Combining Reconfiguration and Instruction Computations with an Asynchronous Crossbar

Pengfei Li, Mingbing Liu, Lvying Yu, Anping He and Jinzhao Wu

ABSTRACT

With the rapid development of artificial intelligence (AI) technology, a more flexible and efficient computing architecture is required. The coarse-grained reconfigurable architectures (CGRAs) as a co-processor that combines the flexibility and specificity of AI would be an appropriate solution. In this paper, we introduce an innovative asynchronous CRGA architecture that combines reconfiguration and specific instructions. We aim to accelerate a common computation with memorized calculation and keep the reconfiguration for rare ones, so that the whole efficiency could be enhanced. To show the feasibility of our method, we choose the sigmoid function that is essentially and widely used in AI algorithms, which can achieve about 40Mhz configuration throughput and about 669ns to finish the whole calculation.

KEYWORDS
CGRA, Asynchronous, FPGA, Function oriented.

INTRODUCTION

With the rapid development and application of artificial intelligence technology, the traditional CPU is flexible enough to support various applications but may not provide sufficient performance to cope with their complexity. Moreover, these algorithms have the characteristics of data-intensive computation of high complexity. Many dedicated AI chips have been proposed to speed up related algorithms but these chips are only for a specific application [1][2][3]. The reconfiguration computing that is first proposed by Estrin in 1963 is a processor architecture that combines flexibility and specificity [4]. It is generally divided into fine-grained reconfigurable architectures (FGRAs) and coarse-grained reconfigurable architectures (CGRAs). FPGA as a representative of fine-grained reconfigurable architecture has been applied to accelerate artificial intelligence algorithms. However, the reconfiguration of FPGA is completed at the underlying structure, so that it would bring too much configuration information. Comparing with FPGA, the CGRAs greatly reduces the delay, area, power and reconfiguration time but at the expense of flexibility.

Pengfei Li, Mingbing Liu, Anping He, School of Information Science and Engineering, Lanzhou University, Gansu 730000, China
Lvying Yu, Xinnuo Pusi Technology (Wuhan) Co., Ltd.
Jinzhao Wu, Guangxi Key Lab of Hybrid Computation and IC Design Analysis, Nanning, China
A lot of work has been done to design a better structure for CGRAs and how to design and configure the network to achieve reconfiguration has become the focus of this research [5][6]. Most reconfigurable systems contain a big reconfigurable matrix and a processor for executing the entire application. In this design, we consider a reconfigurable architecture that combines reconfiguration and specific instructions. We aim to accelerate a common computation with memorized calculation and keep the reconfiguration for rare ones, so that the whole efficiency could be enhanced. However, considering that the reconfiguration architecture that achieve a variety of functions would have different data processing speeds, a method of data-driven that is realized by asynchronous design is applied to this design.

ARCHITECTURE

There are three parts consisting in a typical coarse-grained reconfigurable architecture, and they are microprocessor, a reconfigurable array (RA) and their interface, respectively. In this design, a RA that is designed with an asynchronous approach is places beside the processor as a function unit. In this case, we could use a kind of instructions to control the asynchronous reconfiguration array to perform specific functions as if it was co processing. In addition, a configuration information for it needs to be stored beforehand in order to reconstruct for different functions. Facing the feature of artificial intelligence algorithms that have the feature of high concurrency and the storage and reading for large data, the ports of the asynchronous RA can be used as input ports and output ports to help read and store data from the memory in parallel. The asynchronous RA gets the control signal and the information where it can read the instructions, data and configuration information from the processor. The instructions that come from the memory are used to realize making the asynchronous RA to execute specific functions, and the configuration information control transmission of data between different processing elements. By the way of the using configuration information together with instructions, a more flexible reconfigurable way can be realized.

REALIZATION

In this article, we choose a well-studied click controller [7] and design a circuit that follows link-joint model. By this way, a network is realized. Firstly, we need to configure the network. Then, the arrival of asynchronous handshake signals and data triggers the network to forward data.

A controller module is needed for controlling network. The task of the controller is to check whether all PEs finished their works, choose a configuration information queue to output and start the next time reconfiguration and data transmission. To help solve the above problems and increase the flexibility of the reconfiguration element, we use a series of instruction to control the controller module.
As shown in the Figure 1, the block of our controller module can be seen. The process consists of fetching, decoder, reconfiguration and transmitting data. The instruction is firstly fetched from memory and be sent to the instruction register. Secondly, it is time to divided the instruction into Flags of queue, Flags of PEs and offset of address. It can be seen as a kind of decoder. The signal of In_sequence, In_loop and In_jump is sent to help choose the configuration information to configure the network. The flags of the PEs are sent to judgement module to collect the end signal from PEs. The judgement module could be seen as a conditional trigger that can be triggered when all the conditions are met. The offset of address is stored in a reg. At the point of start of jump or end of loop, an instruction jump might occur. Whether instructions jump or not is determined by the result of a logical calculation. Many traditional logic operations can be implemented by this PE, such as comparison, counting, etc. According to the judgement result, the address of the next instruction can be correctly selected. When all the PEs have finished their work, the next time fetching will begin. As shown in the Figure 2, an asynchronous control chain that can be executed circularly is designed. The start-up, an asynchronous handshake signal would start the asynchronous control chain and control the module to start running. The fire_1 is used to fetch instruction, and the fire_2 is used to decode it. Next, the signal out_configuration would make the horn net change the transmission of data path, and the signal out_start would start the transmission. When all the PEs finished their work, the judgement module would set up handshake signal with click_2. And the click_2 would restart the asynchronous control chain to carry out a new cycle.

Figure 1. Block diagram of controller.
EXPERIMENTAL RESULT

In this design, a function sigmoid is mapped to the reconfiguration element and two ports are chosen as input and output. The design would contain an exponential calculation module, an addition calculation, a division calculation module and a 2x2 data transmission network to support reconfiguration. Besides, a stack module is also needed in this design to store intermediate data. 32-fixed points are designed for calculation. And we use 1-bit for sign bit, 16 bits for integer part and 15 bits for decimal part. In order to realize the exponential calculation, we use the method of look-up table and Taylor formula to get the approximate value. The calculation process of sigmoid can be divided into four steps as shown in Figure 3. Firstly, the data will be sent to the exponential module to get the result. Secondly, data that the numerical value is 1 is sent from the input port and do addition with the calculation results of exponential module. Thirdly, another data with the value also is 1 is sent to the division unit to do division with the results of the previous step. Lastly, the result would be sent out from the output port. And this process contains four times reconfigurations to finish the computation of this function.
In our experiment, the proposed circuits are designed by Verilog-1995 and implemented by Xilinx FPGA of VC707(xc7vx4851157-1). And we can see the specific resource consumption as shown in the following Table I.

| Name         | Sum | Controller | Addr | Exp  | Div |
|--------------|-----|------------|------|------|-----|
| Delay        | 669ns | 24ns      | 10ns | 33ns | 369ns |
| LUT Used     | 1839 | 79         | 195  | 240  | 152  |
| FF Used      | 1251 | 32         | 76   | 92   | 236  |
| DSP Used     | 12   | 0          | 0    | 0    | 12   |

CONCLUSION

CGRAs are considered to be appropriate for executing AI algorithms because it can satisfy both flexibility and high performance. In this paper, we introduce a coarse-grained design and implemented in Xilinx FPGA, which can accelerate a common computation with processing elements (PEs) and keep the reconfiguration for rare ones, so that the whole efficiency could be enhanced. We still continue this type of design, the future works would focus on optimization of the whole architectures, especially the storage structure of configurable information and intermediate variables.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under Grant No. 61772006, the Science and Technology Program of Guangxi under Grant No. AB17129012, the Science and Technology Major Project of Guangxi under Grant No. AA17204096, the Special Fund for Scientific and Technological Bases and Talents of Guangxi under Grant No. AD16380076, and the Special Fund for Bagui Scholars of Guangxi.

REFERENCES

1. Liu S, Du Z, Tao J, et al. Cambricon: An instruction set architecture for neural networks International Symposium on Computer Architecture. Piscataway NJ: IEEE, 2016: 393-405.
2. Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. Fiber, 2015, 56(4): 3-7.
3. TrueNorth: Design and Tool Flow of a 65mW 1 Million Neuron Programmable Neurosynaptic Chip. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 34(10):1537-1557.
4. Estrin G, Bussell B,Turn R,et al. Parallel processing in a restructure computer system. IEEE Transactions on Electronic Computing, 1963, EC-12(6):747-755.
5. Hartenstein, Reiner. "A Decade of of Reconfigurable Computing: a Visionary Retrospective." Proc of the Conference on Design 2001.
6. Mei, Bingfeng, et al. "Design Methodology for a Tightly Coupled VLIW/Reconfigurable Matrix Architecture: A Case Study." Design, Automation Test in Europe Conference & Exhibition IEEE, 2004.

7. Peeters A, Beest F T, Wit M D, et al. “click Elements: An Implementation Style for Data-Driven Compilation,” Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits & Systems, pp. 3-14, 2010.