A CNN ACCELERATOR WITH EMBEDDED RISC-V CONTROLLERS

Li Zhang, Xian Zhou and Chuliang Guo*
College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou 310000, China
*Corresponding Author's Email: andyzhang926@zju.edu.cn

ABSTRACT

Convolutional Neural Network is a promising technology in machine learning. Due to its vast computing and data requirements, it needs to be run with a specific accelerator to achieve reasonable energy efficiency. Improving the performance of accelerators has become the research hotspot. A mixed-precision structure can be used to improve hardware utilization, thus reducing area and power. However, the mixed-precision flow control is so complicated that it costs too much hardware resources. In this paper, a CNN accelerator with embedded RISC-V controllers is introduced to achieve flexible control at a very low cost. The ASIC synthesized results show that the proposed design area with two embedded cores is 5% less than the basic design.

INTRODUCTION

Convolutional neural network (CNN) has been increasingly deployed in various deep learning applications, from computer vision, to speech recognition and natural language processing [1-3]. But devices that run CNN are commonly constrained by lower computation and storage resources and demand more efficient hardware implementations to ensure reduced storage size, faster inference and higher energy efficiency. Although GPU may easily provide significant performance, its power consumption limits its broader applications. Instead, many researchers consider FPGA and ASIC as more promising alternatives to implement low power or energy-efficient CNN accelerators. However, as CNN size and complexity continues growing, there has been substantial progress in designing light weight CNN architecture and better speed-accuracy tradeoff through innovations in hardware implementations.

Among various efforts, quantization for multipliers, which is the most frequent operation in a CNN, has become an active research topic [4-7]. An operand with smaller bit-width, i.e., lower precision, in a CNN, may help reduce energy consumption not only in computation but also transmission. Thus, it is a very appealing option for hardware architects and designers to design the processing element (PE), i.e., the basic functional unit of a CNN, with the desired and optimized precision. Reference [4] proposes to use two 8-bit multipliers in a PE to switch between 8- and 16-bit precisions. ENVISION in [5] presents a booth multiplier based dynamic voltage accuracy frequency scaling technology that can be configured to 4-, 8- or 16-bits. Furthermore, UNPU [6] uses serial multipliers to implement lookup table-based PE to enable precisions from 1 to 16 bits. In short, at the cost of additional area for PE, control logics and storage, the reconfigurability can provide different precisions to different neural networks or different layers in one neural network, thereby improving overall energy saving. However, the precision control of all the prior work is at most at the granularity of layer-wise. In other words, within the same layer, all the operations use the same precision and bit-width.

[7] proposes a general-purpose CNN accelerator architecture with fine granular mixed-precision support to address the challenges mentioned above. It uses two independent PE arrays to handle high-precision and low-precision calculations, respectively. But its flow control is quite complicated and consumes a lot of hardware. It does not have a good effect of reducing the area but limits the flexibility. In this paper, we propose a general-purpose CNN accelerator architecture with a dual RISC-V core controller to improve control flexibility and hardware utilization. The contributions of this work can be summarized as below:

- **Universal accelerator architecture**: The CNN computations are separated into two groups, full and low precision. We use two-level cores to fine-grained control their flow. The complete core controls the whole accelerator, and the simplified one controls the logic-complicated part in the accelerator. Such an architecture is applicable to different CNNs to support simultaneous computations using multi-blocks.
- **A reconfigurable control logic using minimalist hardware**: A unique logic is designed to control the queued operations for the full precision processing element (FPPE). Its hardware implementation and operation flow are straightforward but with a high degree of flexibility

BACKGROUND

RISC-V is a modular instruction set architecture (ISA) [8]. For achieving a good hardware utilization, designer can select instruction subsets in RISC-V according to their requirement. A small RISC-V implementation which called "PicoRV32" needs only 7000 LUTs on FPGA [9]. The PicoRV32 can be used as a microcontroller in a block-designed system.

Since the advent of Tensor Processing Unit (TPU), it has been a popular research topic to design a universal accelerator architecture that can support different neural

networks for different applications. A commonly-used accelerator consists of a PE array, a global buffer (GLB) and additional control logics [4-7]. The PE is the basic computing unit in the accelerator, which contains memory blocks and multiply-and-accumulate (MAC) units. The role of GLB is to temporarily store data for the neural network, such as input feature maps (IFmaps), partial sums (Psums), weights and bias.

At the beginning of the accelerator operation, all the IFmaps and weights are stored in DRAM. According to the convolution process, data is transferred from DRAM to GLB in order. Then, the data in GLB is assigned to a PE through the data bus. Finally, the calculated results of a PE are transferred back to GLB through the data bus. Such a process is repeated until all the operations of a neural network layer is completed. Apparently, most energy consumption comes from the repeated data movement and computations. Thus, a higher bit-width of precision inevitably leads to more energy consumption in both transmission and computation.

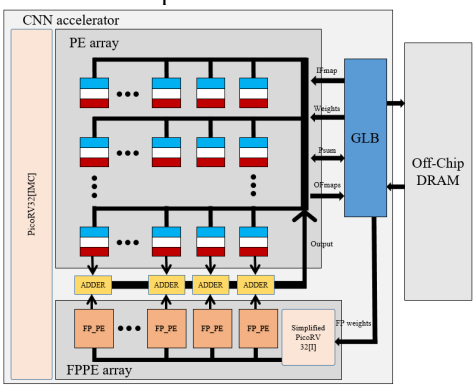

Fig. 1. Architecture of the proposed CNN accelerator with dual RISC-V controller.

PROPOSED ARCHITECTURE

Architecture Overview

The proposed architecture is illustrated in Fig. 1. Its main components are described below:

• PE array's structure and data flow are similar to that in [4]. The difference is the MACs and storage here are 8-bit precision. This PE array only runs the 8-bit weights calculations.

• The 16-bit weights calculations are done by FPPEs. The FPPE contains a 16-bit MAC, 16-bit storage, IFmap receiver and some logic for decode the weights index.

• The Adders between PE array sum the output of PEs and FPPEs, then send back the result to GLB.

• GLB is a group of SRAM that temporarily stores intermediate data of calculations.

• PicoRV32[IMC] is a relatively complete controller. It has I, M, C instruction subsets, and can runs the driver code for the entire accelerator. With this embedded processor, the workload of the external processor can be greatly reduced.

• Simplified PicoRV32[I] is a minimized PicoRV which only has I instruction subset. And its interfaces, memory and fabrics are also removed or compressed to reduce its area. It only does the configuration and control for FPPEs.

In this work, the two PicoRV controllers are designed to handle the whole accelerator and the FPPEs respectively to achieve finer granular operation. However, we can always employ similar techniques in [4-7] to support more diversified operation in each module, as the proposed techniques are general.

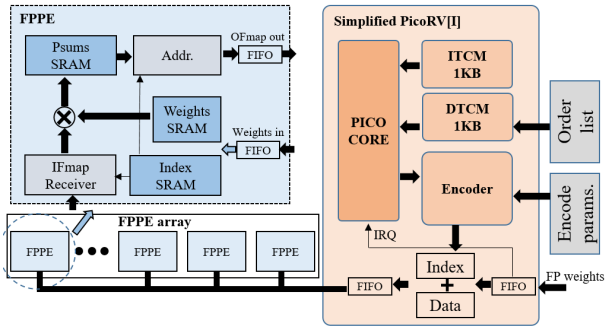

Fig. 2. Architecture of the Simplified PicoRV and FPPE

Simplified PicoRV32 processor

The simplified PicoRV32 controller is shown in the right part of Fig.2. The instruction tightly coupled memory (ITCM) is set to as small as 1KB because it only runs RAM load/save and interrupt requirement (IRQ) response. The data tightly coupled memory (DTCM) only stores a few data, so it can also be set to less than 1KB.

Its workflow is described in Fig.3: When a calculation pass starts, the pre-compiled FP weights order list and encode parameters of this pass are sent to DTCM by PicoRV. Then the simplified PicoRV configures the Encoder with the parameters and puts the first order number into the Encoder. The Encoder is a group of MACs, and it combines the order number and parameters to obtain the Index of FP weight data. When an FP weight occurs, the input FIFO's valid signal triggers the IRQ and reads the encoded index. Then the simplified PicoRV put a new number into Encoder. This process loops until the pass over.

In the previous design, data flow and de/encoding are controlled by fixed logic. In order to achieve functional coverage, the design needs to be very complex and difficult to configure. In this paper, the task of this simplified PicoRV is fairly simple, but it can be changed at any time. You just need to put the compiled code into its ITCM, or even compile its task code on the spot in another complete PicoRV. Therefore, this design achieves high flexibility with minimal hardware complexity.

FPPE

The structure of FPPE is shown in the left part of Fig.2. Compares with [7], we removed lots of parameters

storage and encoding logic. Because the encoding is done by the simplified PicoRV. The FP data and encoded index are sent to FPPE and stored in SRAM. According to the index, the module fetches the corresponding IFmap in the IF data flow. Then the IFmap data is then multiplied by the FP weight data, the product of them becomes the partial sum and save into Psums SRAM. After the MACs operation in PE array is finished, the Psums SRAM data is added to PE array result and sent back to GLB. Psums data's relative address is also extracted from the index.

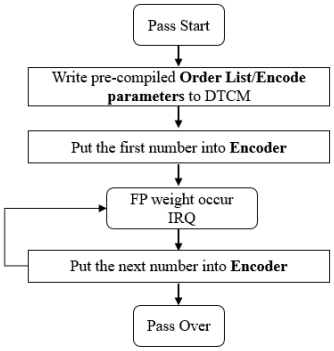

Fig. 3. Code execution flow of Simplified PicoRV.

EXPERIMENTAL RESULTS

We implemented our design using Xilinx ZCU102 FPGA and UMC 40nm ASIC synthesis, respectively. The method in [4] and [7] are also implemented for comparison. In terms of running the neural network, the execution speed of FPPE is related to the proportion of FP data in the weight, and the ratio of FP is associated with the quantification scheme. If we quantify the FP ratio to below 10%, FPPE can quickly finish the task and not become the bottleneck of speed. In the PE array, we use 8-bit MACs, which can undoubtedly run at a higher frequency than the 16-bit MAC. However, it cannot perform the speed advantage well on FPGA, and ASIC is difficult to complete such large-scale simulation, so we cannot objectively compare the three designs' network speed here.

In terms of hardware scale, the advantages of our design can be seen intuitively. Table 1 shows the resource comparison between [4, 7] and this work. The simplified PicoRV only use 4100 LUTs in FPGA and 23k um^2(include TCMs) in ASIC. Replacing 16-bit PE with 8-bit PE resulted in an area reduction of nearly 30%. However, because of the complex control logic, FPPE in [7] has a large area. This work adds a simplified PicoRV to control the FPPE flow so that reduce the area of FPPE is greatly reduced.

CONCLUSION

This work proposed a CNN accelerator architecture with mixed precision processing elements. Two RISC-V core controllers are used to control the flow, one of which is a simplified core with an area of only 14k μm^2. Due to the simplified controller, the accelerator area is reduced by 17% and 5%, respectively, compared with that in [4] and [7].

TABLE 1 HARDWARE AMOUNT COMPARISON
(unit: FPGA-LUTs in logic, Byte in block RAM, ASIC-μm^2)

component	Ref.[4]		Ref.[7]		This work	
	FPGA	ASIC	FPGA	ASIC	FPGA	ASIC
PEs(x168)	265k	2452k	152k	1764k	152k	1764k
--MAC	280	1.6k	128	0.9k	128	0.9k
--storage	608B	10.8k	384B	7.8k	384B	7.8k
--logic	1313	2.2k	777	1.8k	777	1.8k
FPPEs(x14)	-	-	37k	377k	25k	200k
--MAC	-	-	280	1.6k	280	1.6k
--storage	-	-	1.2kB	22k	0.6kB	11k
--logic	-	-	2.4k	3.3k	1.5k	1.7k
PicoRV	-	-	-	-	7k	68k
--logic	-	-	-	-	7k	18k
--TCMs	-	-	-	-	4kB	50k
SimpPicoRV	-	-	-	-	4k	23k
--logic	-	-	-	-	4k	11k
--TCMs	-	-	-	-	2kB	12k
Total	265k	2452k	189k	2141k	188k	2055k

ACKNOWLEDGMENTS

This work was supported in part by Key Area R&D Program of Guangdong Province (Grant No. 2018B030338001).

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Proc. CVPR, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun,"Deep residual learning for image recognition," Proc. CVPR, 2016.

[3] J. Deng, Z. Shi, and C. Zhuo, "Energy Efficient Real-Time UAV Object Detection on Embedded Platforms," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 39(10): 3123-3127, 2020.

[4] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE J. Solid-State Circuits, vol. 52(1):127–138, 2017.

[5] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10 TOPS/W subword-parallel dynamic voltage accuracy frequency scalable convolutional neural network processor 28 nm FDSOI," Proc. ISSCC, 2017.

[6] J. Lee, C. Kim and et al, "UNPU: An Energy-Efficient Deep Neural Network Accelerator with Fully Variable Weight Bit Precision," IEEE Journal of Solid-State Circuits, VOL. 54(1), pp. 173–185, 2019.

[7] X. Zhou, L. Zhang and et al, "A Convolutional Neural Network Accelerator Architecture with Fine-Granular Mixed Precision Configurability," 2020 IEEE International Symposium on Circuits and Systems, 2020.

[8] RISC-V International, https://riscv.org/.

[9] PicoRV, https://github.com/cliffordwolf/picorv32.