Hardware Implementation of Convolutional Neural Network for Face Feature Extraction

Ru Ding[1], Xuemei Tian[1], Guoqiang Bai[1], Guangda Su[2], Xingjun Wu[1]*
[1] Institute of Microelectronics, Tsinghua University, Beijing 100084, China
[2] Electronic engineering. Tsinghua University, Beijing 100084, China
* wuxj@tsinghua.edu.cn

Abstract

As an important feed-forward neural network in the field of deep learning, convolutional neural network (CNN) has been widely used in image classification, face recognition, natural language processing and document analysis in recent years. CNN has a large amount of data and many multiply and accumulate (MAC) operations. With the diversity of application files, the channel sizes and kernel sizes of CNN are diverse, while the existing hardware platform mostly adopts the average optimization technology, which causes the waste of computing resources. In this paper, a special configurable convolution computing array is designed, which contains 15 convolution units, each PE contains 6x6 MAC operations, it can be configured to calculate three different kernel sizes of 5x5, 3x3 and 1x1. At the same time, pipeline structure is used to synchronize convolution and pooling operations, which reduces the storage of intermediate results. We design the special hardware structure to optimize DeepID network. Tested on Altera Cyclone V FPGA, the peak performance of each convolution layer at 50 MHz is 27 GOPS, and the average utilization of the MAC is 92%.

1. Background

CNN is applied to many files such as intelligent security, smart home, smart transportation, and smart medical care. As tasks become more complex, CNN presents a trend of large-scale and diverse. This diversity and the amount of repeated data access make it difficult for existing accelerators to achieve efficient calculation. Literature [1] proposes a CNN accelerator which can provide not only multiple clustering methods for brain-like neurons and link organization among brain-like neurons towards different channel sizes, but also three mapping methods for different convolution kernel sizes, but this structure uses a lot of on-chip storage. In [2], the weights are shared among multiple neurons, and the feature map data is moved on the computational array to achieve multiplexing. However, this structure has a large amount of repeated access to the feature map for multi-channels. This paper proposes a dedicated configurable convolution calculation array for different kernel sizes and multi-channels. The rest of this paper is organized as follows: section 2 describes the architecture of CNN, section 3 introduces the optimization technology of our design, the system structure of design is described in section 4, in section 5, the details of configurable convolutional computing array are presented. Finally, the experimental results are summarized and analyzed.

2. Network structure

Convolutional neural network generally includes convolution layer, pooling layer and full connected layer. Generally, three-dimensional convolution is used in convolution layer, as shown in formula 1.

$$O_{k_i} = f(W_{k_i} \otimes I_{k_i} + B_{k_i}) \qquad (1)$$

The convolution kernel (W) and the input feature map (I) are used to perform a three-dimensional convolution operation. Bias (B) and activation function (f) are added to generate the output feature map (O). Activation functions include ReLu function, sigmoid function and tanh function. We optimize the hardware structure based on a small deepID network, which consists of four layers of convolution (C1, C3, C5, C7), three layers of pooling (S2, S4, S6) and a full connected layer (F8). ReLu function is followed at each convolution layer [3]. The network structure is shown as Figure 1.

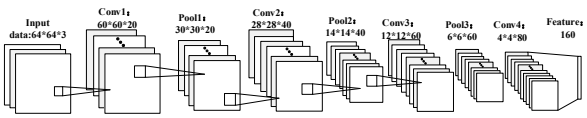

Figure 1. DeepID network structure

The C1 layer, of which the input is 64x64 RGB image and the outputs are 20 feature maps of size 60x60, contains 20 3-channels convolution kernels of size 5x5, max pooling layer is employed after convolution and the stride is 2. The C3 layer contains 40 20-channels convolution kernels of size 3x3 and outputs 40 feature maps of size 28x28. The outputs of S4 layer are 40 feature maps of size 14x14. The C5 layer contains 60 40-channels convolution kernels of size 3x3 and outputs 60 feature maps of size 12x12. The output of S6 layer is 60 feature maps of size 6x6. The C7 layer contains 80 60-channels convolution kernels of size 3x3 and outputs 60 feature maps of size 6x6. F8 is the full connection layer, of which the input is a feature map of 1x1280 and the output is 160 feature points.

3. Hardware optimization technology of CNN

At present, the hardware optimization techniques commonly used in CNN include parallelism, data reuse

978-1-7281-0735-6/19/$31.00 ©2019 IEEE

and memory optimization [4].

There are three parallel modes of convolution layer in our design: (1) parallelism between different channels, because the data of different channels are independent, so it is available to read the feature map data and the convolution kernel data in parallel between different channels, this method can copy multiple identical arithmetic units to achieve parallelism; (2) parallelism between different convolution kernels, since the data of different convolution kernels are independent, the feature map can be shared by multiple convolution kernels; (3) parallel convolution kernels, parallel operations can be used in a single convolution kernel, during one clock cycle multiple MAC operations are performed[5].

Data multiplexing usually has inter-layer data multiplexing and intra-layer data multiplexing. Intra-layer data multiplexing refers to increasing parallelism and increasing data multiplexing when performing layer-by-layer computing. Inter-layer data multiplexing refers to reducing off-chip access. Save the number of times to minimize the intermediate results stored in the off-chip storage block [6].

Memory optimization mainly includes reducing data accuracy, storage block fragmentation and other methods. Research shows that fixed-point numbers on hardware platform can reduce storage space and improve computing performance and throughput. In this paper, 16-bit fixed point is used to ensure the accuracy loss within an acceptable range. Through analysis, different fixed-point representation methods for different convolution layers can reduce the accuracy loss to a certain extent [7].

There are two common hardware architectures of CNN. One is the layer serial structure, all PE units are only used to realize the function of one layer, and the whole network operation is completed layer by layer by time division multiplexer technology (TDM). This structure only considers the single-tier hardware architecture, the design is relatively simple, and the data control is easy. However, this structure also has many shortcomings: on the one hand, the parameters of each layer need to be updated; on the other hand, due to the uneven distribution of calculation in per layer, hardware resources may not be fully used, which will result in the waste of resources. To solve this problem, we design a configurable convolution array, which can maximize the use of DSP resources in each layer by using certain parallelism and reconfigurable technology [8].

The other is layer parallel structure, PE cells are divided into several groups according to the calculation amount of each layer, each group realizes the calculation of one layer. Hardware resources can be maximized, but it is not flexible. When the network structure changes, the whole hardware structure must be redesigned, the workload is heavy [9].

4. System structure

We adopt the layer serial structure with high flexibility and relatively low difficulty. The dedicated hardware implementation platform is mainly used for feature extraction of human face. The image data is derived from camera acquisition, and the weights are stored in DDR3. The on-chip memory communicates with the off-chip DDR3 via the Avalon bus. The on-chip hardware structure includes an input feature map buffer, a weight buffer, an output feature map buffer, an overall control module, a convolution data control module, a convolution weight control module, a convolution computing array composed by 15 convolution operation units, an accumulator unit and a pooling unit [10]. The systemstructure is showas Figure 2.

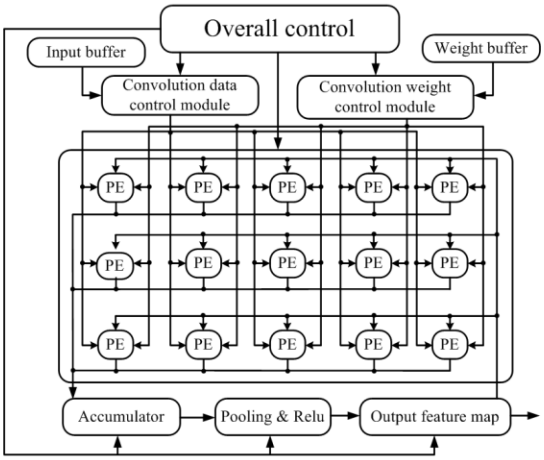

Figure 2. systemstructure

The overall control module is responsible for the control of the entire computing process of CNN, including reading data and weights from off-chip to the on-chip input buffer, giving the select signal to each module during the operation. The convolution data control module is used to allocate input data to the convolution calculation array. Since input image is 3-channels, which is different from 60-channels of other layers, so this module is mainly used to process data distribution of input images.

The convolution weight control module mainly controls reading weights from weight buffer. Because the weights of all layers are stored by 60-channels in weight buffer, we need to control the order in which the data is read. For example, there are 20 3-channels kernels of C1 layer, but they are expanded to 60-channels when stored. When performs calculation of one channel, we should read the kernel data of the corresponding channel.

The input feature map buffer is used to store the input feature maps, the convolution kernels are stored in weight buffer, and the intermediate results are stored in the output feature map buffer. 15 convolution

computational units (PEs) are used for calculations, each containing 6x6 MAC operations, a convolution weight allocation unit, and multiple convolution data allocation units.

There are two units of the accumulation operation module, one is an accumulator controller and other is an accumulator unit. The function of the accumulation operation module is to accumulate calculation results of different channels. The pooling unit performs the maximum pooling operation, and the pooling time with the activation function operation time are hidden in the convolution calculation time.

4.1 The overall Controller

The overall control module, which is primarily a state machine, controls and schedules the operation of other functional modules. After the system is powered on, the overall controller will load image data, weights from the off-chip DDR3 into the input feature map buffer and weight buffer. If the network parameters are small, they can also be cached on the chip. After the data and parameters are loaded, the state machine jumps to conv state and start calculation. At this state control signal will give to other modules. In our design, the pooling layer, the activation function layer and the convolution layer are made into a full-flow structure. The middle results are stored in the on-chip output feature map buffer, and the calculation data of the next layer is loaded from this buffer and broadcast to each convolution module until the network calculation is complete.

4.2 On-chip buffer

The on-chip buffer is divided into three parts: input feature map buffer, convolution weight buffer and output feature map buffer. A special data storage method is designed to make full use of bandwidth and parallelism and improve the overall computing performance [11].

Usually, the matrix is stored in the memory bank by row. In our design, the feature map and weight data are stored by channel. For the input three-channel RGB image, it is stored in different BRAMs according to channel classification [12]. To maximize the use of hardware resources, the number of channels of the weight buffer is set to 60, if the channels of kernel are less than 60, we unroll the channels of parallel convolution kernels or make up 0. The output feature map buffer is also stored by 60-channels and if the output result matrix is less than 60 channels, add 0 or unroll loop.

4.3 Convolution Computing Array

There are 15 convolution computing units composing a convolution computing array. Each PE can be configured to calculate a 5x5 convolution kernel and 4 3x3 convolution kernels and 36 1x1 convolution kernels. The composition of the PE includes a convolution weight allocation module, a plurality of convolution data configuration modules, and 6x6 multipliers. The hardware structure of the convolution computing unit is shown as Figure 3.

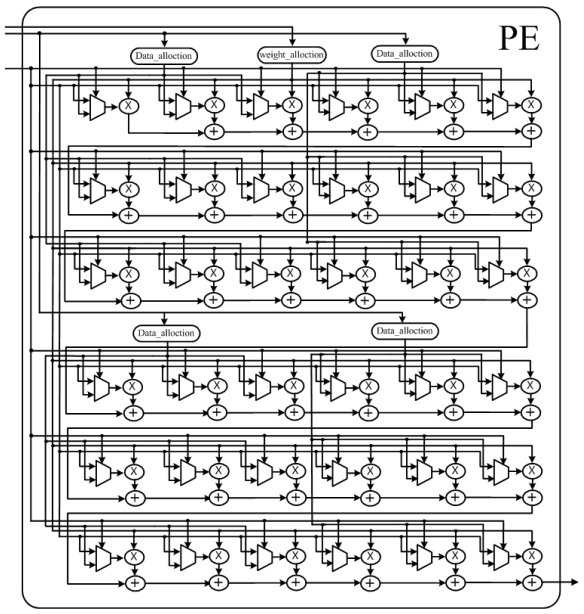

Figure 3. convolutional computing unit

In our design, there are 3x3 convolution kernels except C1 layer, so four convolution data configuration modules are designed. After C1 layer is done, the overall controller gives a signal and controls convolution data configuration module to read the data from the output feature map buffer and allocate to each multiplier to perform the convolution operation. Convolution kernel intra-parallelism is used in a single PE to maximize the utilization of hardware resources. In our design, 540 multiplication operations can be calculated in parallel at one clock cycle.

The convolution weight allocation module mainly needs to block the weights that loaded from the convolution weight control module when performing multi-size convolution kernel operations, and then allocates them to each multiplier to complete the convolution. After C1 layer is completed, the convolution kernel allocation module divides the 36 convolution kernel parameters into 4 blocks, and each block calculates 3x3 convolution, so each PE can calculate 4 kernels of 3x3.

The convolution calculation of the C1 layer can adopt a parallel strategy of 3 channels, 5 kernels of size 5x5. Each PE calculates 6x6 multiply accumulates, so at one clock cycle 5x5 input data is ready and the other is set to 0, the utilization of the MAC is 70%. Except for C1 layer, the concurrency of remaining layers is 60, and the kernel size is 3x3. One PE can calculate 4-channels feature map, the calculation of each channel is 3x3, so the utilization of the MAC is 100%, which maximizes hardware

resources.

4.4 Accumulating operation module

The accumulation unit is mainly used to accumulate the values of the convolution calculation array to obtain the final result of each convolution layer. Because the channels of each layer are different, the accumulation method is different. The accumulator is composed of a control unit and an accumulating unit, and the control unit receives select signal from the overall controller to control which results of the accumulating unit should be output. The accumulating unit performs three different accumulating operations because of different parallelism. The structure is shown as Figure 4.

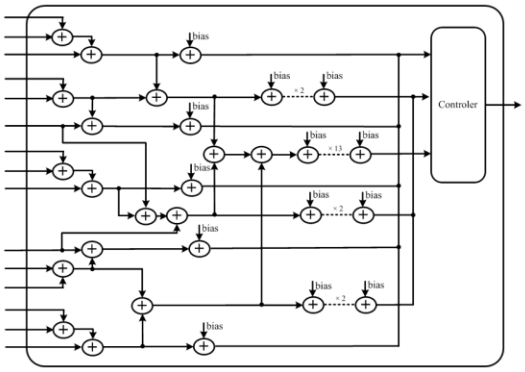

Figure 4. accumulation unit

5. Experimental results and analysis

This design uses Verilog programming, Quartus II software as a development tool for simulation, synthesis and implementation, and modelsim software for functional simulation. FPGA is used as the hardware verification and implementation platform. The design uses Altera Cyclone V 5CGTFD9E5F35C7 series FPGA chip to verify the function.

Table 1. source usage rate

Source type	used	Available
Registers	57589	454,240
ALMs	51880	113560
I/O	0	616
Block memory bits	514048	12492800
DSP	270	342

The input of the convolution computing array is a 3-channels RGB image and 20 convolution kernels of size 5x5. The system running clock is 50MHz. The 5 convolution kernels are performed in parallel under one clock cycle. The output feature map of 20-channels is obtained by 4 loops and stored in the on-chip output buffer. The result is compared with the results of MATLAB. It is proved that the data precision of 16-bit fixed point number can meet the requirements, precision loss does not exceed 1%. The resource consumption given by the FPGA is shown as Table 1.

6.Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grants 61472208).

7.References

[1] QIAQ Ruixiu, CHEN Gang, GONG Guoliang, LU Huaxiang. Journal of xidian university, 46(3) (2019).

[2] Du Z, Fashuber R, Chen T, et al. ACM/IEEE International Symposium on Computer Architecture, 43(3),92-104 (2015).

[3] Sun Y, Wang X, Tang X. IEEE Conference on Computer Vision & Pattern Recognition(CVPR), arXiv.1406.4773 (2014).

[4] Sze V, Chen Y H, Yang T J, et al. Proceedings of the IEEE, 105(12):2295-2329 (2017).

[5] Zhiwei Li, Yan Li, Song Chen, Feng Wu. IEEE International Conference on ASIC (ASICON) (2017).

[6] Weijia Chen, Hui Wu, Shaojun Wei, Anping He, Hong Chen. IEEE Asian Solid-State Circuits Conference (A-SSCC),8064 (2018).

[7] Solovyev,R. A., Kalinin, A. A., Kustov, A. G, Telpukhov, D. V., & Ruhlov,V. S. . IEEE Conference on Computer Vision & Pattern Recognition(CVPR), arXiv.1810.09945 (2018).

[8] Yu J, Hu Y, Ning X, et al. International Conference on Field Programmable Technology (ICFPT),22 (2017).

[9] Li H, Fan X, Jiao L, et al. International Conference on Field Programmable Logic and Applications (FPL). IEEE (2016).

[10] Qiu J, Song S, Wang Y, et al. ACM/SIGDA International Symposium, International Symposium on Field-Programmable Gate Arrays (FPGA), 26-35 (2016).

[11] Chao Huang, Siyu Ni, Gengsheng Chen. IEEE 12th International Conference on ASIC (ASICON) (2017).

[12] Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., & Cong, J. . Acm/sigda International Symposium on Field-programmable Gate Arrays. ACM (2015).