ATLANTIS PRESS

Advances in Computer Science Research, volume 86

International Symposium on Communication Engineering & Computer Science (CECS 2018)

The Storage Structure of Convolutional Neural Network Reconfigurable Accelerator Based on ASIC

Jingqun Li [a], Mingjiang Wang [b] and Hongli Pan [c]

School of Electronic and Information Engineering, Harbin Institute of Technology, Shenzhen 518000, China.

[a] jingqunli520@163.com, [b] mjwang@hit.edu.cn, [c] panhonglihit@163.com

Abstract. With the development of deep convolutional neural networks (CNNs), it can be achieved higher accuracy in many aspects, including computer vision, speech and natural language processing. Performance efficiency of CNN at the hardware level requires overcoming the large calculation-related problems, so memory bandwidth and power budgets, should be in economical limits. CNNs models also adopts different kernel sizes, depends on the application nature, therefore it is important for designed architecture to be reconfigurable. In this work, we propose a new high-performance multi-precision reconfigurable architecture (MPRA) and optimize it for recent CNNs using 3×3/5×5/7×7 convolution such as AlexNet, GoogLeNet and ResNet with 16-bit fixed and 8-bit fixed precision. The architecture synthesized on 65 nm CMOS technologies achieves average performance (GOPS) of 276.5 in 16bit×16bit and 1105.9 in 8bit×8bit mode, running at 640 MHz and 1 V with a power dissipation of 599 mW respectively. Compared to state-of-the-art designs, the proposed architecture achieves 2.36× energy efficiency, 2.4× to 6.8× area efficiency, and 16.3% to 27.4% higher computational efficiency for AlexNet benchmarked reference.

Keywords: Convolutional Neural Network, CNNs, accelerator, CMOS, AlexNet.

1. Introduction

Deep convolutional neural networks CNNs can achieve unprecedented accuracy for many tasks such as object recognition [1], detection [2] and scene understanding. However, two important issues are faced in the applications which based on CNNs: the computational complexity is much higher than that of traditional methods and the parameter data transfer requires high memory bandwidth.

Improving CNN performance demands overcoming the throughput and memory bandwidth issues associated with the large number of computations. The traditional way to speed up CNN is to use of fast GPUs that can handle matrix multiplication and able the researchers to train networks 10 or 20 times faster [3-4], but the high-power consumption of GPUs cannot be neglected. In addition to the GPUs, various types of FPGA (Field Programmable Gate Array) implementations have been proposed [5-6]. Although FPGAs have shown good promise in efficiently computing CNNs, but at the same time, due to the flexibility of programming, there will be a lot of redundancy in the circuit.

In this paper, we propose an efficient adoptive architecture to accelerate the CNNs, architecture, includes a combination of 24 parallel processing engines (PEs) where each engine can support different 9 (16-bit×16-bit) MAC, 18(16-bit by 8-bit) MAC, or 36(8-bit by 8-bit) operations. we will focus on convolutional layers. At the same time, this paper optimizes the data structure by pre-storing and pre-reading the part of the image data and weight parameters. Integrating a Registers Group in the middle to pre-read the image data and parameters to the Buffer instead of read directly from the DRAM and flowed into the space computing array. This way system becomes more time efficient and achieves higher computational efficiency. The input image data and weight parameters are transferred from the external off-chip memory, to the separated on-chip data buffer and parameter buffer, and then transferred into the PE units for convolution operation. We use a SRAM to store the intermediate convolution results this improve the overall computational efficiency.

2. The Architecture for Top-level

Top-level architecture block diagram is given in Fig.1; it is mainly composed of PE array, Data Register Group, Weight Register Group and P_SRAM. The PE array consists of 24 parallel PEs, a

single PE can compute a 3×3(16-bit) convolution or four 3×3(8-bit) convolution. By reconfiguring the PE array, the combination of 3 PEs can perform a 5×5 convolution operation, therefore, this array can simultaneously perform eight 5×5 convolutions. Therefore, the PE array is able to calculate different shapes of convolutions through combinations of PEs. The size of Data Buffer is designed to be 128 KB for temporary storage of image data and part of the intermediate convolution results. The size of Parameter Buffer is designed to be 1 KB, to store the convolution parameter values, including filter weights and offset values. The intermediate convolution results are stored in P_SRAM, which is 48 KB.

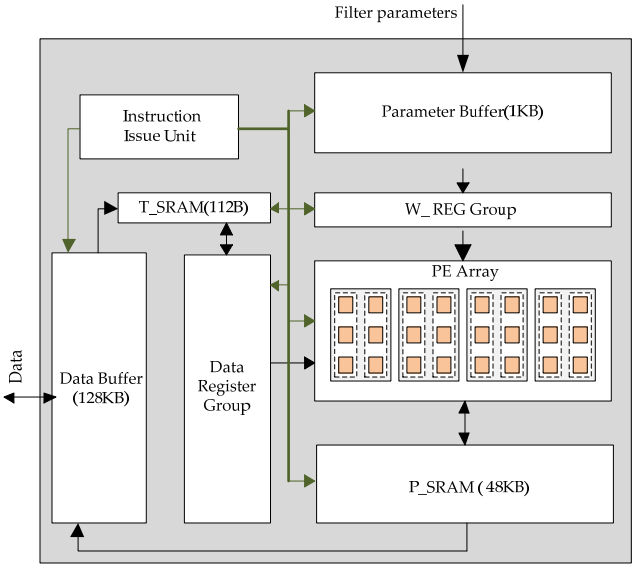

Fig. 1 Top-level architecture block diagram

3. The Structure of Data Buffer

Data buffer is mainly used to store input image data and features to reduce DRAM access. However, for different layers of CNN convolution operations, the amount of image data or features needed are different.

In Fig.2 it is shown that, 56 × 56 image is applied as input to, 3×3 convolution layer of AlexNet. All 56 rows of image data are grouped into set of eight rows (r0… r7), where each set contains seven independent lines of pixels i.e. (a0 to a6) and one redundant line (a7). The redundant line comes from the first line of the next set. All entities of same column in one set are stored sequentially in the Data Buffer, as shown in the figure. First column entities as denoted by D0 are stored in the Data Buffer unit at address 0 and similarly others can be stored, this facilitates data reuse and reduces the time computations.

When the 3 × 3 convolutions are conducted, the 24 PEs simultaneously read the same data corresponding to the 3 rows by 3 columns of input image data, and different parameter values corresponding to different filters. The output results, representing different output feature maps, are stored in the corresponding P_SRAM units.

The following is to analyze the 3 × 3 convolution of data in transition from r0 to r1. It can be seen that the 3 × 3 convolution only convolve, the line a0 to a6 in r0. To perform convolution with the center data in line a7 of next row r1 still required a line a6 from previous row r0, therefore the 112 bytes T_SRAM is used to temporarily store the data from previous row, which is required in the convolution of the adjacent row. For example, when the data register group reads data of r0 for convolution processing, the data of line a6 from r0 are stored temporarily in T_SRAM. Stored data from T_SRAM then transferred to Data Register Group directly, to perform the convolution of center data in line a7 of r1. In this way a small size, T_SRAM ensures data reuse and compact data storage.

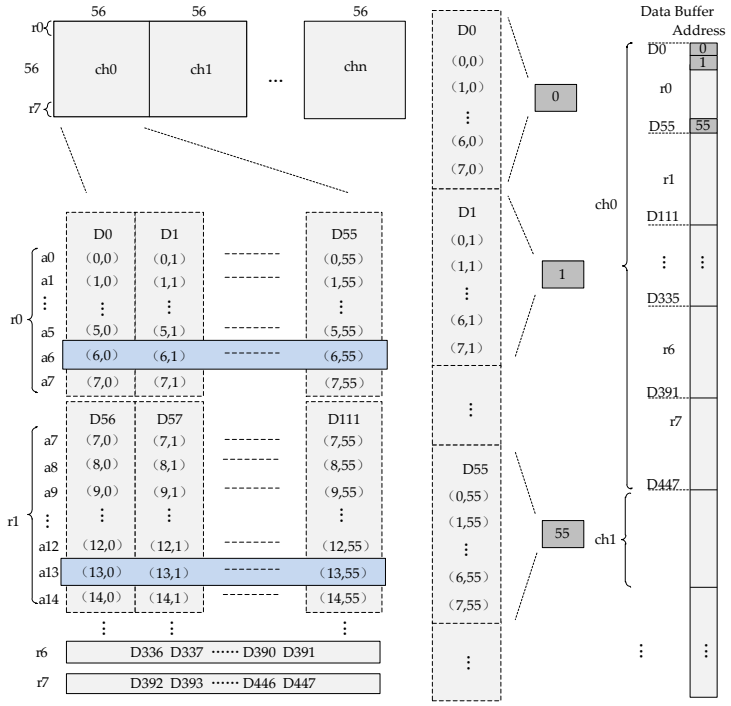

Fig. 2 Image 56 × 56 Storage Structure Diagram

As shown in Fig. 3, 28 × 28 image is applied as input to, 5×5 convolution layer of AlexNet. The 5 × 5 convolution with the center data in line (a0 to a5), can be done by data in row r0 only, but when convolution is performed on the data of line a6 it requires, data of line a8 from row r1. Therefore, before calculating on line a6 data, we need to pre-read Data Buffer for line a8 from row r1, and then stored to the T_SRAM. When calculating convolutions in row r1, line a5 and a6 data from row r0 is required, so after finishing convolutions in row r0, the T_SRAM is filled by data of line a5, a6 from row r0. Similar method is applied for other, 5 × 5 convolution operations, which lead us to correct results.

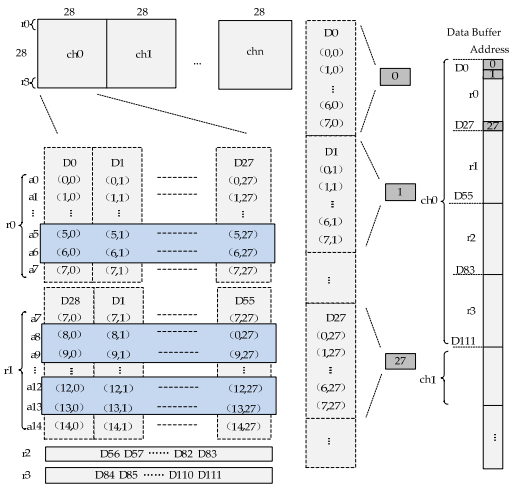

Fig.3 Image 28 × 28

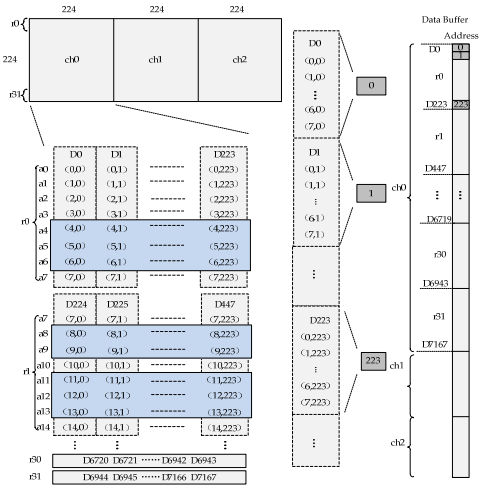

Fig.4 Image 224 × 224

As shown in Fig.4, an image of 224×224 is applied as input to, 7×7 convolution layer of AlexNet. Similar approaches are adopted here for robust convolution operation with the help of T_SRAM, which reduces data storage redundancy.

4. Experiments Results

The proposed architecture is synthesized in 65nm TSMC CMOS, when feature map data and weight parameters are all represented by 16-bit fixed point; it achieves a peak performance of 276.5

GOPS, running at 640 MHz and 1 V with a power dissipation of 599 mW. It implements a frame rate of 176.7 fps on the AlexNet convolution layers, respectively. Similarly, when the feature map data and weight parameters are all represented by 8-bit fixed point. It achieves a peak performance of 1105.9 GOPS running at 640 MHz and 1 V with a power dissipation of 599 mW. The performance of this architecture is shown in Table 1.

Table 1. Architecture performance

Technology	TSMC 65 nm CMOS	
Word Bit-width	16-bit fixed	8-bit fixed
Core Area	4.64 mm2	4.64 mm2
On-chip SRAM	177 kB	177kB
# of PE	24 (9 16b×16b MAC per PE)	24 (36 8b×8b MAC per PE)
Supply Voltage	1 V	1 V
Core Frequency	640 MHz	640 MHz
Peak Performance	276.5 GOPS	1105.9 GOPS
Power	599 mW	599 mW

5. Summary

This paper presents a high-performance reconfigurable architecture and optimize it for recent CNNs using 3×3/5×5/7×7 convolution such as AlexNet, GoogLeNet and ResNet with 16-bit fixed and 8-bit fixed precision. The designed structure consists of combination of 24 parallel PEs where each engine can contain 9(16bit×16bit) MAC or 18(16bit×8bit) MAC, or 36(8bit×8bit) MAC, therefore it supports multiple types of bit-width data. Architecture contains set of buffers (data, parameters) and registers (data and weight); to maximize data reuse and this make the parallel PEs operation more efficient. Compared to state-of-the-art designs, the proposed architecture achieves 2.36× energy efficiency, 2.4× to 6.8× area efficiency, and 16.3% to 27.4% higher computational efficiency for AlexNet benchmarked reference. The proposed architecture implements fast processing rate, small output latency, less circuit area, and low power consumption simultaneously.

References

[1]. Simonyan, K.; Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs].

[2]. Lee, Y.; Kim, H.; Park, E.; Yim, B.; Kim, H. (2016). Optimization for object detector using deep residual network on embedded board. In 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Seoul, South Korea, 26-28 Oct; 1–4.

[3]. Ma, H.; Mao, F.; Taylor, G. W. (2016). Theano-MPI: A Theano-Based Distributed Training Framework. In Euro-Par 2016: Parallel Processing Workshops, Lecture Notes in Computer Science, Grenoble, France, August 24-26; 800–813.

[4]. Gawande, N. A.; Landwehr, J. B.; Daily, J. A.; Tallent, N. R.; Vishnu, A.; Kerbyson, D. J. (2017). Scaling Deep Learning Workloads: NVIDIA DGX-1/Pascal and Intel Knights Landing. In 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Lake Buena Vista, FL, USA, 29 May-2 June; 399–408.

[5]. Shen, Y.; Ferdman, M.; Milder, P. (2017). Maximizing CNN Accelerator Efficiency Through Resource Partitioning. In Proceedings of the 44th Annual International Symposium on Computer Architecture; Toronto, ON, Canada, June 24 - 28; 535–547.

[6]. Gokhale, V.; Zaidy, A.; Chang, A. X. M.; Culurciello, E. (2017). Snowflake: An efficient hardware accelerator for convolutional neural networks. In 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28-31 May; 1–4.