis. Pattern Recognit. (2015) (DOI: 10.1109/CVPR.2015.7298594).

[19] K. He, et al.: “Deep residual learning for image recognition,” IEEE Conf. Comput. Vis. Pattern Recognit. (2016) 770 (DOI: 10.1109/CVPR.2016.90).

[20] A. Karpathy, et al.: “Large-scale video classification with convolutional neural networks,” IEEE Conf. Comput. Vis. Pattern Recognit. (2014) (DOI: 10.1109/CVPR.2014.223).

[21] A. El-Sawy, et al.: “CNN for handwritten arabic digits recognition based on LeNet-5,” International Conference on Advanced Intelligent Systems and Informatics 533 (2016) 566 (DOI: https://doi.org/10.1007/978-3-319-48308-5_54).

[22] H.T. Kung and C.E. Leiserson: “Systolic arrays (for VLSI),” Symposium on Sparse Matrix Computations (1979) 256.

[23] H.T. Kung and C.E. Leiserson: “Why systolic architectures?,” IEEE Comput. 15 (1982) 37 (DOI: 10.1109/MC.1982.1653825).

[24] Y.-C. Chen, et al.: “Two-dimensional multiply-accumulator for classification of neural signals,” IEEE Access 6 (2018) 19714 (DOI: 10.1109/ACCESS.2018.2814625).

[25] J.Z. Shen, et al.: “Towards a multi-array architecture for accelerating large-scale matrix multiplication on FPGAs,” 2018 IEEE ISCAS (2018) (DOI: 10.1109/ISCAS.2018.8351474).

[26] K.K. Parhi: VLSI Digital Signal Processing Systems: Design and Implementation (John Wiley & Sons, New Jersey, 2007) 353.

[27] R. Lyon: “Two’s complement pipeline multipliers,” IEEE Trans. Commun. 24 (1976) 418 (DOI: 10.1109/TCOM.1976.1093315).

[28] Y.-H. Chen, et al.: “Eyeriss v2: a flexible accelerator for emerging deep neural networks on mobile devices,” IEEE J. Emerg. Sel. Topics Circuits Syst. 9 (2019) 292 (DOI: 10.1109/JETCAS.2019.2910252).