High Throughput CNN Accelerator Design Based on FPGA

Liang Xie, Xitian Fan, Wei Cao*, Lingli Wang
State Key Laboratory of ASIC an System
Fudan University, Shanghai, China
*caow@fudan.edu.cn

Abstract—Due to the fact that FPGA on-chip memory capacity increases significantly, the feature maps and weights of convolutional layers can be stored on chip, which can reduce the data movement between on-chip memory and off-chip memory. Hence, the bottleneck can shift from the bandwidth to the computing resources in convolutional layers, which will improve the performance dramatically. Under this circumstance, this paper quantitatively analyze how to design the hardware architecture based on the roofline model to optimize the performance under the constraints of available on-chip computing resources and propose an efficient architecture. Our accelerator is implemented on Xilinx UltraScale+ FPGA with the performance of 9.39 TOPS and 6.86 TOPS for 8-bit data width with 100MHz main frequency and 400MHz DSP frequency on ResNet-50 and AlexNet, which outperforms the existing FPGA-based CNN accelerators.

Index Terms—Convolutional Neural Networks; High Performance and Throughput; FPGA Accelerator; Design Methodology.

I. INTRODUCTION

In the past few years, Convolutional neural networks (CNNs) have shown very good performance in solving many problems such as visual recognition, speech recognition and natural language processing. Since AlexNet won the ImageNet contest and led the application boom of neuron networks in 2012, CNN networks become the first choice for image classification. Its rapid development has benefited from the designs of different architectures such as [1],[2], [3], [4], [5], and [6]. With the inference accuracy of CNNs continuously improving, the networks have been deepen and expanded which brings massive computing demand. Using FPGA as hardware accelerator for CNNs thus becomes a popular choice. However, FPGAs cannot store large amounts of data due to the limitation of on-chip memory resources, which limits the FPGA accelerator performance by the bandwidth bottleneck.

With the continuous development of FPGA devices, the on-chip storage capacity has increased significantly. For example, the storage capacity of Xilinx UltraScale+ architecture has increased to nearly 350 Mb, which makes it possible for the CNN feature maps to be stored on chip instead of being exchanged between the device and the external memory. Under this circumstance, the burden of the bandwidth can be released. It means the bottleneck can shift from the bandwidth to on-chip computing resources. Therefore, we propose an accelerator design method that can fully utilize on-chip computing and storage resources to improve the performance.

The key contributions of this paper are listed as follows.

- This paper explores the design space of CNN hardware accelerator architecture with the significant development of on-chip memory capacity based on the roofline model.
- We propose an architecture for CNN accelerator to improve the utilization of on-chip resources and keep the DSP efficiency at a high level for most of the convolutional layers.
- We implement our CNN accelerator on Xilinx UltraScale+ FPGA with Vivado 2018.2 and achieve the performance of 9.39 TOPS and 6.64 TOPS with 100MHz main frequency and 400MHz DSP frequency on ResNet-50 and AlexNet, which outperforms the existing accelerators.

II. BACKGROUND

A. Convolutional Neuron Networks

CNNs have three main characteristics. One is the partial receptive field, another is the weight sharing, and the third one is the downsampling. For images or voice information, the local correlation is strong while the correlation among relatively distant information is weak. Through local perception, primary features of partial receptive fields can be extracted and integrated in the next layer to obtain global information. Weight sharing can greatly reduce the amount of weight data and make the extracted features become invariant to translation, scaling and rotation. In order to reduce the amount of computation and prevent over-fitting, downsampling is performed after the features are extracted. Downsampling divides the input matrix into non-overlapping submatrices and compute the average or maximum value. Pooling layers further reduce the computational complexity and decrease the sensitivity to rotation and distortion, which greatly enhances the robustness of the model.

B. Roofline Model

Roofline model provide an easy-to-understand, visual performance model that offers insights to programmers and architects on improving parallel software and hardware for floating point computations [7]. It uses the operational intensity($OI$) which means operations per byte of DRAM traffic to present the relationship between attainable performance(Perf) and bandwidth($BW$). There are two kinds of roofs that limit the performance. Under the bandwidth roof, though the platforms have much computing power, they can’t get enough data from off-chip memory to reach the peak performance. Likewise, higher bandwidth can no longer increase the performance under the computation roof either.
III. DESIGN SPACE EXPLORATION

A. CNN accelerator analysis

The performance of the CNN accelerators is mainly determined by the computing resources, memory, external bandwidth, and the utilization of processing element (PE) array. The computational resources determine the upper limit of the FPGA accelerator performance. In order to approach this upper limit, the key is to keep more PEs at work which is mainly influenced by two aspects. The first one is the utilization of PE array for the whole networks, which is related to the hardware architecture and the mapping strategy. The problem is that the utilization may not always maintain at a high level when mapping different networks or different layers in the same network. The other aspect is to meet the data transmission requirement of accelerator which means plenty of on-chip memory or bandwidth should be available. Under the circumstance of limited storage and bandwidth, the actual performance of accelerators can be severely degraded. Due to the significant lack of on-chip memory resources, the feature maps have to be moved back and forth between on-chip and off-chip memory, which brings a lot of pressure on the FPGA bandwidth. However, with the development of FPGA devices, on-chip memory capacity has shown a significant increase. Even the feature maps of recent large-scale networks can be completely stored on chip. This trend has brought the following changes to the design of CNN accelerators:

- Feature maps can exist on chip instead of being transformed repeatedly, which greatly reduces the bandwidth pressure and virtually shifts the bottleneck from bandwidth to computing resources on chip.
- The assumption that off-chip memory bandwidth is the constraining resource [8] in roofline model does not hold any more. The performance limitation in the situation of different memory capacity should be explored.

B. Computation Optimization

The red line in Fig. 1 shows the attainable performance diversity of the same networks between two platforms with different memory capacity in the roofline model. Although convolutional layers generally belong to the computational intensive computing, they can still be restricted by the bandwidth bottleneck with the increasing computation roof of hardware accelerators as the yellow triangle represents. This problem can be solved by improving the operational intensity with more memory, which will shift the location in the roofline model from the yellow triangle to the blue one. The operational intensity $OI_1$, $OI_2$ and their relationship can be expressed as

$$ OI_1 = \frac{\text{number of operations}}{\text{external data access}} = \frac{O}{(2 \times M_D + M_W)} \quad (1) $$

$$ OI_2 = \frac{\text{number of operations}}{\text{external data access}} = \frac{O}{M_W} \quad (2) $$

$$ \frac{OI_2}{OI_1} = \frac{2 \times M_D \times K + M_W}{M_W} \quad (3) $$

In these equations, $M_D$ represents the memory requirement for input or output features of one image. $M_W$ represents the memory requirement for weights and $O$ represents the total number of operations. $K$ represents the number of images the accelerator can compute in parallel. It shows the operational intensity of both convolutional layers and FC layers can be greatly improved to $\frac{2 \times M_D \times K + M_W}{M_W}$ with the memory increasing.

In the roofline model, we assume that all the computation resources are fully utilized. However, it’s almost impossible in a real accelerator. The PE efficiency must be considered when analyzing the actual computation performance. In a CNN hardware accelerator architecture, the total number of PEs $C$ can be represented as $C = m \times n \times K$. The accelerator may have several cores for parallel computing, the number of PE columns and rows in a single-core are represented as $m$ and $n$ while $K$ represents the number of cores in the accelerator. Besides, $M$ and $N$ represent the number of the input channels and the output channels. $M_{all}$ represents the total memory capacity on FPGA. With these factors, we can derive that

$$ \text{Computational roof} = \text{peak-performance} \times \text{PE-efficiency} $$

$$ = 2 \times C \times \left( \frac{M}{M_D} \right) \times \left( \frac{N}{M_W} \right) $$

where

$$ m \times n = \frac{C}{K} > C \times \frac{M_D}{(M_{all} - M_W)} \quad (4) $$

From these equations, we can observe that the computation roof is the function of $m$ and $n$. An accelerator can achieve the peak performance when $\frac{M}{M_D}$ and $\frac{N}{M_W}$ are both integers. However, different layers have different $M$ and $N$ which is hard for a fixed architecture to maintain the peak performance. In that case, a smaller $m$ and $n$ can improve the overall performance according to equation 4. Besides, the value of $m$ and $n$ is also restricted by $M_{all}$ as equation 5 shows. With the $M_{all}$ increases, beside the big performance leap we have already discussed above, the actual performance will further approach the computational bound from the green triangle to the red one in Fig. 1 for example. As for the FC layers, since multiple images share the same weights, the performance is related to $K$ directly.
IV. IMPLEMENTATION

A. System Architecture

The system architecture of the high-throughput CNN accelerator is shown in Fig. 2. It is mainly divided into three parts: PE array, on-chip BRAM/UltraRAM memory and Arithmetic Logic Unit (ALU) modules. The original images and weights are read from the DDR into the on-chip memory, and then finish the operations of convolutional layers or FC layers which are mapped in the PE array. The functions of the ALU modules are the accumulation of partial sums, RELU and the pooling operations. The results will then be written back to the UltraRAM memory. Each part has its own control module which ensures that the data can flow in the correct way provided by the instructions.

In this architecture, we use a 2-dimensional systolic array of PEs with pipelined registers between every row of PEs to execute the multiply-accumulate operations. PEs in each column perform the computation of two output features independently, while the PEs in the same row share common input features with different weights and bias. On one hand, it can achieve high performance by data reuse strategy and high parallelism. On the other hand, using one DSP instead of several PEs in series to process a kernel is more flexible and ensure 100 percent utilization when confronting various sizes of kernels. Meanwhile, as PEs in each row correspond to an input channel, the results can be directly connected down to the next row as the input of partial sums. By transferring the partial sums in series rather than accumulating them, only the PEs of the last row gets the effective results. In this way, it can decrease the number of ALU modules and the interface to memory from \( m \times n \) down to \( n \) where \( m \) means the row number and \( n \) means the column number. Since the number of input channels of most layers in CNNs is a multiple of 32 and the number of output channels is a multiple of 64 except the first layer whose proportion in CNNs are rarely small, \( m \) and \( n \) should be set as the times of 32 and 64 to maximize the utilization and performance.

B. PE unit

Fig. 3 illustrates the architecture of PE. Except the convolution operation, PE also has the functions of distance calculation and vector multiplication. Two DSPs are designed as the basic computation element to execute the multiply-accumulate computation or distance calculation in one PE. When the accelerator works in the distance calculation mode, one Xilinx DSP is unable to compute \((A - B)^2 + C\). We have to split the formula apart into two steps of \((A - B)\) and \((A - B) \times (A - B) + C\). Secondly, there are no broadcast data in vector multiplication while the input data channel is fixed to the broadcast mode to increase the data reuse. Under this circumstance, vectors will go through two weight channels into DSP2 and the input data channel will be closed. There are four kinds of interfaces of PEs. Among them the input features and the weights interfaces are 16 bit while the partial sums and the results are 48 bit for the accuracy. The function of DSP is \((A + D) \times B + C\) with 8-16 bit precision most of the time, except that the function of DSP1 is \((A - C)\) when the accelerator is in the distance calculation mode. The 16-bit input features can go through port A directly to perform a 16-bit multiply-add operation or be separated into two parts and then go through port A and port D to perform two 8-bit multiply-add operations.

C. Optimization of BRAM/UltraRAM memory

To further improve performance, we optimize the memory architecture to support parallel computing with more images. In our architecture, UltraRAM is used to store feature maps for its high capacity and BRAM is used to store weights and bias. UltraRAM can be divided into three areas for storing input/output data, partial sums and bypass data. The allocation of memory space is not fixed, but configurable. Considering that the data volume of these data is generally not the same, one of these three areas has a larger capacity for storing a larger amount of data, and the other two areas are relatively small. At the same time, our storage architecture stores the input and output data in the same block area. When part of the input data complete all computation and the final results of a layer are obtained, the output data are placed in the position originally belonging to the input data. This not only improves memory utilization, but also avoids the memory conflicting problem when the amounts of both input and output data exceed the capacity of the above two small areas.
D. The Interface Between Different Clock Domains

In fact, if DSPs work at the main frequency, the memory are already surplus even if all DSPs are used on Xilinx UltraScale+ FPGA for example. On the other side, although the frequency of memory is difficult to improve, the operating frequency of the DSP is far from the limit. To further enhance the performance and take full advantage of on-chip memory resources, we implement an interface between high frequency DSPs and other low frequency modules to compute multiple images with more UltraRAM and ALU modules. In this way, we approach the peak performance of our accelerator on FPGA.

V. EXPERIMENTAL RESULTS

We implement our architecture on Xilinx XC7VU9P FPGA which contains 6840 DSPs with 27 x 18 precision, 345.9Mb memory and two DDR3 memory. The main frequency is set to 100 MHz while the frequency of DSPs is 400 MHz. Then we synthesize and implement the proposed architecture in Vivado(2018.2). Two widely used CNN models, AlexNet and ResNet-50 are used to evaluate our accelerator with both 8-bit and 16-bit precisions.

Table II: Resource utilization

<table>
<thead>
<tr>
<th>Resource</th>
<th>FF</th>
<th>LUT</th>
<th>DSP</th>
<th>BRAM</th>
<th>UltraRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>2564880</td>
<td>1182280</td>
<td>6840</td>
<td>2160</td>
<td>960</td>
</tr>
<tr>
<td>Utilization</td>
<td>845111</td>
<td>360872</td>
<td>6130</td>
<td>1824</td>
<td>900</td>
</tr>
<tr>
<td>Percentage (%)</td>
<td>33.7</td>
<td>31.2</td>
<td>89.9</td>
<td>84.4</td>
<td>93.8</td>
</tr>
</tbody>
</table>

Resource utilization is shown in Table II which shows that nearly 90% of the on-chip resources are occupied. We implement a three-core architecture on the target device. Each core contains a 32x64 DSP array, making the utilization of DSP reach 89.8%. In this way, the utilization of UltraRAM can increase to 93.75%, which shows that FPGA storage capacity has gradually met the trend of accelerator demand. In this way, a 12-core parallel computing is realized to nearly make full use of on-chip resources.

We then make an overall comparison between our architecture and previous designs in Table I. Thereinto, KLUTs, DSPs, Memory represent the resource efficiency, GOPS and FPS based on a specific network represent the performance and throughput. As can be seen, our accelerator has a much higher performance than other accelerators and the peak performance exceeds all these accelerators [9][10][11][12][13]. We then evaluate the performance of each DSP with the same bit-width of 16. On ResNet-50, our accelerator exceeds [13] by 2.83x which proves our improvement is not just by using more DSPs. The main reason is these previous works were restricted by bandwidth bottleneck in convolutional layers and couldn’t make full use of computational resources in their target devices. As for AlexNet, compared to [11] which stores all the data on chip as well, the performance of each DSP still exceeds [11] by 2.13x because we increase the operational intensity of FC layers through our 12-core parallel computing.

VI. CONCLUSION

In this paper, we explore the design space of CNN hardware accelerator architecture with the recent development of FPGA on-chip memory capacity. Based on the roofline model, we analyze the optimization method for CNN accelerator designs depending on the available computing and memory resources. We then propose the architecture based on our optimization method which aims for high throughput CNN accelerators. The proposed accelerator is tested on XC7VU9P FPGA and achieves a peak performance of 9.39 TOPS and 6.86 TOPS with 100MHz main frequency and 400MHz DSP frequency for 8-bit data width on ResNet-50 and AlexNet respectively, which outperforms the previous designs.

REFERENCES