Reconfigurable FPGA-based CNN Accelerator using Module Scheduling: A Focus on ResNet

(Abstract) In deep learning, due to the complexity and inflexibility of FPGAs, implementing deep learning models on the FPGA is still challenging. This paper proposes a practical architecture to allow the implementation of different CNN models on the FPGA without multiple HDL compilations. The proposed architecture uses a scheduler that controls computational order of fundamental modules, such as convolution and pooling layers, to perform as if the CNN model was already implemented. This characteristic of our proposed architecture provides flexibility for the model. Moreover, the proposed architecture includes Multiply-Accumulate (MAC) units that employ the concepts of systolic array and zero look-ahead to mitigate data dependency bottlenecks.

1. INTRODUCTION

Recently, convolutional neural networks (CNNs) have distinguished themselves for their human-level performance in image detection and recognition tasks. Due to the large amount of computation for CNNs, many researchers or developers have started to use hardware accelerators, such as GPUs, ASICs, and FPGAs, to run models faster during training or inference. Compared to other accelerators, FPGA has shown impressive performance in parallel processing and energy efficiency [1]. However, it is difficult to execute deep learning models on FPGAs. Deep learning models involve complex operations, and implementing these operations on FPGAs requires significant efforts [2]. In addition, FPGAs are typically programmed using hardware description language (HDL), which is not appropriate in situations where models are changed or updated frequently [1].

In many researches, moreover, they have focused on optimizing a specific CNN model to run on FPGA or increasing throughputs for a specific CNN model by quantization, pruning, or increasing more processing elements (PEs) [3]. Other studies focus on reducing the size of data items between layers or DRAM and PEs to increase throughput or reduce the bottlenecks in data movement, such as compressed sparse column (CSC) matrices [4]. These researches are still difficult to support various CNN models that may be changed or updated frequently.

Motivated by this, this paper introduces a practical architecture to allow the implementation of different ResNet configurations on the FPGA without multiple compilation. This proposed architecture consists of fundamental modules such as convolution and pooling layers, controlled by instructions. The proposed architecture aims to efficiently optimize the diverse implementations of ResNet configuration on a single FPGA compilation, offering flexibility in constructing model structures and dynamically adapting based on user-sent instructions. This approach eliminates the need for individual compilations for each ResNet variation, promoting efficient and accessible deployment. To address data dependency bottleneck and accelerate convolution computation, the proposed architecture employs Multiply-Accumulate (MAC) units that involve the concepts of systolic array and zero look-ahead. In a simulated environment, our approach demonstrates compatibility with the original ResNet FPGA architecture, aligning with its computational expectations and constructs. Our experiment aims to present a practical architecture for FPGAs to address these issues, offering a more accessible and flexible architecture for deep learning model implementation on FPGAs, aiming to enhance the efficient use of FPGAs in current deep learning research.

2. THE PROPOSED ARCHITECTURE

2.1. Reconfigurable CNN accelerator

In the proposed architecture shown in figure 1 focuses on a flexible and dynamic configuration, avoiding the stiffness of traditional designs. It is designed to responsively operate on user-generated instructions, which include integration of fundamental modules such as Convolution (Conv), ReLU, and Pooling layer modules. This approach allows a non-linear, adaptable model construction, overcoming the restrictions of fixed sequence operations seen in conventional architectures.

Examining the established structure of ResNet-18, it simply consists of 18 layers of convolution with 8 groups of a residual block sequence (Conv-ReLU-Conv), which works either in a residual or linear manner. With the proposed architecture, the combinations and sequences of modules can work as if the model were already implemented on FPGA. For example, the residual block sequence consisting of 18 convolutional layers operates equivalently to ResNet-18, while one with 34 convolutional layers functions as ResNet-34. This implies that the proposed architecture can transition the model across different ResNet configurations, not limited to ResNet-18 or ResNet-34, all with a single compilation process.

2.2. Systolic Array

Figure 1 shows the overview of the proposed CNN accelerator architecture. The proposed architecture employs the systolic array that works in parallel and is strategically used here for its capability to perform repetitive calculations on multiple data sets efficiently [8]. By connecting processing elements (MAC) in a uniform manner, it adeptly replaces a pipeline structure with a programmable array of processing elements. This systolic array can improve the overall processing efficiency, which is a critical aspect of handling complex CNN operations

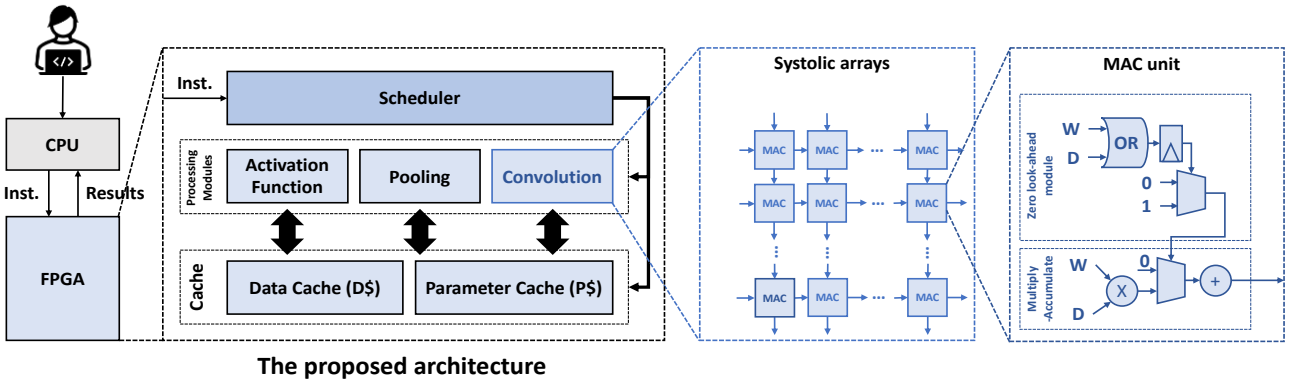

Figure 1. Overview of the proposed architecture and MAC unit.

2.3 Zeros Look-ahead based MAC Unit

Figure 1 further details our processing elements termed as MAC (Multiply-Accumulate operation). The basic MAC contains elements of Input/kernel data register, multiplier, and adder. Observing a data dependency bottleneck at Systolic array architecture, we apply the prefetching data register for reducing this problem. As processed images pass through several CNN layers, an abundance of zero data emerges. The propagation of zero data, in this setup, contributes to making our processor faster and more efficient. The integration of prefetching memory to our processor allows the application of a ZD (zero detector) to the previous MAC by MUX. Upon the detection of zero data in the prefetching zone, a MAC unit propagates a zero signal to the subsequent MAC unit. This mechanism ensures the immediate output of a return value of zero upon the reception of a zero signal, optimizing efficiency and speed in the MAC operations.

3. EVALUATION

3.1. Experimental setup

To demonstrate simulation results of validation for the proposed MAC unit, we use a simple image to compare CNN results between two units one is the naive convolution (GPU by Python) , and other is our proposed MAC units and architecture. For simulation and evaluation, the Intel Modelsim Software 2020.01 version/Pytorch 1.13.1 are employed, ensuring detailed and reliable results.

3.2. Experimental results

The verification of the two architecture designs is depicted in Figure 2. This figure provides a comprehensive view of the MAC and error counts between Pytorch result and MAC result. When compared to the Python results, we were able to confirm the successful validation, as it showed an error of zero, indicating that each of the three different filters had the same values.

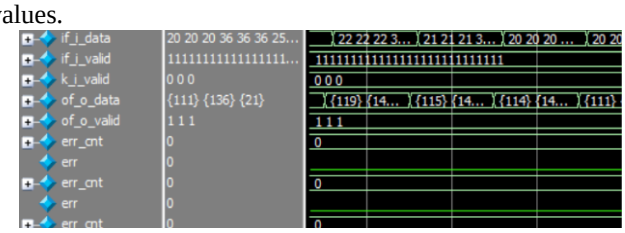

Fig. 2. Modelsim report for validation

4. CONCLUSION

This paper introduces a practical architecture for CNN accelerators on FPGAs. The proposed architecture can avoid multiple HDL compilations by using a scheduler that controls the computational orders of modules to perform as if the CNN model was already implemented on the FPGA. This flexibility enhances the ease and efficiency of the model deployment for FPGAs. By integrating fundamental modules with our MAC design, the architecture adeptly navigates the challenges of data dependency bottlenecks and improves the computational speed and efficiency. Tested in simulated environments, the proposed model has demonstrated compatibility with conventional ResNet FPGA architectures.

In summary, this work contributes a practical idea, a step forward in the field of FPGA-based CNN accelerators, supporting the efficient and adaptive implementation of deep learning models in diverse applications. The insights and results shared in this paper emphasize the potential for ongoing innovation and enhancement in the performance and flexibility of FPGA-based neural network implementations.

REFERENCES

[1] A. Shawahna et al. "FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review," in IEEE Access, vol. 7, pp. 7823-7859, 2019

[2] Khan et al. "A survey of the recent architectures of deep convolutional neural networks." Artificial intelligence review 53 (2020): 5455-5516.

[3] Chen, Yu-Hsin, et al. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks." IEEE journal of solid-state circuits 52.1 (2016): 127-138.

[4] Han, Song, et al. "EIE: Efficient inference engine on compressed deep neural network." ACM SIGARCH Computer Architecture News 44.3 (2016): 243-254.

[5] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[6] Schluter, D. *The Ecology of Adaptive Radiation* (Oxford Univ. Press, 2000)

[7] Plazzo, Anna Pia et al. “Bioinformatic and mutational analysis of channelrhodopsin-2 protein cation-conducting pathway.” The Journal of biological chemistry vol. 287,7 (2012)

[8] C. Bagavathi MTech, in Deep Learning and Parallel Computing Environment for Bioengineering Systems, 2019, Pages 207-223