High-performance Convolutional Neural Network Accelerator Based on Systolic Arrays and Quantization

Yufeng Li, Shengli Lu*, Jihe Luo, Wei Pang, Hao Liu
National ASIC System Engineering Research Center
Southeast University
Nanjing, China
e-mail: {lyf_seu, lsl, luojh, pw, nicky_lh}@seu.edu.cn

***Abstract*—In recent years, convolutional neural networks (CNN) has achieved great success in computer vision tasks, such as object detection, face recognition and image classification. With the diversification of CNN models, accelerators that only support a single network can no longer meet the needs of applications. Due to the computational intensiveness of convolution operations, the implementation of CNN on the FPGA platform faces many challenges. In this paper, a convolution unit based on systolic arrays is proposed in the design of CNN accelerator, and the fixed-point quantization method is adopted to save a large amount of storage resources and reduce the required transmission bandwidth, thus improving throughput and power efficiency. The performance density and power efficiency of our design can reach 0.165 GOPS/DSP and 36.3 GOPS/W under 100MHz clock frequency.**

Keywords-CNN accelerator; systolic arrays; quantization; high-performance; FPGA

I. INTRODUCTION

As an advanced deep learning method, convolutional neural network (CNN) is widely used in computer vision tasks. With the development of CNN, the network becomes deeper and deeper, leading to a sharp increase in parameters and computation. Recently, many accelerators have been proposed to accelerate the forward inference of CNN, and FPGA-based accelerators have become increasingly popular due to their reconfigurable ability and low power consumption.

There are plenty of researches on FPGA-based convolution accelerators. However, most of these accelerators are designed for a specific CNN. In this paper, a universal CNN accelerator is designed, which can perform a variety of operations, including convolution calculation, batch normalization [1], pooling, activation and quantization. Based on DoReFa-Net [2], 8bits fixed point quantization is used for features and weights, which improves the efficiency of storage and calculation.

When training the CNN model, the features after ReLU will be quantified into 8-bit numbers. After the model is updated, the weight value is quantified into 8-bit numbers for subsequent training. Although the quantization operation is not backpropagated, the next update of the model parameters is optimized toward the quantized model. After several iterations, we can finally get a convergent quantized model.

Although the data of each layer are very large, the power consumption of memory access can be reduced by improving the data reuse rate [3]. By increasing the parallelism of convolution unit and improving the reuse rate of input data, the performance of the accelerator can be improved. In this paper, a high-performance CNN accelerator is designed based on FPGA, which can realize hardware acceleration for most of the CNN models. The convolution unit is designed based on the systolic array [4], and the weights and activations is quantized. The main contributions of this work are summarized as follows.

(1) The proposed accelerator quantifies weights and activations to 8 bits simultaneously, which greatly reducing the requirement of storage resources and transmission bandwidth. Compared with the full-precision model (32-bit), the model using the quantization strategy uses only a quarter of the storage resources.

(2) A convolution unit based on systolic array is proposed, which increases the reuse rate of input data and improves the throughput of the accelerator tremendously.

(3) The performance density and power efficiency of the accelerator can reach 0.165 GOPS/DSP and 36.3 GOPS/W under 100MHz clock frequency, which is tested on Zynq-XC7Z035 FPGA using Tiny-YOLO model.

II. BACKGROUND

A. Convolutional Neural Network

Convolutional neural network (CNN) is a kind of deep artificial neural network, which usually includes convolution layer, pooling layer, activation layer and fully connection layer.

- The convolution layer is used to extract image features, and a convolution operation consists of a set of multiplication and addition of two three-dimensional matrices, the feature matrix and the weight matrix. Equation (1) shows the convolution operation, where fm_i^{in} and fm_j^{out} are the *i-th* input feature map and the *j-th* output feature map, is the number of input channels, wei_{ij} is the weight matrix of the convolution kernel, and $bias_j$ is the j-th bias.

$$fm_j^{out} = \sum_{i=0}^{n_{in}-1} fm_i^{in} \otimes wei_{ij} + bias_j \tag{1}$$

- The pooling layer reduces the amount of data and calculation by merging the information of adjacent data, and it only retains the main features. Pooling layer can reduce overfitting and improve the generalization ability. The maximum pooling retains only the maximum value of adjacent features, while the average pooling preserves the average of adjacent data.
- In the activation layer, nonlinear operation is performed on each feature to enable the CNN model to learn nonlinear mapping, thus enhancing the expression ability of the model. When the rectified linear unit (ReLU) is used as the activation function but the model does not work well, the leaky ReLU function can be considered as a substitute, which is likely to solve the problem.
- The fully connection layer integrates the learned features for the final prediction. For a classified network, the last layer of the network is generally the fully connection layer, and the output size is the number of categories, followed by softmax function to convert data into the probability value of each category.

B. CNN model Tiny-YOLO

Table I shows the architecture of Tiny-YOLO model which consists of 9 convolution layers and 6 max-pool layers. The input of the model is an image with a size of 416x416x3 and the output is a 125-dimensional vector. All the max-pool layers have the same size 2x2 and stride 2. The size of the last convolution layer is 1x1 and the stride is 1. The size of other convolution layers is 3x3, and the stride is 1. When the flag of BN is "1", the data after convolution will perform batch normalization. The data will be activated by using ReLU as the activation function when ReLU is "1".

TABLE I. TINY-YOLO MODEL

Type	Filters	Size/Stride	BN	ReLU
Convolutional	16	3 x 3 / 1	1	1
Maxpool	-	2 x 2 / 2	-	-
Convolutional	32	3 x 3 / 1	1	1
Maxpool	-	2 x 2 / 2	-	-
Convolutional	64	3 x 3 / 1	1	1
Maxpool	-	2 x 2 / 2	-	-
Convolutional	128	3 x 3 / 1	1	1
Maxpool	-	2 x 2 / 2	-	-
Convolutional	256	3 x 3 / 1	1	1
Maxpool	-	2 x 2 / 2	-	-
Convolutional	512	3 x 3 / 1	1	1
Maxpool	-	2 x 2 / 2	-	-
Convolutional	1024	3 x 3 / 1	1	1
Convolutional	1024	3 x 3 / 1	1	1
Convolutional	125	1 x 1 / 1	0	0

C. CNN Quantization Method

In order to compress the network model, decrease the memory size and access, and reduce the power consumption and improve the performance of the CNN accelerator, lots of CNN quantization methods have been studied.

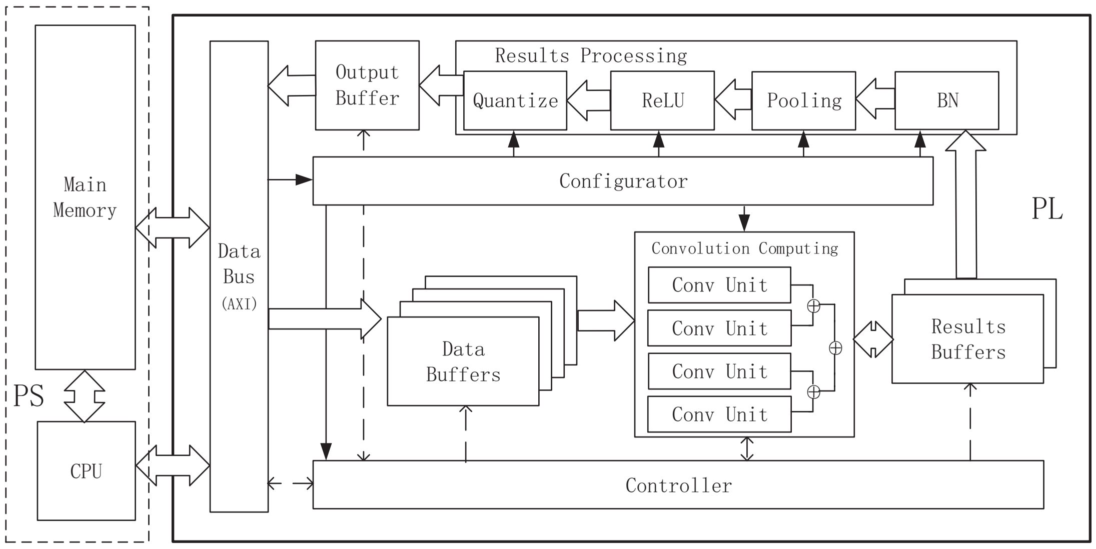

Figure 1. System Architecture

[5] introduces "BinaryConnect" method which can train a deep neural network (DNN) model with binary weights. By constraining the weights to only two possible values -1 or 1, and replacing multiplication and accumulation with simple accumulation, the accelerator performance can be greatly improved. [6] proposes XNOR-Networks, in which weights and inputs are binary. [7] designs ternary weight networks (TWNs), which constrains the weights to -1, 0 and +1. The performance of TWNs is only slightly worse than the full precision model, but much better than the binary weights model. In [2], "DoReFa-Net" method is proposed in which both the weights and activations are represented by low bit-width numbers. [8] proposes "deep compression", a three-stage pipeline work together to reduce the storage requirement, without loss of accuracy. In [9], Quantized Neural Networks (QNNs) were introduced, which use extremely low precision (e.g., 1-bit) weights and activations for forward inference and replaces most arithmetic operations with bit-wise operations.

III. CNN ACCELERATOR

A. *Overall Architecture*

As shown in Fig. 1, the whole system is divided into two parts: processing system (PS) and programmable logic (PL). The CPU on the PS side is responsible for initializing the PL, and configuring the working mode. CPU is also responsible for transporting the image data and weight data in the main memory to the PL side. The PL side includes the controller, configurator, convolution computing module, batch normalization module, pooling module, ReLU module, quantization module and so on.

- Configurator is used to configure the work mode of the accelerator for each layer of the CNN. Subsequent operations such as pooling, ReLU, batch normalization and quantization can be configured after convolution calculation.

- Controller is designed to manage the data flow inside the accelerator and divide it into four pipeline levels. The adjacent levels can be concurrently executed to reduce the waiting time. Data buffers, results Buffers and output Buffers constitute the main on-chip storage of the accelerator, which is responsible for storing the input data for convolution calculations, convolution results and output data. Data buffers and results buffers use ping-pong operations to send and receive data simultaneously.

- A convolution unit based on the systolic array is designed as the basis of the convolution computing module, and accelerated the convolution process by processing multiple convolutional calculation units in parallel. The convolution computing module can realize multiple convolution operations. Kernel size of the filter can be configured as 1x1,3x3,5x5,7x7, and the stride can be set to 1 or 2.

- Results processing module is designed to further process the data after the convolution to obtain the output feature maps. Batch normalization units perform batch normalization of convolution results. Pooling module can realize multiple pooling operations. Kernel size of pooling layers can be configured as 2*2 or 3*3, and the stride can be set to 1,2 or 3. ReLU unit activates the data to realize nonlinear operation. Quantization unit quantizes the activated features to 8bits to reduce the requirements for storage and transmission bandwidth.

B. *8bits Quantization*

The 8bits quantization is to quantize the full precision number (32bits) to 8-bit fixed point number [10]. It can greatly reduce the requirement for storage resources, improve the computational efficiency, and reduce the access frequency to memory, thus reducing the power consumption. Although numerical fixed-point quantization leads to loss of precision, the use of a quantization strategy can greatly improve the overall performance of the accelerator with a small loss of precision.

The precision loss of the value is only related to the number of decimal places before and after quantization, so the precision loss caused by the quantization can be reduced by maximizing the number of decimal places after quantization. For 8-bit quantization, the unsigned features after ReLU can be quantized into a number of 8-bit for decimal places, and the signed weights value is quantized into a number of 1-bit for sign and 7-bit for decimal places. And this strategy minimizes the loss of precision in 8bits quantization.

C. *Convolution Unit*

The convolution calculation is divided into rows, and each convolution unit completes the convolution operation on a row of data. The update of the feature data shown in Fig. 2 is done with the shift register. When the shift register is filled, every time a map data is moved, all PE can get a valid input data to realize the convolution calculation of adjacent positions. The registers RS_1 and RS_0 are not enabled when the stride of convolution is 1.

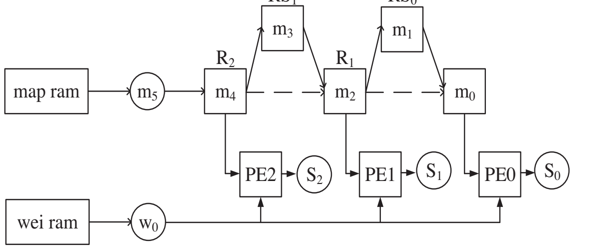

Figure 2. A Simplified Version of Convolution Unit

TABLE II. CONVOLUTION CALCULATION PROCESS (3*3/S2)

Clock Cycle	R_2	RS_1	R_1	RS_0	R_0	W	S_2	S_1	S_0	res_vld
0	m_0	-	-	-	-	-	-	-	-	0
1	m_1	m_0	-	-	-	-	-	-	-	0
2	m_2	m_1	m_0	-	-	-	-	-	-	0
3	m_3	m_2	m_1	m_0	-	-	-	-	-	0
4	m_4	m_3	m_2	m_1	m_0	w_0	$m_4 * w_0$	$m_2 * w_0$	$m_0 * w_0$	0
5	m_5	m_4	m_3	m_2	m_1	w_1	$S_2^{(4)} + m_5 * w_1$	$S_1^{(4)} + m_3 * w_1$	$S_0^{(4)} + m_1 * w_1$	0
6	m_6	m_5	m_4	m_3	m_2	w_2	$S_2^{(5)} + m_6 * w_2$	$S_1^{(5)} + m_4 * w_2$	$S_0^{(5)} + m_2 * w_2$	1

Table II shows the specific convolution calculation process when the size of the convolution kernel is 3*3 and the stride is 2. $S_1^{(5)}$ refers to the calculation result of the fifth clock cycle of PE1, the calculation result will be transmitted when res_vld is 1. Registers RS_1 and RS_0 ensure that PEs is always fully loaded. In order to compute a batch of valid output data, input features needs to be shifted twice. When the trailing data of a row is less than 3 bits, the first data of the next row is passed in for calculation, and invalid data is discarded in the output result, which can reduce the shift waiting time before the next line calculation starts.

After calculating a batch of valid data, the value in the shift register may not reach the starting position of the next convolution calculation, and it requires additional clock cycles for adjustment. To solve this problem, a double buffering strategy is employed for the shift register. At the beginning of a convolution calculation, both registers enr and eng are enabled. Register enr is used to pass feature data for convolution calculation, and register eng is used to cache the feature data needed for the next batch of convolution calculations. After the data reaches the correct location, register eng stops updating the data. When a batch of convolution calculations is completed, the roles of registers enr and eng are exchanged, eng is used for calculation, and enr is used to cache data.

The double-cached shift register uses the time of convolution computation to cache feature data for the next batch of convolution computation, which greatly reduces the waiting time and increases the throughput of the accelerator. In order to improve the performance of the accelerator, the feature data is broadcast to more PE units and convoluted in parallel with different convolution kernels. The convolution unit finally designed is shown in Fig. 3.

IV. EXPERIMENTAL RESULTS

The proposed CNN accelerator was implemented on the Xilinx Zynq-XC7Z035 FPGA. It makes full use of DSP resources on FPGA to improve the computing speed of CNN accelerator. Table III shows the resource utilization of the proposed accelerator.

TABLE III. FPGA RESOURCE UTILIZATION

	BRAM	DSP	FF	LUT
Used	312	758	91444	75241
Available	500	900	343800	171900
Utilization (%)	62.4%	84.2%	26.6%	43.8%

To evaluate the proposed accelerator, Tiny-YOLO model was tested on the Xilinx Zynq-XC7Z035 FPGA. The specific network structure of the Tiny-YOLO model is shown in Table I, which has a total of 6.6G multiply and accumulate operations. Table IV shows the results of the performance comparison between this accelerator and others. Under 100MHz clock frequency, the Performance Density and Power Efficiency of the proposed accelerator can reach 0.165GOPS/DSP and 36.3 GOPS/W, which are better than other accelerators.

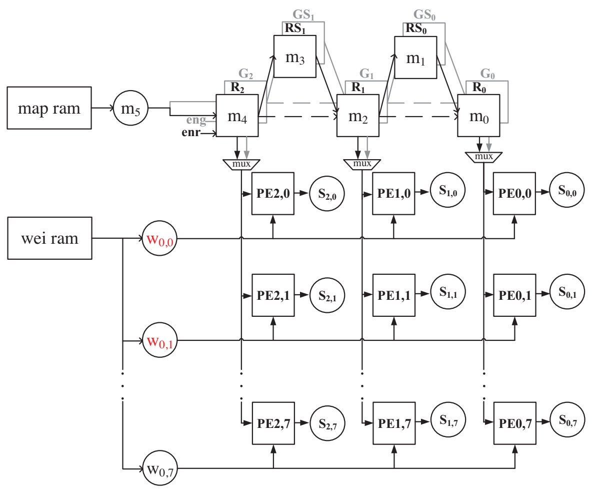

Figure 3. Convolution Unit Based on Systolic Array

TABLE IV. COMPARISON WITH OTHER WORKS

	[11]	**[12]**	**[13]**	**This work**
Platform	Zynq XC7Z045	Zynq XC7Z100	Virtex 690T	Zynq XC7Z035
CNN Size (GOP)	30.76	3.88	30.95	6.6
Throughput (GOPS)	137	19.8	488	125
Performance Density (GOPS/DSP @100MHz)	0.117	0.054	0.115	0.165
Power Efficiency (GOPS/W)	14.3	5.2	18.7	36.3

V. CONCLUSION

In this paper, a high-performance CNN accelerator is proposed which can accelerate multiple CNN models. The convolution unit is designed based on the systolic array, which improves the data reuse rate. By quantifying the feature data and weight data to 8bits, this accelerator reduces the requirement for storage resources and bandwidth, and improves the computational speed and performance. Tiny-YOLO model was used for the test on Zynq-XC7Z035 FPGA. The accelerator's performance density and power efficiency can reach 0.165 GOPS/DSP and 36.3 GOPS/W, which is better than many current accelerators.

REFERENCES

[1] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," ArXiv Prepr. ArXiv150203167, 2015.

[2] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," ArXiv Prepr. ArXiv160606160, 2016.

[3] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei, "Deep convolutional neural network architecture with reconfigurable computation patterns," IEEE Trans. Very Large Scale Integr. VLSI Syst., vol. 25, no. 8, pp. 2220–2233, 2017.

[4] X. Wei et al., "Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs," in Proceedings of the 54th Annual Design Automation Conference 2017, 2017, p. 29.

[5] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in Advances in neural information processing systems, 2015, pp. 3123–3131.

[6] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in European Conference on Computer Vision, 2016, pp. 525–542.

[7] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," ArXiv Prepr. ArXiv160504711, 2016.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," ArXiv Prepr. ArXiv151000149, 2015.

[9] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," J. Mach. Learn. Res., vol. 18, no. 1, pp. 6869–6898, 2017.

[10] M. Mathew, K. Desappan, P. Kumar Swami, and S. Nagori, "Sparse, quantized, full frame cnn for low power embedded devices," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 11–19.

[11] J. Qiu et al., "Going deeper with embedded fpga platform for convolutional neural network," in Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2016, pp. 26–35.

[12] Y. Yao et al., "A FPGA-based Hardware Accelerator for Multiple Convolutional Neural Networks," in 2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), 2018, pp. 1–3.

[13] C. Zhang, G. Sun, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 2018.