An Efficient OpenCL-Based FPGA Accelerator for MobileNet

Liu Wei¹* and Lv Peng²

¹ School of Computer Engineering and Science, Shanghai University, Shanghai, 200444, China
² School of Computer Engineering and Science, Shanghai University, Shanghai, 200444, China

*Corresponding author’s e-mail: lokyi@shu.edu.cn

Abstract. Convolutional neural network is widely used in image processing and image recognition. In order to obtain higher accuracy, the computational complexity and the scale of model and parameters are increasing. FPGA has become a good choice because of its low power consumption and high flexibility. MobileNet replaces the standard convolution by depthwise convolution and pointwise convolution, which greatly reduces the computational complexity and parameters of the model in the case of less precision loss, so that it can be applied to the equipment with limited computing resources. In this paper, we propose an efficient OpenCL-based FPGA CNN accelerator to realize inferencing acceleration of MobileNet. We designed the convolution layer with modularization. We used pipeline to design a depth separable convolution parallel acceleration scheme, and made full use of the DSP resources of FPGA. This design finally achieved a good balance of hardware resources, processing speed and power consumption. The experiments show that the accelerator can reach the inferencing speed of 32.56ms, and the energy consumption is 20W. The speedup is 4x compared with CPU and energy efficiency is 3x compared with GPU.

1. Introduction

In recent years, with the expansion of dataset and the innovation of neural network structure, neural network has achieved higher accuracy and performance [1][2]. However, in order to achieve higher accuracy, the neural network model becomes more and more complex, up to hundreds of layers; the amount of data used for training is also more and more huge, reaching billions (10gflops) [3]. In practical use, efficiency and cost are also factors that must be considered, so the requirements for the computing power and energy consumption of neural network are becoming higher and higher [4][5].

FPGA has the characteristics of high flexibility, programmability and low power consumption. It is usually used to replace GPU as an efficient accelerator for deep neural network. [6] implemented the corresponding acceleration based on OpenCL by using efficient architecture and pipelined kernels to implement large CNN networks. [7] LACS reduces the negative impact on computing efficiency caused by invalid operations. Similar to the CPU computing mode, it reduces the amount of data and movement between the register and in-chip buffer BRAM. However, the size and number of large CNNs are still a factor influencing the inferencing speed. Researchers have also started to study the acceleration of small neural networks such as MobileNet [8], ShuffleNet [9], GhostNet [10] on the FPGA. [11] proposed redundancy reduced MobileNet, which saves the parameters in FPGA on-chip memory, the parameters are reduced and the accuracy is improved. [12] designed a depth separable convolution framework that can automatically generate and optimize CNN hardware design through depthwise separable
convolution (DSC). Compared with vgg-16 for ImageNet, the speed is increased by 3.43 times. [13] proposed a scalable high performance DSC optimized CNN accelerator. They processed the computing tasks of different layers in one unit, used hierarchical storage to reduce the on-chip memory limit.

However, using hardware description language to develop FPGA accelerator is very complex. Programmers can use high-level language to develop without understanding hardware circuit. We designed a convolution layer with modularization, designed configurable kernel programs to speed up the different layers. We designed the convolution layer with modularization, and completed the modular implementation of depthwise convolution (DC) and pointwise convolution (PC). We used pipeline to design a depth separable convolution parallel acceleration scheme, and made full use of the DSP resources of FPGA. This design finally achieved a good balance of hardware resources, processing speed and power consumption. In the final experiment, we classified the pictures in ImageNet on Inspur_F10A. The FPGA design achieves 4x speedup if compared to CPU and 3x less energy consumption than GPU.

2. MobileNet and OpenCL

2.1. MobileNet model architecture

MobileNet is a lightweight CNN, which reduces the size and number of models, while ensuring the accuracy, so that it can be implemented under resource-constrained conditions. The main difference between MobileNet and VGG is that the standard convolution (SC) is replaced by the DSC. As shown in Figure 2, MobileNet decomposes a 3*3 SC layer into 3*3 DC and 1*1 PC, with BN batch normalization and ReLU activation processing behind each section [14].

![Figure 1. Depthwise Convolution and Pointwise Convolution.](image1)

![Figure 2. Standard Convolution structure and Depthwise Separable Convolution structure.](image2)

2.2. MobileNet Performance Analysis

For standard convolution layers, there are \( N \) convolution cores of size \( D_k \times D_k \times M \), and the number of operations for each convolution core is \( WHDDM \times DD \times N \times D_k \times D_k \times D_k \) . The computation of standard convolution is:

\[
D_k \times D_k \times M \times N \times D_k \times D_k \times D_k
\]

(1)

The computation of DSC includes two parts: the computation of DC and PC. The convolution kernel size of depth convolution kernel is \( D_k \times D_k \times M \), each convolution kernel needs \( D_k \times D_k \times M \) times multiplication and addition; PC has \( N \) convolution kernels of \( 1 \times 1 \times M \), and each convolution kernel needs \( D_k \times D_k \times M \times N \times D_k \times D_k \times D_k \) times multiplication and addition. Therefore, the computation for DSC is as follows:

\[
D_k \times D_k \times \alpha M \times \beta D_k \times \beta D_k + \alpha N \times \alpha M \times \beta D_k \times \beta D_k
\]

(2)

Compared with the standard convolution and the depth separable convolution, the results are as follows:

\[
\frac{D_k \times D_k \times \alpha M \times \beta D_k \times \beta D_k + \alpha M \times \alpha N \times \beta D_k \times \beta D_k}{D_k \times D_k \times \alpha M \times \alpha N \times \beta D_k \times \beta D_k} = \frac{1}{N} + \frac{1}{D_k^2}
\]

(3)
It can be seen that by using depth separable convolution, the number of parameters and computation of MobileNet can be reduced to 1/8 to 1/9 of the standard convolution (when using 3*3 convolution kernel).

2.3. OpenCL
OpenCL is a standard API and programming language for parallel operations on heterogeneous computing devices. It is a heterogeneous computing language based on C language and can run on accelerator such as CPU, GPU and FPGA. Compared with HDL hardware description language (Verilog, VHDL) FPGA design process, OpenCL programmers can develop high-performance applications in a higher level to design abstraction environment without understanding the hardware circuit details, which can make it put on the market timely. Figure 3 shows the development process on FPGA using OpenCL. In the process of FPGA development, it is divided into host program and kernel program. Kernel programs need to take full account of the parallel computing power of the FPGA. The computing task is written as a module into the kernel, which will be compiled as hardware logic circuits on the development board of the FPGA. The host program runs on the CPU and is responsible for assigning different kernel tasks to the FPGA according to the algorithm logic that is written.

3. Accelerator design
We use OpenCL to design mobile net based on FPGA, which is composed of a series of OpenCL kernel programs. According to the structure of MobileNet, we divide all the kernel modules into two types: Conv3D and Conv2D. Conv3D includes SC, PC and full join layer operation; Conv2D includes DC and ReLU. In the process of operation, we adjust the data to be transmitted, and use channel pipe to realize data transmission between the kernels. Through the reuse analysis of parameters and weights in DSC, we share memory data to reduce the transmission between on-chip storage and external storage, so as to reduce the time of memory access and data exchange and improve the efficiency of memory utilization.

3.1. Kernels design
3.1.1. Conv2D. In this group of kernels, the kernel performs DC (2D) and average pooled ReLU operations on the data in memory according to the configuration of the host program. The DC splits the convolution kernel of SC into single channel mode. We do not change the depth of the input feature image, but we do convolution for each channel, and then we can achieve the same number of channels of output feature image and input feature image. Because the dimension and channel number of the
output feature map after DC are less than that of the standard convolution, the output feature information in the volume and operation will be less, so we need to use PC to increase and reduce the dimension of the feature map after DC, so as to obtain the same feature information as the SC.

We import the input feature image data from memory into the DC kernel, the output of each channel is obtained by multiplying and accumulating the input features and the same number of weights in the feature graph. In the process of DC, each different channel performs 2D calculation on the input image, and the calculation of each channel has no data exchange. We parallelize the convolution process of different channels to increase the utilization of FPGA on-chip resources and improve the computational efficiency. The figure shows the architecture of Conv2D kernel group.

![Figure 4. The Architecture of Conv2D kernel group.](image)

### 3.1.2 Conv3D

Different input channels are used to calculate the data of different layers, and different computing tasks are configured through the CPU. The pointwise convolution has high parallelism in convolution. Each different output channel and each pixel operation are independent. We make the operation of PC highly parallel, so as to speed up the model inferencing. In the kernel program of PC, we adjust the parallelism of PC by adjusting the number of convolution kernels that can be calculated at the same time. By using the loop expansion method, #pragma unroll in OpenCL, we optimize the original six layers 3D convolution kernel and change it into two layers, which reduces the number of loops and increases the efficiency of operation. The pseudo code of the kernel module is as follows:

**Algorithm 1 Conv module operation pseudo code**

| Input: | conv_input_data, conv_input_weight |
| Output: | conv_out_data |
| #pragma unroll | shiftReg[0] = mult |
| for each pipeline do: | end for |
| for each conv_out_data do: | sum = Σ shiftReg[N] |
| shiftReg[N] = 0 | end for |
| for each vec_size do: | data = sum + bias |
| mult = conv_input_data[i] * conv_input_weight[i] + shiftReg[N-1] | conv_out_data = data > 0? data: 0 //ReLU |
| shiftReg[N] = shiftReg[N-1] | end for |

The calculation logic of standard convolution and pointwise convolution is similar. Setting the channel number of pointwise convolution to $K$ is standard convolution. The same computing logic applies to the fully connected layer in this kind of kernel. After decomposing the weight matrix, we can convert the calculation of this layer into matrix multiplication and output the results separately. In the Conv3d kernel group, all three kernels can form the addition operation of one-dimensional vectors after logical decomposition. By adjusting the size of these one-dimensional vectors and the frequency of iteration, we can reuse the resources in memory, reduce data exchange, and increase the efficiency of model inferencing by increasing the resource utilization of the FPGA.
3.2. Data transfer design
In the process of inferencing on the FPGA, the in-chip memory will exchange data with the external memory several times, while frequent data exchange will affect the speed and cause a lot of energy consumption. We redesigned the kernels of read and write operations by adding data reuse and adjusting the logic of data access to achieve high-efficiency data transmission. We adjust the data structure that needs to be sent from the host to the device for operation, and use channel pipes for transmission, reducing the frequency of data transmission. FPGA has limited memory, so it needs to use external memory to save and exchange a large number of parameters. When the convolution Stride and the edge length of the convolution kernel $D_k$ are in the convolution operation, the convolution kernel will slide on the input feature map. There are $(D_k - S) \times D_k$ parameters that can be reused, and we can store them as buffers to reuse data. The weight data of each layer in Conv2D is $D_k \times N$. The memory of the FPGA can be stored completely. We can convolute in different channels and extract the output of parameter calculation. Conv3D has a very large number of weight parameters that are reused. Three dimensional convolution has more channels than 2D convolution and a large number of parameters. We use buffered weight stream to batch import data, and make full use of each read data. Experimental results

3.3. Experiment setup
We use Inspur_F10A FPGA development board to complete the design. We use the Intel OpenCL Pro 19.1 SDK on the development board. Table 1 shows the hardware information.

| Device               | Description                                      |
|----------------------|--------------------------------------------------|
| Arria10 GX1150       | 16G DDR3; 1,150K LEs; 1,518 DSPs                 |
| Xeon CPU Gold 5115   | 10 cores; 2.4GHz                                 |
| NVIDIA 2080Ti        | 11G GDDR6; 4352 CUDA cores                       |

3.4. Resource utilization analysis
The design is experimented with in Inspur_F10A platform, after summarizing the resources occupied by each module, the final resource usage is shown in Table 2.

|                        | FFs    | ALMs   | RAMs   | DSPs  |
|------------------------|--------|--------|--------|-------|
| FPGA resource          | 1708800| 854400 | 2713   | 1518  |
| Number of uses         | 546816 | 410112 | 1411   | 668   |
| Usage rate             | 32%    | 48%    | 52%    | 44%   |

The same MobileNet neural network model is implemented on CPU and GPU respectively to compare with the accelerator we designed for FPGA. We do inferencing on CPU, GPU and FPGA respectively, and compare the acceleration effect and power consumption in different ways. As shown in Table 3, the order of speed from fast to slow is CPU (1x), FPGA (4x), GPU (13x). The speed of neural network running on the FPGA is better than that on the CPU, GPU consumes 3x as much energy to process a single image as the field programmer. Combining acceleration efficiency and energy consumption, the FPGA accelerator has better performance and lower energy consumption.
Table 3. System resource statistics.

| Device          | CPU          | GPU          | FPGA         |
|-----------------|--------------|--------------|--------------|
|                 | Xeon CPU gold 5115 | Nvidia 2080Ti | Inspur_F10A  |
| Delay (ms)      | 96.68        | 8.06         | 32.56        |
| Power (W)       | /            | 225W         | 20           |
| J/per Image     | /            | 2.4          | 0.79         |
| Speedup         | 1x           | 13x          | 4x           |
| Energy Efficiency | /            | 1x           | 3x           |

4. Conclusion
In this paper, MobileNet is studied and an efficient lightweight CNN accelerator is proposed. MobileNet is implemented for inferencing acceleration based on the OpenCL on the FPGA. We implement the depthwise separable convolution module and design a pipeline parallel acceleration scheme. Our experimental results show that the accelerator achieves 4x of speedup compared with CPU and 3x of energy efficiency and compared with GPU in the experiment of image classification in ImageNet. It makes full use of the DSP resources available in FPGA board, achieves a better balance in hardware resources, speed and power consumption, and fully demonstrates the performance and energy consumption advantages of FPGA in the implementation of lightweight neural network.

Acknowledgments
This research was funded by Science and technology commission of Shanghai Municipality under grant No. 19511121002.

References
[1] He, K., Zhang, X., Ren, S., Sun, J. (2015) Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.
[2] Girshick, R., Donahue, J., Darrell, T., Malik, J. (2013) Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.
[3] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S. (2015) ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3):211-252.
[4] He, K., Zhang, X., Ren, S., Sun, J. (2016) Deep residual learning for image recognition.
[5] Simonyan, K., Zisserman, A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. Computer Science.
[6] Wang, D., Xu, K., Jiang, D. (2017) PipeCNN: An OpenCL-based open-source FPGA accelerator for convolution neural networks. 2017 International Conference on Field Programmable Technology (ICFPT).
[7] Shang, J., Qian L., Zhang, Z., Xue L., Liu, H. (2020) LACS: A High-Computational-Efficiency Accelerator for CNNs. In: IEEE Access, vol. 8, pp. 6045-6059.
[8] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M. (2019) Searching for MobilenetV3.
[9] Zhang, X., Zhou, X., Lin, M., Sun, J. (2017) ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.
[10] Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C. (2020) GhostNet: More Features From Cheap Operations. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, pp. 1577-1586.
[11] Su, J., Faraone, J., Liu, J., Zhao, Y., Cheung, P.Y.K. (2018) Redundancy-Reduced MobileNet Acceleration on Reconfigurable Logic for ImageNet Classification. In: ARC 2018. pp. 16-28.
[12] Zhao, R., Niu, X., Luk, W. (2018) Automatic Optimising CNN with Depthwise Separable Convolution on FPGA: (Abstract Only).
[13] Bai, L., Zhao, Y., Huang, X. (2018) A CNN Accelerator on FPGA Using Depthwise Separable Convolution. IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, no. 10, pp. 1415-1419.

[14] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T. (2017) MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.